UNIVERSIDADE DA CORUÑA
Departamento de Computación

TESIS DOCTORAL

INTERPRETACIÓN TABULAR DE AUTÓMATAS PARA LENGUAJES DE ADJUNCIÓN DE ÁRBOLES

Autor: MIGUEL A. ALONSO PARDO

Directores: MANUEL VILARES FERRO
ERIC VILLEMONTE DE LA CLERGERIE

25 de septiembre de 2000

Dr. Manuel Vilares Ferro
Profesor Titular de Universidad
Departamento de Computación
Universidad de La Coruña
España

Dr. Eric Villemonte de la Clergerie
Chargé de Recherche
INRIA
Francia

CERTIFICAN:

Que la memoria titulada "Interpretación tabular de autómatas para lenguajes de adjunción de árboles" ha sido realizada por D. Miguel A. Alonso Pardo bajo nuestra dirección en el Departamento de Computación de la Universidad de La Coruña y concluye la Tesis que presenta para optar al grado de Doctor.

La Coruña, 31 de marzo de 2000

Dr. Manuel Vilares Ferro
Director de la Tesis Doctoral

Dr. Eric Villemonte de la Clergerie
Director de la Tesis Doctoral

Tesis Doctoral:	INTERPRETACIÓN TABULAR DE AUTÓMATAS PARA LENGUAJES DE ADJUNCIÓN DE ÁRBOLES
Autor:	D. Miguel A. Alonso Pardo
Directores:	Dr. Manuel Vilares Ferro Dr. Eric Villemonte de la Clergerie
Fecha:	25 de septiembre de 2000

Tribunal

Presidente: Josep Miró (Universitat de les Illes Balears, España)

Vocal 1º: José Mira Mira (UNED, España)

Vocal 2º: Pierre Boullier (INRIA, Francia)

Vocal 3º: Mark-Jan Nederhof (DFKI, Alemania)

Secretario: Antonio Blanco Ferro (Universidade da Coruña, España)

Calificación: Sobresaliente Cum Laude

A Adriana

Agradecimientos

Deseo expresar mi gratitud a todos los que han contribuido a crear un agradable ambiente de trabajo para el desarrollo de esta tesis, tanto en la Facultad de Informática de La Coruña como en el centro de investigación del INRIA[1] en Rocquencourt, Francia. En La Coruña, destacar la ayuda prestada por los miembros que a lo largo del tiempo ha tenido el grupo de investigación COLE[2], con mención especial a David Cabrero, Jorge Graña y Manuel Vilares, paciente director de esta tesis. En el INRIA, destacar la paciencia de mi compañero de despacho, Frédéric Tendeau, y la colaboración del resto de componentes del proyecto Atoll[3], con mención especial a Pierre Boullier, por sus valiosos comentarios, Eric de la Clergerie, por haber aceptado codirigir esta tesis, y Bernard Lang, por haber aceptado ser el director de mi estancia.

Fructíferas han sido también las discusiones con Tilman Becker, Vicente Carrillo, Víctor Díaz, Patrice Lopez, Mark-Jan Nederhof y Giorgio Satta, así como los comentarios aportados por numerosos revisores anónimos de las publicaciones resultantes del trabajo realizado en esta tesis.

Especialmente importante ha sido el apoyo prestado por Adriana Dapena durante todo este tiempo.

Agradezco también a la Xunta de Galicia la financiación de parte de la investigación mediante una beca predoctoral, a Caixa Galicia la financiación de mi estancia en Francia y a la Universidade da Coruña la financiación parcial de los desplazamientos a congresos.

El proyecto de investigación en el que se enmarca la realización de esta tesis ha sido financiado, en el periodo en que he participado, en sucesivas fases por los fondos FEDER de la Unión Europea (proyecto 1FD97-0047-C04-02), la Xunta de Galicia (proyectos XUGA10501B93, XUGA20403B95, XUGA10505B96, XUGA20402B97, PGIDT99XI10502B), los Ministerio de Educación y Cultura y Asuntos Exteriores (acciones integradas HF 96-36 y HF 97-223) y el Centro Ramón Piñeiro para a Investigación en Humanidades.

[1] Institut National de Recherche en Informatique et en Automatique (`http://www.inria.fr/`).
[2] Compiladores y Lenguajes (`http://coleweb.dc.fi.udc.es/`).
[3] Atelier d'Outils Logiciels pour le Langage Naturel (`http://atoll.inria.fr/`).

Resumen

Las gramáticas de adjunción de árboles son una extensión de las gramáticas independientes del contexto que utilizan árboles en vez de producciones como estructuras elementales y que resultan adecuadas para la descripción de la mayor parte de las construcciones sintácticas presentes en el lenguaje natural. Los lenguajes generados por esta clase de gramáticas se denominan lenguajes de adjunción de árboles y son equivalentes a los lenguajes generados por las gramáticas lineales de índices y otros formalismos suavemente dependientes del contexto.

En la primera parte de esta memoria se presenta el problema del análisis sintáctico de los lenguajes de adjunción de árboles. Para ello, se establece un camino evolutivo continuo en el que se sitúan los algoritmos de análisis sintáctico que incorporan las estrategias de análisis más importantes, tanto para el caso de las gramáticas de adjunción de árboles como para el caso de las gramáticas lineales de índices.

En la segunda parte se definen diferentes modelos de autómata que aceptan exactamente los lenguajes de adjunción de árboles y se proponen técnicas que permiten su ejecución eficiente. La utilización de autómatas para realizar el análisis sintáctico es interesante porque permite separar el problema de la definición de un algoritmo de análisis sintáctico del problema de la ejecución del mismo, al tiempo que simplifica las pruebas de corrección. Concretamente, hemos estudiado los siguientes modelos de autómata:

- Los autómatas a pila embebidos descendentes y ascendentes, dos extensiones de los autómatas a pila que utilizan como estructura de almacenamiento una pila de pilas. Hemos definido nuevas versiones de estos autómatas en las cuales se simplifica la forma de las transiciones y se elimina el control de estado finito, manteniendo la potencia expresiva.

- La restricción de los autómatas lógicos a pila para adaptarlos al reconocimiento de las gramáticas lineales de índices, obteniéndose diferentes tipos de autómatas especializados en diversas estrategias de análisis según el conjunto de transiciones permitido.

- Los autómatas lineales de índices, tanto los orientados a la derecha, adecuados para estrategias en las cuales las adjunciones se reconocen de manera ascendente, los orientados a la izquierda, aptos para estrategias de análisis en las que las adjunciones se tratan de forma descendente, como los fuertemente dirigidos, capaces de incorporar estrategias de análisis en las cuales las adjunciones se tratan de manera ascendente y/o descendente.

- Los autómatas con dos pilas, una extensión de los autómatas a pila que trabaja con una pila maestra encargada de dirigir el proceso de análisis y una pila auxiliar que restringe las transiciones aplicables en un momento dado. Hemos descrito dos versiones diferentes de este tipo de autómatas, los autómatas con dos pilas fuertemente dirigidos, aptos para describir estrategias de análisis arbitrarias, y los autómatas con dos pilas ascendentes, adecuados para describir estrategias de análisis en las cuales las adjunciones se procesan ascendentemente.

Hemos definido esquemas de compilación para todos estos modelos de autómata. Estos esquemas permiten obtener el conjunto de transiciones correspondiente a la implantación de una determinada estrategia de análisis sintáctico para una gramática dada.

Todos los modelos de autómata pueden ser ejecutados en tiempo polinomial con respecto a la longitud de la cadena de entrada mediante la aplicación de técnicas de interpretación tabular. Estas técnicas se basan en la manipulación de representaciones colapsadas de las configuraciones del autómata, denominadas ítems, que se almacenan en una tabla para su posterior reutilización. Con ello se evita la realización de cálculos redundantes.

Finalmente, hemos analizado conjuntamente los diferentes modelos de autómata, los cuales se pueden dividir en tres grandes grupos: la familia de los autómatas generales, de la que forman parte los autómatas lineales de índices fuertemente dirigidos y los autómatas con dos pilas fuertemente dirigidos; la familia de los autómatas descendentes, en la que se encuadran los autómatas a pila embebidos y los autómatas lineales de índices orientados a la izquierda; y la familia de los autómatas ascendentes, en la que se enmarcan los autómatas a pila embebidos ascendentes, los autómatas lineales de índices orientados a la derecha y los autómatas con dos pilas ascendentes.

Abstract

Tree adjoining grammars are an extension of context-free grammars that use trees instead of productions as the primary representing structure and that are considered to be adequate to describe most of syntactic phenomena occurring in natural languages. These grammars generate the class of tree adjoining languages, which is equivalent to the class of languages generated by linear indexed grammars and other mildly context-sensitive formalisms.

In the first part of this dissertation, we introduce the problem of parsing tree adjoining grammars and linear indexed grammars, creating, for both formalisms, a continuum from simple pure bottom-up algorithms to complex predictive algorithms and showing what transformations must be applied to each one in order to obtain the next one in the continuum.

In the second part, we define several models of automata that accept the class of tree adjoining languages, proposing techniques for their efficient execution. The use of automata for parsing is interesting because they allow us to separate the problem of the definition of parsing algorithms from the problem of their execution. We have considered the following types of automata:

- Top-down and bottom-up embedded push-down automata, two extensions of push-down automata working on nested stacks. A new definition is provided in which the finite-state control has been eliminated and several kinds of normalized transition have been defined, preserving the equivalence with tree adjoining languages.
- Logical push-down automata restricted to the case of tree adjoining languages. Depending on the set of allowed transitions, we obtain three different types of automata.
- Linear indexed automata, left-oriented and right-oriented to describe parsing strategies in which adjuntions are recognized top-down and bottom-up, respectively, and strongly-driven to define parsing strategies recognizing adjunctions top-down and/or bottom-up.
- 2–stack automata, an extension of push-down automata working on a pair of stacks, a master stack driving the parsing process and an auxiliary stack restricting the set of transitions that can be applied at a given moment. Strongly-driven 2–stack automata can be used to describe bottom-up, top-down or mixed parsing strategies for tree adjoining languages with respect to the recognition of the adjunctions. Bottom-up 2-stack automata are specifically designed for parsing strategies recognizing adjunctions bottom-up.

Compilation schemata for these models of automata have been defined. A compilation schema allow us to obtain the set of transitions corresponding to the implementation of a parsing strategy for a given grammar.

All the presented automata can be executed in polynomial time with respect to the length of the input string by applying tabulation techniques. A tabular technique makes possible to interpret an automaton by means of the manipulation of collapsed representation of configurations (called items) instead of actual configurations. Items are stored into a table in order to be reused, avoiding redundant computations.

Finally, we have studied the relations among the different classes of automata, the main difference being the storage structure used: embedded stacks, indices lists or coupled stacks. According to the strategies that can be implemented, we can distinguish three kinds of automata: bottom-up automata, including bottom-up embedded push-down automata, bottom-up restricted logic push-down automata, right-oriented linear indexed automata and bottom-up 2–stack automata; top-down automata, including (top-down) embedded push-down automata, top-down restricted logic push-down automata and left-oriented linear indexed automata; and general automata, including strongly-driven linear indexed automata and strongly-driven 2–stack automata.

Índice General

Índice de Figuras

Índice de Tablas

Capítulo 1

Introducción

La estructura de un lenguaje se describe mediante su sintaxis, de tal modo que una frase bien formada puede dividirse en constituyentes de acuerdo a unas determinadas reglas sintácticas. Cada uno de los constituyentes puede a su vez dividirse en constituyentes más pequeños de acuerdo con las mismas reglas, hasta que finalmente se obtiene una descripción jerárquica completa de la estructura de la frase. Las reglas sintácticas pueden describirse utilizando diversos formalismos. En esta memoria se considera un subconjunto de tales formalismos que recibe la denominación genérica de *formalismos gramaticales suavemente dependientes del contexto*. Dentro de estos, nos centraremos en dos, las *gramáticas de adjunción de árboles* y las *gramáticas lineales de índices*. El conjunto de lenguajes cuya estructura puede ser descrita por estos dos formalismos recibe el nombre de *lenguajes de adjunción de árboles*.

Un programa de ordenador que se encarga de asignar a una frase la estructura jerárquica que le corresponde de acuerdo con su sintaxis recibe el nombre de *analizador sintáctico*. Un analizador sintáctico puede trabajar directamente a partir de la gramática que define la estructura del lenguaje o bien puede trabajar sobre una máquina abstracta o *autómata*. Un autómata no es más que un dispositivo matemático que reescribe el contenido almacenado en una estructura, que habitualmente es una pila, de acuerdo con unas reglas de reescritura que reciben el nombre de *transiciones*. En el caso de los analizadores sintácticos que trabajan sobre autómatas es preciso realizar un paso previo en el cual las reglas de la gramática se transforman en un conjunto equivalente de transiciones mediante un *esquema de compilación*. Esta memoria trata de la ejecución eficiente de los autómatas con el fin de obtener analizadores sintácticos eficientes para la clase de los lenguajes de adjunción de árboles.

1.1 El lenguaje natural

Tanto las lenguas que hablan las personas como las que *hablan* los ordenadores tienen estructura. Las primeras han sido creadas de forma natural a lo largo de siglos de evolución y por ello reciben el nombre de *lenguas naturales*[1]. En cambio, todo lenguaje de programación ha sido creado en un momento dado con un propósito concreto y le ha sido impuesta una estructura fija, por ello son lenguajes artificiales. En cuanto a la sintaxis, existe una diferencia importante entre ambos, pues mientras en los lenguajes de programación se diseña una gramática a partir de la cual se crean los programas, en las lenguas naturales la gente *diseña* un idioma mediante la creación de frases sin que exista una gramática explícita. Es precisamente una de las tareas más complicadas de la lingüística el hacer explícitas las reglas sintácticas de una lengua. Estas diferencias de origen

[1] A menudo utilizaremos el término *lenguaje natural* para referiremos en su globalidad a las lenguas naturales, sin especificar ninguna en concreto.

han permitido que los lenguajes de programación rehuyan las construcciones ambiguas, aquellas en las que es posible asociar más de una interpretación a una frase. En cambio, la ambigüedad es intrínseca en las lenguas naturales, tanto a nivel morfológico como sintáctico y semántico. En el caso de la sintaxis, el hecho de que una frase sea ambigua se traduce en que es posible asociar dos o más estructuras sintagmáticas correctas a dicha frase. Como ejemplo, tomaremos una frase conocida: "Juan vio un hombre con un telescopio en una colina". Diferentes ubicaciones de las subestructuras correspondientes a los fragmentos "con un telescopio" y "en una colina" llevan a diferentes estructuras sintagmáticas completas para la frase, todas ellas correctas, que se corresponden con los significados siguientes:

- Juan vio un hombre que estaba en una colina y que tenía un telescopio.
- Juan estaba en una colina, desde donde miraba con un telescopio, a través del cual vio un hombre.
- Juan estaba en una colina, desde donde vio un hombre que tenía un telescopio.
- Juan miraba por un telescopio, a través del cual vio un hombre que estaba en una colina.

Esta característica hace que los analizadores sintácticos diseñados para tratar el lenguaje natural sean más complejos que los algoritmos dedicados al análisis de los lenguajes de programación, pues deben ser capaces de manejar la ambigüedad, determinando todas las posibles estructuras sintácticas asociadas a una frase y almacenándolas en forma compacta y compartible al objeto de evitar la realización de cálculos duplicados. Para este propósito se recurre habitualmente a técnicas de tabulación.

1.2 La programación dinámica en el análisis sintáctico

Bellman [29] introduce en 1957 la programación dinámica para responder a los problemas de optimización[2]. Este tipo de problemas se puede descomponer en subproblemas más simples de tal modo que la solución final se obtiene combinando las soluciones obtenidas en los subproblemas. Por tanto, la programación dinámica responde esencialmente a la idea de descomponer un problema en subproblemas que se resuelven una única vez, cuyos resultados se almacenan en una tabla y, lo más importante, se reutilizan varias veces sin necesidad de volver a calcularlos. De hecho, el término *dinámica* se refiere a la noción de *tabulación*, el almacenamiento de la información en una tabla dinámica, lo cual permite cambiar dinámicamente la manera de calcular un subproblema mediante la reutilización de las soluciones presentes en la tabla para este subproblema.

Sin embargo, la programación dinámica no es la panacea. Existen algoritmos que recurren al principio de *divide y vencerás*[3] en los que, a pesar de aplicar la descomposición del problema en subproblemas, el uso de la tabulación hace aumentar el coste de los cálculos sin que se reduzca su complejidad[4]. Ello se debe a que los resultados de los subproblemas no se calculan más de una vez, por lo que no se evita su repetición [183].

No se conoce ninguna condición general necesaria y suficiente que se deba satisfacer para garantizar la bondad de la aplicación de la programación dinámica a un problema dado. No obstante, la experiencia ha mostrado que esta técnica es útil en problemas que presentan las siguientes características:

[2] Un ejemplo típico consiste en buscar la distancia mínima entre dos puntos de un grafo.
[3] Dividir un problema complejo en varios más simples que son útiles para la solución del problema original.
[4] Un problema con estas características lo constituye la búsqueda dicotómica.

1. Existe una descomposición del problema inicial en subproblemas más simples. Esta descomposición generalmente se expresa mediante una definición recursiva uniforme del problema.

2. Gran parte de los subproblemas son idénticos, permitiendo la reutilización de gran parte de las soluciones calculadas previamente.

3. El coste de resolver los subproblemas es mayor que el coste de almacenamiento y búsqueda de la solución de un problema en la tabla.

Ejemplos típicos de problemas para los cuales la programación dinámica resulta beneficiosa son el cálculo de la función de Fibonacci, el recorrido del camino más corto de un grafo ponderado, y la resolución ascendente utilizada en la programación lógica [56, 195, 203].

El problema de fondo de todos los algoritmos de programación dinámica es la gestión de la tabla de objetos. La eficacia en cuanto a tiempos de ejecución depende de los tiempos de acceso a la tabla, de ahí el interés en los métodos de indexación de la misma. Esta es especialmente eficaz cuando conocemos *a priori* el tamaño del problema a tratar. Por desgracia, este no es el caso del problema que nos ocupa. El número de objetos que necesitamos calcular durante el análisis de una frase no se puede conocer de antemano, pues depende tanto de la frase como de la gramática que define la estructura del lenguaje. Ello nos obliga a manejar la tabla de objetos de manera dinámica.

En el contexto del análisis sintáctico, los algoritmos que hacen uso de la programación dinámica se denominan *algoritmos tabulares* y se caracterizan por almacenar los resultados intermedios del análisis en una tabla cuyos componentes reciben el nombre de *ítems*. El algoritmo de Earley (véase el apéndice B) es el ejemplo clásico de algoritmo tabular de análisis sintáctico. A partir de este algoritmo se han desarrollado otros para multitud de formalismos gramaticales [189], entre los que figuran las gramáticas de adjunción de árboles [8, 9] y las gramáticas lineales de índices [15, 19]. Frecuentemente, el algoritmo de Earley es también considerado como la base para la interpretación tabular de los autómatas a pila [104] y sus extensiones [105, 52].

1.3 Formalismos gramaticales

La potencia expresiva de las gramáticas independientes del contexto es habitualmente suficiente para describir la sintaxis de los lenguajes de programación. Sin embargo, las lenguas naturales presentan construcciones que no pueden ser descritas mediante gramáticas independientes del contexto. Surge entonces la necesidad de encontrar otro formalismo gramatical más adecuado. Un obstáculo importante en esta búsqueda es que no se sabe a ciencia cierta qué lugar ocuparían las lenguas naturales en la jerarquía de lenguajes definida por Chomsky, aunque se cree que estarían situados entre los lenguajes independientes del contexto y los lenguajes dependientes del contexto, posiblemente más cerca de los primeros que de los segundos. Esta suposición se basa en el hecho de que la mayoría de las construcciones sintácticas sólo dependen *suavemente* del contexto en el cual son aplicadas.

Puesto que la estructura sintáctica asociada a las frases es una estructura jerárquica representada normalmente como un árbol o, en el caso de frases ambiguas, como un conjunto de árboles, parece natural pensar que un formalismo que manipule árboles y que presente cierta dependencia suave del contexto debe facilitar la descripción de la sintaxis de las lenguas naturales. En esta dirección las gramáticas de adjunción de árboles [94], un formalismo suavemente dependiente del contexto que manipula árboles, se ha mostrado adecuado para la descripción de los fenómenos sintácticos que aparecen en el lenguaje natural [90].

El conjunto de los lenguajes generados por las gramáticas de adjunción de árboles constituye la clase de los lenguajes de adjunción de árboles. Esta clase también puede ser generada por otros formalismos gramaticales, por ejemplo las gramáticas lineales de índices [75]. Es este un formalismo cuyas construcciones son más próximas a las gramáticas independientes del contexto, por lo que presenta ciertas ventajas a la hora de realizar el tratamiento computacional, puesto que los algoritmos de análisis sintáctico y los modelos de autómatas diseñados para las primeras pueden ser más fácilmente extendidas a las segundas que a otros formalismos.

1.4 Ámbito de la tesis

El trabajo presentado en esta memoria se enmarca en lo que se conoce comúnmente como *procesamiento del lenguaje natural*, el punto donde se cruzan disciplinas aparentemente tan dispares como la informática, la lingüística, la inteligencia artificial y la psicología.

En lo que respecta a la informática, el trabajo realizado supone aportaciones significativas a la teoría de autómatas y lenguajes formales y al ámbito de los compiladores y procesadores del lenguaje. Las aportaciones a la teoría de autómatas y lenguajes formales se centran en los siguientes aspectos:

- La definición de nuevos algoritmos de análisis sintáctico para los lenguajes de adjunción de árboles, una familia abstracta de lenguajes bien localizada en la jerarquía de Chomsky.
- La definición de diferentes modelos de autómata para dicha clase de lenguajes.
- El estudio de la complejidad temporal y espacial de los algoritmos de análisis sintáctico y de la ejecución de los autómatas.

En relación a los compiladores y procesadores del lenguaje:

- Se proponen esquemas de compilación automática de gramáticas de adjunción de árboles y de gramáticas lineales de índices en diversos modelos de autómata.
- Se definen técnicas de interpretación tabular para diversos modelos de autómata que permiten la ejecución eficiente de las estrategias de análisis.

Con respecto a la lingüística, las aportaciones realizadas afectan al área de la *lingüística computacional*, una disciplina surgida de la cooperación de la informática y la lingüística. En su sentido más general, la lingüística computacional engloba todas las aplicaciones informáticas que tratan del lenguaje natural mediante la aplicación de conocimientos lingüísticos referidos a la morfología, la sintaxis, la semántica o la pragmática. Las aportaciones más relevantes realizadas a este área son:

- El estudio de métodos de análisis sintáctico para lenguajes de adjunción de árboles, una clase de lenguajes suavemente dependientes del contexto que ha suscitado un gran interés en los últimos años. Actualmente existen gramáticas de adjunción de árboles de gran cobertura para el inglés [67, 198] y el francés [2, 1] y gramáticas parciales para el alemán [154], el coreano [79, 233], el español [22, 103, 41], el italiano [38], el portugués [3], el rumano [110] y otros idiomas.
- La definición de técnicas eficientes para el análisis no determinista de lenguajes de adjunción de árboles, aspecto muy importante en el tratamiento de las lenguas naturales puesto que estas suelen presentar un elevado grado de ambigüedad en ciertas construcciones sintácticas.

La obtención de modelos computacionales que permitan comprender el lenguaje utilizado por los seres humanos para comunicarse, siempre ha supuesto un objetivo importante para la inteligencia artificial. Uno de los problemas a resolver para alcanzar ese objetivo es el de obtener eficientemente la estructura sintagmática correspondiente a las frases. El trabajo que se expone en esta memoria supone un avance en esta dirección.

1.5 Estructura de la memoria

Esta memoria se estructura en tres partes. En la primera se presentan los lenguajes de adjunción de árboles y las técnicas de análisis sintáctico para dicha clase de lenguajes. En la segunda, que constituye el núcleo de la memoria, se presentan diversos modelos de autómata para esta clase de lenguajes junto con las técnicas que permiten realizar la interpretación tabular de cada uno de ellos. La tercera parte la constituyen una serie de apéndices en los que se presenta material que, si bien no es imprescindible, es de interés en el ámbito de la tesis. A continuación presentamos un breve resumen del contenido de cada uno de los capítulos.

Parte I. Lenguajes de adjunción de árboles

Capítulo 2. En este capítulo se realiza una presentación de los lenguajes de adjunción de árboles, situándolos en la jerarquía de Chomsky. Se tratan en detalle dos formalismos gramaticales que generan esta clase de lenguajes, las gramáticas de adjunción de árboles y las gramáticas lineales de índices, pues son los formalismos sobre los que se trabajará en el resto de la memoria. También se presentan brevemente otros formalismos que generan la misma clase de lenguajes.

Capítulo 3. Este capítulo constituye un estudio sobre el estado actual del análisis sintáctico de las gramáticas de adjunción de árboles, aunque incluye aportaciones novedosas. En particular, se presenta una línea evolutiva continua en la cual se sitúan los algoritmos tabulares correspondientes a las principales estrategias de análisis para gramáticas de adjunción de árboles, abarcando desde estrategias puramente ascendente hasta estrategias de tipo Earley que preservan la propiedad del prefijo válido. Todos estos algoritmos se definen mediante esquemas de análisis sintáctico, de tal modo que los algoritmos más complejos se derivan a partir de los menos complejos aplicando una secuencia de transformaciones simples. También se presentan aquellos algoritmos que incorporan estrategias bidireccionales, que realizan el proceso de análisis en varias fases, que basan el análisis en una compilación en gramáticas lineales de índices, que precompilan parte de la información en forma de un autómata de tipo LR y aquellos diseñados específicamente para su ejecución en máquinas paralelas.

Capítulo 4. En este capítulo se realiza un estudio sobre el estado actual del análisis sintáctico de las gramáticas lineales de índices, al que se ha contribuido con el desarrollo de algoritmos tabulares de tipo Earley con y sin la propiedad del prefijo válido. El diseño de estos algoritmos nos ha permitido crear una línea evolutiva continua paralela a la desarrollada en el capítulo precedente para el caso de las gramáticas de adjunción de árboles.

Parte II. Modelos de autómata para los lenguajes de adjunción de árboles

Capítulo 5. Antes de proceder a la definición de nuevos modelos de autómata, se presenta en este capítulo un repaso de los autómatas a pila y de las técnicas de tabulación disponibles para los mismos.

Capítulo 6. En este capítulo se presentan los autómatas a pila embebidos, en los cuales la estructura principal de almacenamiento la constituye una pila de pilas. Junto a la definición clásica se presenta una nueva formulación en la cual se elimina el control de estado finito y se simplifica la forma de las transiciones al tiempo que se mantiene la potencia expresiva. Esta nueva formulación permite diseñar una técnica de tabulación para la ejecución eficiente de los diversos esquemas de compilación para gramáticas de adjunción de árboles y gramáticas lineales de índices que se definen en el capítulo.

Capítulo 6. La versión dual del modelo de autómata tratado en el capítulo anterior la constituyen los autómatas a pila embebidos ascendentes. En este capítulo se realiza una definición formal de los mismos, algo que no se había logrado hasta el momento. La eliminación del control de estado finito permite simplificar la forma de las transiciones, lo cual facilita la definición de una técnica de tabulación para este modelo de autómata.

Capítulo 8. En este capítulo mostramos cómo las gramáticas lineales de índices constituyen un tipo específico de gramáticas de cláusulas definidas en el cual los predicados tienen un único argumento en forma de pila de índices. Aprovechamos esta característica para definir una versión restringida de los autómatas lógicos a pila adecuada al tratamiento de este tipo de gramáticas y de las gramáticas de adjunción de árboles. Dependiendo de la forma de las transiciones permitidas, podemos distinguir tres tipos diferentes de autómata, uno que permite el análisis ascendente de los índices o adjunciones, otro que permite el análisis descendente y otro que permite estrategias mixtas. En los dos últimos casos es preciso establecer restricciones en la combinación de las transiciones para garantizar que dichos autómatas aceptan exactamente la clase de los lenguajes de adjunción de árboles. Se presentan esquemas de compilación y técnicas de tabulación para los tres tipos de autómata.

Capítulo 9. En este capítulo se presentan los autómatas lineales de índices, que utilizan la misma estructura de almacenamiento que los autómatas lógicos a pila restringidos pero con un juego diferente de transiciones. Distinguimos tres tipos diferentes de autómata: los autómatas lineales de índices orientados a la derecha para estrategias en las cuales las pilas de índices se evalúan de modo ascendente, los autómatas lineales de índices orientados a la izquierda en los cuales las pilas de se evalúan de modo descendente y los autómatas lineales de índices fuertemente dirigidos que permiten definir estrategias mixtas de análisis para el tratamiento de las pilas de índices. Es precisamente la definición de este último tipo de autómatas y de la correspondiente técnica de tabulación la principal aportación de este capítulo.

Capítulo 10. En este capítulo se opta por un modelo de autómata con una nueva estructura de almacenamiento. Se preserva la pila de los autómatas a pila tradicionales, a la que acompaña una pila auxiliar cuyo contenido restringe el conjunto de transiciones aplicables es un momento dado. Los autómatas con dos pilas fuertemente dirigidos permiten definir esquemas de compilación arbitrarios para gramáticas de adjunción de árboles y gramáticas lineales de índices. Por su parte, los autómatas con dos pilas ascendentes sólo permiten describir esquemas de compilación que incorporan estrategias ascendentes en lo referente al tratamiento de las adjunciones y de las pilas de índices. Se presentan las técnicas de tabulación que permiten una ejecución eficiente de ambos modelos de autómata.

Capítulo 11. Una vez definidos los diferentes modelos de autómata, llega el momento de analizarlos conjuntamente, percibiéndose la existencia de tres grandes grupos de autómatas: los autómatas generales, entre los que se incluyen los autómatas lineales de índices fuertemente dirigidos y los autómatas con dos pilas fuertemente dirigidos; los autómatas descendentes, entre los que se encuadran los autómatas a pila embebidos y los autómatas lineales de índices orientados a la izquierda; y los autómatas ascendentes, que incluyen los autómatas a pila embebidos ascendentes, los autómatas lineales de índices orientados a la derecha y los autómatas con dos pilas ascendentes.

Parte III. Apéndices

Capítulo A. En este apéndice se presenta un resumen de los esquemas de análisis sintáctico, la estructura formal en la cual se describen los algoritmos de análisis sintáctico para los diferentes formalismos gramaticales utilizados en esta memoria.

Capítulo B. Este apéndice contiene la definición de los algoritmos de análisis sintáctico CYK y Earley para gramáticas independientes del contexto, que constituyen la base de la mayor parte de los algoritmos de análisis sintáctico para gramáticas de adjunción de árboles y para gramáticas lineales de índices.

Capítulo C. A partir del algoritmo de Earley se derivan las técnicas de interpretación tabular de los diferentes tipos de algoritmos LR para gramáticas independientes del contexto. A continuación se presenta un algoritmo de tipo LR para extensiones basadas en unificación de las gramáticas independientes del contexto, finalizando con la presentación de un algoritmo LR para gramáticas lineales de índices.

Para facilitar la lectura, los párrafos que rompen la continuidad del texto están claramente señalados: los lemas, propiedades y teoremas están en cursiva; las demostraciones presentan unos márgenes laterales mayores, un tipo de letra ligeramente más pequeño y terminan con la marca □; los esquemas de análisis y de compilación finalizan con la marca § mientras que los ejemplos finalizan con ¶.

1.6 Difusión de resultados

El material generado durante la realización de la presente tesis doctoral ha dado lugar a varios artículos de revista, capítulos de libro y ponencias en congresos. A continuación detallamos los trabajos surgidos de los diferentes capítulos.

Capítulo 3 Los resultados fundamentales de este capítulo han sido publicados en inglés y en español en los siguientes trabajos:

- Miguel A. Alonso Pardo, David Cabrero Souto, Eric de la Clergerie, y Manuel Vilares Ferro. Tabular algorithms for TAG parsing. In *Proc. of EACL'99, Ninth Conference of the European Chapter of the Association for Computational Linguistics*, páginas 150–157, Bergen, Noruega, junio de 1999. ACL.
- Miguel A. Alonso Pardo, David Cabrero Souto, Eric de la Clergerie, y Manuel Vilares Ferro. Algoritmos tabulares para el análisis de TAG. *Procesamiento del Languaje Natural*, 23:157–164, septiembre de 1998.

Capítulo 4 Los resultados de este capítulo aparecen en inglés y en español en:

- Miguel A. Alonso Pardo, Eric de la Clergerie, Jorge Graña Gil, y Manuel Vilares Ferro. New tabular algorithms for LIG parsing. In *Proc. of the Sixth International Workshop on Parsing Technologies (IWPT 2000)*, páginas 29–40, Trento, Italia, febrero de 2000. ACL/SIGPARSE.
- Miguel A. Alonso Pardo, Jorge Graña Gil, y Manuel Vilares Ferro. Nuevos algoritmos tabulares para el análisis de LIG. *Procesamiento del Lenguaje Natural*, 25:7–14, septiembre de 1999.

Capítulo 6 El material de este capítulo ha sido publicado en:

- Miguel A. Alonso Pardo, Eric de la Clergerie, y Manuel Vilares Ferro. A redefinition of Embedded Push-Down Automata. In *Proc. of the 5th International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+5)*, páginas 19–26, París, Francia, mayo de 2000.

Capítulo 7 Una parte significativa de este capítulo ha sido publicada en:

- Miguel A. Alonso Pardo, Eric de la Clergerie, and Manuel Vilares Ferro. A formal definition of Bottom-up Embedded Push-Down Automata and their tabulation technique. In *Proc. of Second International Workshop on Tabulation in Parsing and Deduction (TAPD 2000)*, Vigo, España, septiembre de 2000.

Capítulo 8 Los resultados más relevantes de este capítulo han sido publicados en:

- Miguel A. Alonso Pardo, Eric de la Clergerie, y David Cabrero Souto. Tabulation of automata for tree adjoining languages. In *Proc. of the Sixth Meeting on Mathematics of Language (MOL 6)*, páginas 127–141, Orlando, Florida, USA, julio de 1999.

Capítulo 9 Las ideas presentadas en este capítulo han dado lugar al siguiente artículo:

- Miguel A. Alonso Pardo, Mark-Jan Nederhof, y Eric de la Clergerie. Tabulation of automata for tree adjoining languages. *Grammars*, a aparecer.

Capítulo 10 El material de la sección 10.3 ha sido publicado en:

- Eric de la Clergerie y Miguel A. Alonso Pardo. A tabular interpretation of a class of 2-Stack Automata. In *COLING-ACL'98, 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Proceedings of the Conference*, volumen II, páginas 1333–1339, Montreal, Quebec, Canadá, agosto de 1998. ACL.
- Miguel A. Alonso Pardo, Djamé Seddah, y Eric de la Clergerie. Practical aspects in compiling tabular TAG parsers. In *Proc. of 5th International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+5)*, páginas 27–32, París, Francia, mayo de 2000.

Los resultados de la sección 10.4 han sido publicados en:

- Eric de la Clergerie, Miguel A. Alonso Pardo, y David Cabrero Souto. A tabular interpretation of bottom-up automata for TAG. In *Proc. of Fourth International Workshop on Tree-Adjoining Grammars and Related Frameworks (TAG+4)*, páginas 42–45, Filadelfia, PA, USA, agosto de 1998.

Apéndice C Una parte importante del material de este apéndice ha aparecido en el siguiente capítulo de libro:

- Miguel A. Alonso Pardo, David Cabrero Souto, y Manuel Vilares Ferro. Construction of efficient generalized LR parsers. In Derick Wood y Sheng Yu, editores, *Automata Implementation*, volumen 1436 de *Lecture Notes in Computer Science*, páginas 7–24. Springer-Verlag, Berlín-Heidelberg-Nueva York, 1998.

cuya versión previa apareció en:

- Miguel A. Alonso Pardo, David Cabrero Souto, y Manuel Vilares Ferro. Construction of efficient generalized LR parsers. In *Proc. of Second International Workshop on Implementing Automata (WIA'97)*, páginas 131–140, London, Ontario, Canadá, septiembre de 1997.

El material de la sección C.7 ha sido publicado en el sigiente capítulo de libro:

- Miguel A. Alonso Pardo, David Cabrero Souto, y Manuel Vilares Ferro. Generalized LR parsing for extensions of context-free grammars. In Nicolas Nicolov y Ruslan Mitkov, editores, *Recent Advances in Natural Language Processing II*, volumen 189 de *Current Issues in Linguistic Theory*. John Benjamins Publishing Company, Amsterdam & Filadelfia, 1999.

cuya versión previa apareció en:

- Miguel A. Alonso Pardo, David Cabrero Souto, y Manuel Vilares Ferro. A new approach to the construction of Generalized LR parsing algorithms. In Ruslan Mitkov, Nicolas Nicolov, y Nikolai Nikolov, editores, *Proc. of Recent Advances in Natural Language Processing (RANLP'97)*, páginas 171–178, Tzigov Chark, Bulgaria, septiembre de 1997.

El material de la sección C.8 ha sido publicado en:

- Miguel A. Alonso Pardo, Eric de la Clergerie, y Manuel Vilares Ferro. Automata-based parsing in dynamic programming for Linear Indexed Grammars. In A. S. Narin'yani, editor, *Proc. of DIALOGUE'97 Computational Linguistics and its Applications International Workshop*, páginas 22–27, Moscú, Rusia, junio de 1997.

1.7 Comunicación con el autor

Los comentarios y sugerencias acerca de esta memoria y del trabajo en ella reflejado son bienvenidos. Se puede contactar con el autor en la dirección

Miguel A. Alonso Pardo
Departamento de Computación
Facultad de Informática
Campus de Elviña s/n
15071 La Coruña (España)

o bien mediante correo electrónico en la dirección `alonso@dc.fi.udc.es`.

En las páginas web del autor está disponible información adicional referente a esta tesis y a trabajos relacionados. La dirección es `http://www.dc.fi.udc.es/~alonso/`

Parte I

Lenguajes de adjunción de árboles

Capítulo 2

Lenguajes de adjunción de árboles

La clase de los lenguajes suavemente dependientes del contexto (*Mildly Context-Sensitive Languages*, MCSL) se halla entre la clase de los lenguajes independientes del contexto y la de los lenguajes dependientes del contexto. Los lenguajes de adjunción de árboles (*Tree Adjoining Languages*, TAL) constituyen una de las subclases más importantes dentro de los MCSL. Aunque los TAL sólo son ligeramente más expresivos que los lenguajes independientes al contexto, presentan un gran interés tanto para la teoría de lenguajes formales como para la lingüística computacional puesto que constituyen una familia abstracta de lenguajes y permiten modelar ciertos fenómenos sintácticos propios de las lenguas naturales. A lo largo de los últimos años se han desarrollado varios formalismos gramaticales que pueden ser utilizados para describir los lenguajes de adjunción de árboles. De entre ellos, nos centraremos en las gramáticas de adjunción de árboles y en las gramáticas lineales de índices.

2.1 Lenguajes suavemente dependientes del contexto

Una de las tareas de la teoría de lenguajes formales consiste en identificar clases de lenguajes con propiedades relevantes. Puesto que la forma más habitual de describir un lenguaje es a través de una gramática que genere todas las cadenas o palabras de dicho lenguaje, podemos considerar que la descripción de una clase de lenguajes es equivalente a la descripción de una clase de gramáticas que lo genera. Siguiendo este pensamiento, Chomksy diseñó a finales de los años 60 una jerarquía de sistemas gramaticales. La jerarquía de Chomsky[49] es comúnmente aceptada y sirve de referente a las nuevas clases de lenguajes y gramáticas[1]. Se han dedicado importantes esfuerzos dentro de la comunidad de la lingüística computacional para ubicar en esta jerarquía el lugar correspondiente al lenguaje natural. Dicha clase estaría situada en algún punto entre los lenguajes independientes del contexto y los lenguajes dependientes del contexto, aunque posiblemente más cerca de los primeros que de los últimos.

Los lenguajes independientes del contexto no son suficientemente potentes para describir las lenguas naturales, puesto que existen ciertas construcciones básicas que no pueden ser descritas [50], tales como:

- *Replicación*, que produce lenguajes de la forma $\{ww\}$. Este tipo de construcciones se dan en ciertas variantes del alemán [75].
- *Concordancias cruzadas*, modeladas por lenguajes de la forma $\{a^n b^m c^n d^m \mid n, m \geq 1\}$. Este tipo de construcciones se dan por ejemplo en el holandés [229, 90].

[1]Sin embargo existen sistemas gramaticales que no se adaptan a la jerarquía de Chomsky. Por ejemplo, los lenguajes generados por las Gramáticas Contextuales [116] son incomparables con los lenguajes independientes del contexto aunque están propiamente incluidos en los lenguajes dependientes del contexto.

Las gramáticas y/o la clase de lenguajes que se pretenda capaz de describir los lenguajes naturales deben satisfacer una serie de propiedades formales [90, 95], como son:

1. *Inclusión de los lenguajes independientes del contexto.*
2. *Análisis sintáctico en tiempo polinomial.*
3. *Captura de ciertas clases de dependencias*, tales como las dependencias anidadas y ciertas clases de dependencias cruzadas.
4. *Propiedad del crecimiento constante.*

Las tres primeras propiedades tienen un significado claro. La última propiedad hace referencia al hecho de que si las cadenas de un lenguaje se disponen en orden de longitud creciente, la longitud de dos cadenas situadas en posiciones consecutivas no pueden diferir en cantidades arbitrariamente grandes. De hecho, la longitud de cualquier cadena deber poder obtenerse como una combinación lineal de un conjunto finito de longitudes fijas. La propiedad del crecimiento constante es ligeramente más débil que la propiedad de semilinealidad[2] y hace referencia a la intuición lingüística de que la frases de un lenguaje natural se pueden construir a partir de un conjunto finito de construcciones de tamaño acotado mediante la utilización de operaciones lineales.

Los lenguajes que satisfacen estas cuatro propiedades reciben el nombre de *lenguajes suavemente dependientes del contexto* (*Mildly Context-Sensitive Languages*, MCSL) y los formalismos gramaticales que los generan reciben en consecuencia la denominación de *gramáticas suavemente dependientes del contexto* (*Mildly Context-Sensitive Grammars*, MCSG). Las propiedades enumeradas no proporcionan una caracterización precisa de los MCSL ni de las MCSG sino que proporcionan una descripción intuitiva. Podríamos decir que dichas propiedades son condiciones necesarias.

El estudio de los lenguajes y gramáticas suavemente dependientes del contexto y de sus propiedades aumentó en interés al irse descubriendo la equivalencia de varios formalismos gramaticales suavemente dependientes del contexto que habían nacido a partir de ideas y principios totalmente distintos [216, 138]. Concretamente, nos estamos refiriendo a las gramáticas de adjunción de árboles [94], las gramáticas lineales de índices [75], las gramáticas de núcleo [146] y las gramáticas categoriales combinatorias [194]. Vijay-Shanker y Weir muestran en [216] la inclusión de los lenguajes categoriales combinatorios en los lenguajes lineales de índices, la de estos en los lenguajes de núcleo, la de estos en los lenguajes de adjunción de árboles y la de estos últimos en los lenguajes categoriales combinatorios. Aplicando la transitividad de la relación de equivalencia vemos que los cuatro formalismos gramaticales generan la misma clase de lenguajes. A este tipo de equivalencia se la denomina *equivalencia débil* por oposición a la *equivalencia fuerte*. Dos formalismos gramaticales son débilmente equivalentes si generan las mismas cadenas o palabras (la misma clase de lenguaje) y son fuertemente equivalente si además de serlo débilmente imponen la misma estructura a las cadenas generadas por ambos formalismos [92].

Ejemplo 2.1 Para ilustrar los conceptos de equivalencia mostraremos un ejemplo que trata de la equivalencia débil y la equivalencia fuerte aplicada a gramáticas pertenecientes a un mismo formalismo.

Sean $\mathcal{G}_1 = (V_N, V_T, P_1, S)$ y $\mathcal{G}_2 = (V_N, V_T, P_2, S)$ dos gramáticas independientes del contexto, donde $V_N = \{S\}$ es el conjunto de símbolos no-terminales, $V_T = \{a\}$ es el conjunto de símbolos

[2]Un lenguaje tiene la propiedad de la semilinealidad si el número de ocurrencias de cada símbolo en cualquier cadena es una combinación lineal de las ocurrencias de esos símbolos en algún conjunto finito de cadenas [230].

terminales, $S \in V_N$ es el axioma de ambas gramáticas y donde el conjunto de producciones P_1 contiene

$$S \to aS$$
$$S \to \epsilon$$

mientras que P_2 contiene las producciones

$$S \to aSa$$
$$S \to a$$
$$S \to \epsilon$$

Las gramáticas $\mathcal{G}_2$ y $\mathcal{G}_2$ son débilmente equivalentes puesto que los lenguajes generados, denotados respectivamente como $L(\mathcal{G}_1)$ y $L(\mathcal{G}_2)$, son iguales a $\{a^*\}$. Sin embargo, $\mathcal{G}_1$ y $\mathcal{G}_2$ no son fuertemente equivalente puesto que $\mathcal{G}_1$ asigna a la cadena aaa la estructura que se muestra en la parte izquierda de la figura 2.1 mientras que $\mathcal{G}_2$ asigna a dicha cadena la estrutura mostrada en la parte derecha de la figura 2.1. ¶

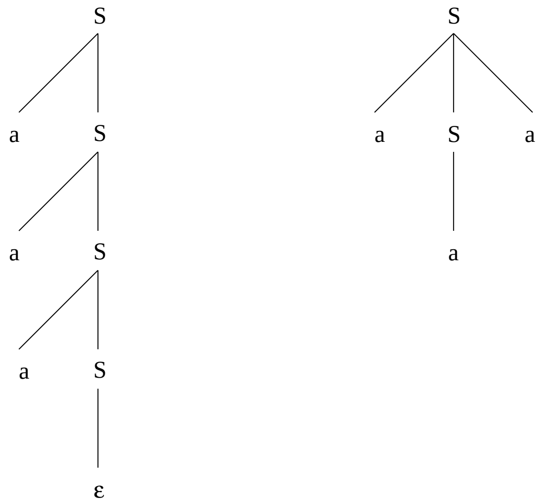

Figura 2.1: Estruturas asociadas a la cadena aaa por $\mathcal{G}_1$ y por $\mathcal{G}_2$

De todos estos formalismos, el que más interés has despertado son las gramáticas de adjunción de árboles por su adecuación para la descripción de fenómenos lingüísticos. Tal vez por ello la clase de lenguajes generada por todos estos formalismos es conocida habitualmente por la clase de los lenguajes de adjunción de árboles (TAL). Por su parte, las gramáticas lineales de índices han sido ampliamente estudiadas debido a su mejor adaptación al tratamiento computacional. Puesto que TAG y LIG son equivalentes, se puede aprovechar la adecuación lingüística de TAG para describir la gramática de una lengua natural y posteriormente traducir dicha gramática a una LIG equivalente con el fin de aplicar sobre ella el proceso de análisis sintáctico. Es esta la razón por la cual en este trabajo nos hemos centrado en el estudio de estos dos formalismos.

2.2 Gramáticas de adjunción de árboles

Las gramáticas de adjunción de árboles (*Tree Adjoining Grammars*, TAG) son una extensión de las gramáticas independientes del contexto y fueron definidas inicialmente por Joshi, Levy y Takahashi en [93]. Joshi refina ciertos aspectos en [91], estableciendo la definición moderna de TAG. En [94] puede encontrarse un estudio reciente de Joshi y Schabes acerca de las características de este formalismo gramatical.

Formalmente, una gramática de adjunción de árboles es una quíntupla $(V_T, V_N, \boldsymbol{I}, \boldsymbol{A}, S)$ donde:

- V_T es un conjunto finito de símbolos *terminales*.
- V_N es un conjunto finito de símbolos *no-terminales*. Se cumple que $V_T \cap V_N = \emptyset$.
- $\boldsymbol{I}$ es un conjunto finito de *árboles iniciales*.
- $\boldsymbol{A}$ es un conjunto finito de *árboles auxiliares*.
- $S \in V_N$ es el axioma de la gramática.

Los árboles en $\boldsymbol{I} \cup \boldsymbol{A}$ se denominan *árboles elementales* de la gramática. Los árboles iniciales se caracterizan porque su raíz está etiquetada por el axioma de la gramática, sus nodos interiores están etiquetados por no-terminales y sus nodos hoja están etiquetados por terminales o por la palabra vacía, denotada por ϵ. Los árboles auxiliares son como los árboles iniciales con la excepción de que la etiqueta de su raíz puede ser un no-terminal arbitrario y porque uno de sus nodos hoja, que recibe el nombre de *pie* está etiquetado por el mismo no-terminal que etiqueta su raíz. El camino desde el nodo raíz hasta el nodo pie recibe el nombre de *espina*.

2.2.1 La operación de adjunción

Los árboles elementales se pueden combinar entre sí para crear *árboles derivados*, los cuales a su vez se pueden combinar con otros árboles para formar árboles derivados más grandes. La operación mediante la cual se combinan los árboles se denomina *adjunción* y se muestra gráficamente en la figura 2.2. Mediante una adjunción se construye un nuevo árbol a partir de un árbol auxiliar β y de otro árbol γ, que puede ser un árbol inicial, auxiliar o derivado de adjunciones realizadas previamente. En su forma más simple, una adjunción puede tener lugar si la etiqueta de un nodo del árbol γ (denominado *nodo de adjunción*) coincide con la etiqueta del nodo raíz de un árbol auxiliar β. En tal caso, el árbol derivado resultante se construye como sigue:

1. El subárbol de γ dominado por el nodo de adjunción se escinde de γ, aunque se deja una copia del nodo de adjunción en γ.
2. El árbol auxiliar β se pega a la copia del nodo de adjunción de tal forma que la raíz del árbol auxiliar se identifica con dicha copia.
3. El subárbol escindido de γ se pega al nodo pie del árbol auxiliar β de tal modo que la raíz del subárbol escindido (el nodo de adjunción) se identifica con el nodo pie de β.

La aplicabilidad de una adjunción tal y como ha sido descrita sólo depende de las etiquetas de los nodos. Sin embargo, por conveniencia se puede especificar para cada nodo un conjunto de restricciones que permite indicar con más precisión los árboles auxiliares que pueden ser adjuntados. Las restricciones asociadas a un nodo, que se denominan *restricciones de adjunción*, pueden ser de los tipos siguientes:

- Restricciones de *adjunción selectiva* (SA) que especifican el subconjunto de árboles auxiliares que pueden participar en una operación de adjunción. En todo caso, no es obligatorio realizar una adjunción.

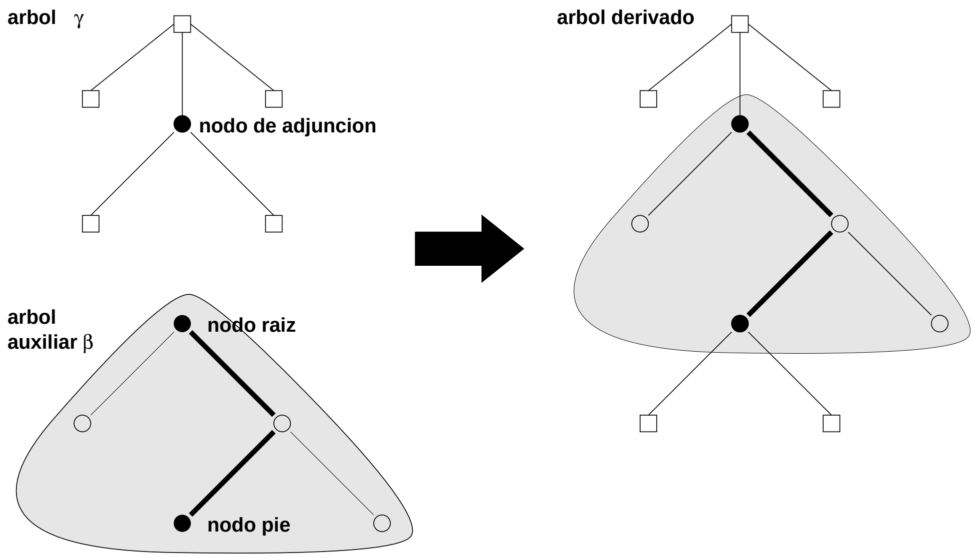

Figura 2.2: Operación de adjunción

- Restricciones de *adjunción nula* (NA) que impiden la realización de adjunciones[3].
- Restricciones de *adjunción obligatoria* (OA) que especifica un subconjunto de árboles auxiliares, uno de los cuales ha de ser utilizado obligatoriamente en una operación de adjunción.

El lenguaje definido por una gramática de adjunción de árboles es el conjunto de cadenas $w \in V_T^*$ tal que w constituye la frontera de un árbol derivado a partir de un árbol inicial[4].

Ejemplo 2.2 En la figura 2.3 se muestra la gramática de adjunción de árboles $(V_T, V_N, \boldsymbol{I}, \boldsymbol{A}, S)$ donde $V_T = \{a, b, c, d\}$, $V_N = \{S, A, B\}$, $\boldsymbol{I} = \{\alpha\}$, $\boldsymbol{A} = \{\beta_1, \beta_2, \beta_3\}$ y S es el axioma de la gramática. Dicha gramática genera el lenguaje $a^n b^m c^n d^m$ para $n, m \geq 1$. Junto a cada nodo se muestran las restricciones de adjunción. Si no se especifica un conjunto de árboles en una restricción, se supone que la restricción es válida para todos los árboles auxiliares. Los nodos sin anotación se supone que pueden realizar o no una adjunción con cualquier árbol auxiliar. Los nodos pie se señalan mediante un asterisco. En la parte izquierda de la figura 2.4 se muestra el árbol derivado para la cadena de entrada *aabbbccddd* y en la figura 2.5 se muestran las relaciones cruzadas entre los diferentes elementos de dicha cadena. ¶

2.2.2 Árbol de derivación

A diferencia de las gramáticas independientes del contexto, en las cuales el árbol derivado contiene toda la información necesaria para determinar qué operaciones han sido realizadas sobre qué nodos a lo largo de una derivación, en las gramáticas de adjunción dicha información no se puede obtener directamente a partir del árbol derivado [215, 161, 142]. En consecuencia, surge un objeto diferente, denominado *árbol de derivación*, que especifica de modo inequívoco

[3]Una restricción de adjunción nula es equivalente a una restricción selectiva donde el conjunto especificado es vacío.

[4]Aunque en [93] se diferencia entre *frontera* (secuencia de los nodos que constituyen las hojas de un árbol) y *cosecha* (secuencia de las etiquetas en los nodos de la frontera), en los artículos posteriores sobre TAG se obvia tal diferencia y se utiliza el término frontera indistintamente en uno y otro caso.

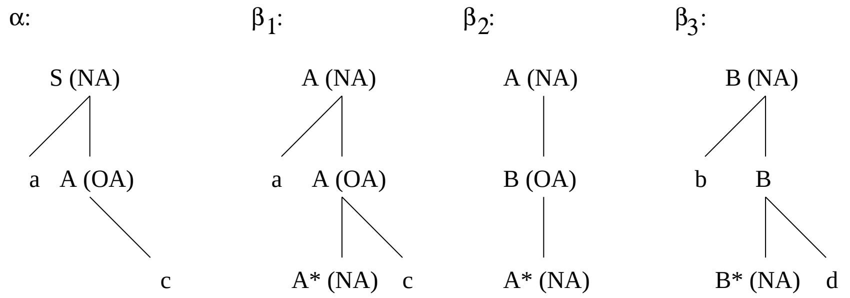

Figura 2.3: Gramática de adjunción de árboles que genera el lenguaje $a^n b^m c^n d^m$

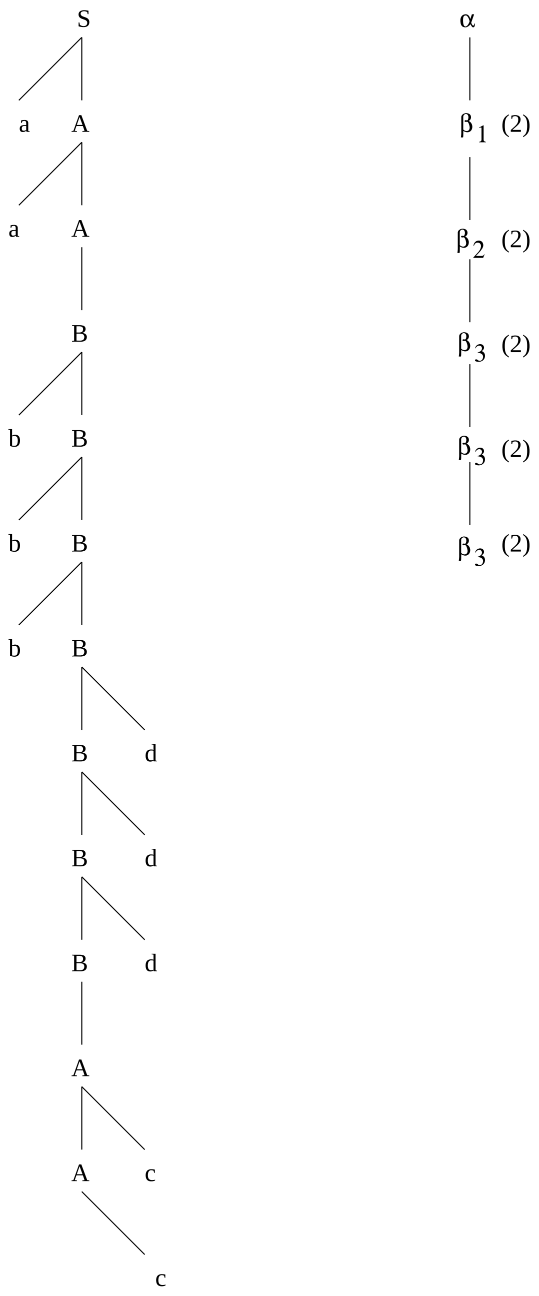

Figura 2.4: Árbol derivado (izquierda) y de derivación (derecha) en TAG para *aabbbccddd*

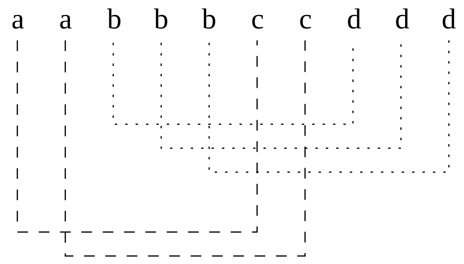

Figura 2.5: Relaciones cruzadas en la cadena *aabbbccddd*

cómo ha sido construido un árbol derivado, esto es, especifica una sucesión de adjunciones indicando para cada una de ellas el nodo en el cual tuvo lugar y el árbol auxiliar involucrado. La raíz de un árbol de derivación deberá estar etiquetada por un árbol inicial. Los demás nodos del árbol de derivación estarán etiquetados por un árbol auxiliar y por el nodo en el que se realizó la adjunción. Habitualmente se utilizan direcciones de Gorn para referirse únivocamente a los nodos de un árbol elemental[5]. En la parte derecha de la figura 2.4 se muestra el árbol de derivación correspondiente al análisis de la cadena *aaabbbcccddd* según la gramática de la figura 2.3.

2.2.3 TAG lexicalizadas

Una *gramática* se dice que está *lexicalizada* si toda estructura elemental está asociada con un símbolo terminal, denominado *ancla*. En tal caso, también podemos considerar una gramática como un lexicón en el que cada palabra está asociada con un conjunto de estructuras sintácticas elementales en las que dicha palabra actúa como ancla.

Una gramática de adjunción de árboles se dice que está lexicalizada si cada uno de los árboles elementales contiene al menos un símbolo terminal en su frontera. Dicho terminal es el *ancla* del árbol elemental. Para facilitar la descripción de los árboles elementales, en las TAG lexicalizadas (*Lexicalized Tree Adjoining Grammars*, LTAG) se permite que cualquier no-terminal etiquete la raíz de un árbol inicial, con lo que se relaja la condición que establece que la raíz de los árboles iniciales debe estar etiquetada por el axioma de la gramática. Como consecuencia, además de la operación de adjunción se define la operación de *sustitución*, por la cual se permite que un árbol inicial se pegue en un *nodo de sustitución* de la frontera de otro árbol elemental con la condición de que el no-terminal que etiqueta dicho nodo de sustitución coincida con la etiqueta de la raíz del árbol a sustituir[6]. Los nodos de sustitución no pueden actuar como nodos de adjunción.

Ejemplo 2.3 La gramática de adjunción de árboles de la figura 2.3 no está lexicalizada puesto que la frontera del árbol auxiliar β_2 no contiene ningún terminal. En la figura 2.6 se muestra una versión lexicalizada de dicha gramática, en la cual se ha modificado la forma del árbol β_2 y se han añadido dos árboles iniciales α_2 y α_3. Los nodos marcados con $\downarrow$ son nodos de sustitución. En consecuencia, el árbol α_2 puede ser sustituido en los nodos etiquetados por $C\downarrow$ de los árboles α_1, β_1 y β_2, mientras que el árbol α_3 puede ser sustituido en el nodo etiquetado por $D\downarrow$ del árbol β_3. Podemos ver esta gramática como un lexicón en el que el terminal a determina las estructuras sintácticas definidas por los árboles α_1, β_1 y β_2, el terminal b determina la estructura

[5] En el direccionamiento de Gorn se utiliza 0 para referirse al nodo raíz, k para referirse al k-ésimo hijo del nodo raíz y $p.q$ para referirse al q-ésimo hijo del nodo con dirección p.

[6] En una TAG lexicalizada la frontera de un árbol elemental puede contener nodos etiquetados por terminales, un nodo pie etiquetado por un no-terminal en el caso de los árbol auxiliares, y nodos etiquetados por no-terminales siempre que dichos nodos sean marcados como nodos de sustitución.

definida por β_3, el terminal c la estructura definida por α_2 y el terminal d la estrutura definida por α_3. ¶

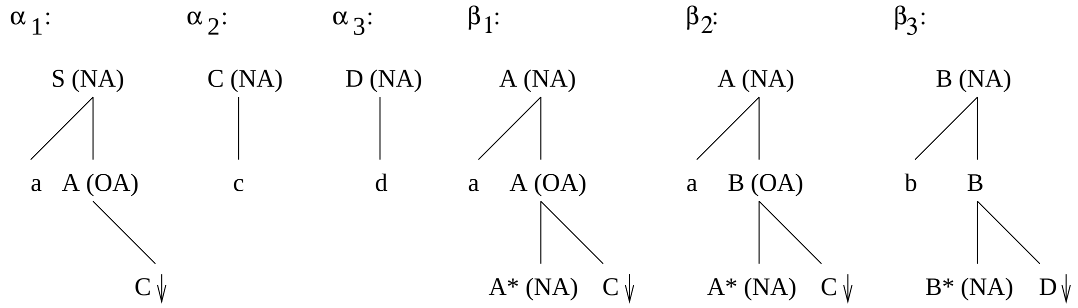

Figura 2.6: TAG lexicalizada que genera el lenguaje $a^n b^m c^n d^m$

Chen y Vijay-Shanker proponen en [47] un método para la extracción automática de LTAG a partir de un banco de árboles (*treebank*). Una gramática de adjunción de árboles lexicalizada que pretenda describir una parte considerable de los fenómenos sintácticos de una lengua puede llegar a adquirir un tamaño muy grande. Vijay-Shanker y Schabes en [212] y Evans et al. en [72] estudian el problema del almacenamiento compacto de grandes gramáticas de adjunción de árboles lexicalizadas. Para ello proponen una organización jerárquica del lexicón y la utilización de reglas léxicas y sintácticas que especifican descripciones parciales de árboles.

2.2.4 Propiedades

Sea el *conjunto de árboles* de una TAG el conjunto de árboles derivados a partir de un árbol inicial[7]. El lenguaje generado por una TAG se define como la frontera de todos los árboles en su conjunto de árboles. Denominamos *lenguajes de adjunción de árboles* (TAL) a los lenguajes generados por las gramáticas de adjunción de árboles

Las propiedades más importantes de los lenguajes y gramáticas de adjunción de árboles son las siguientes:

- Los lenguajes independientes del contexto están propiamente incluidos en los lenguajes de adjunción de árboles, aunque las gramáticas de adjunción de árboles pueden asignar a las cadenas de un lenguaje independiente del contexto una estructura que es imposible de generar utilizando gramáticas independientes del contexto [91].

  Ejemplo 2.4 La gramática mostrada en la parte izquierda de la figura 2.7 genera el lenguaje $\{a^n b^n e\}$ con $n \geq 0$, que es independiente del contexto, pero asigna a la cadena *aabbe* el árbol mostrado en la parte derecha de la misma figura, que no puede ser generado por ninguna gramática independiente del contexto. ¶

- Los lenguajes de adjunción de árboles están propiamente incluidos en los lenguajes de índices [218].

[7] En el caso de TAG lexicalizadas el conjunto de árboles se define como el conjunto de árboles elementales completados (sin nodos de sustitución en su frontera) derivados a partir de un árbol inicial cuyo nodo raíz está etiquetado por el axioma de la gramática.

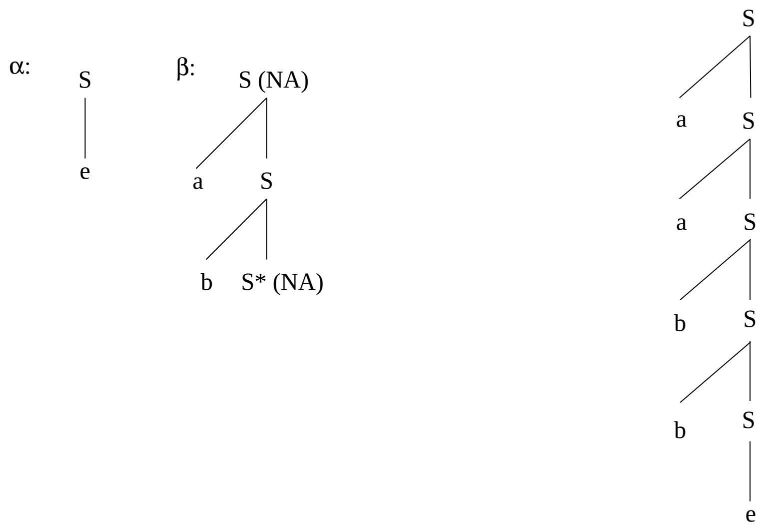

Figura 2.7: Gramática de adjunción de árboles para $a^n b^n e$ y árbol derivado para *aabbe*

- Los lenguajes de adjunción de árboles forman una familia abstracta de lenguajes completa (*Full AFL*) [85], por lo que son cerrados bajo intersección con lenguajes regulares, homomorfismo directo, homomorfismo inverso, sustitución, unión, concatenación y cierre de Kleene [209].

- Los lenguajes de adjunción de árboles pueden ser analizados en tiempo polinomial [8].

- Las gramáticas de adjunción de árboles permiten capturar ciertas dependencias cruzadas, tales como las mostradas en la figura 2.5.

- Los lenguajes de adjunción de árboles son semilineales [230].

2.2.5 Relevancia lingüística

Las gramáticas de adjunción de árboles constituyen un formalismo de generación de árboles con algunas propiedades atractivas para caracterizar las descripciones estructurales asociadas con las frases de las lenguas naturales, lo que hace de las TAG un formalismo adecuado tanto para el análisis como para la generación [117]. Entre estas propiedades las más interesantes son:

- El *dominio extendido de localidad*. Las gramáticas de adjunción de árboles poseen un dominio de localidad más amplio que las gramáticas independientes del contexto y que las gramáticas basadas en un esqueleto independiente del contexto, tales como las gramáticas de estructura de frase dirigidas por el núcleo (*Head-Driven Phrase Structure Grammar*, HPSG) [147] y las gramáticas léxico-funcionales (*Lexical Functional Grammars*, LFG) [96, 132], de tal modo que permiten la localización de dependencias de larga distancia dentro de una misma estrutura elemental.

  Ejemplo 2.5 Para ilustrar esta propiedad utilizaremos una pequeña porción de gramática del inglés, tomada de [94]. Consideremos una gramática independiente del contexto con las siguientes producciones:

$$\begin{gathered} S \to NP \;\; VP \\ VP \to VP \;\; ADV \\ VP \to V \;\; NP \\ NP \to \text{Harry} \\ NP \to \text{peanuts} \\ V \to \text{likes} \\ ADV \to \text{passionately} \end{gathered}$$

donde la dependencia entre el verbo *likes* y dos argumentos, sujeto (*Harry*) y objeto (*peanuts*), se especifica mediante las tres primeras producciones de la gramática. No es posible especificar esta dependencia en una única producción sin abandonar el nodo VP (sintagma verbal) en la estructura. Por ejemplo, si introducimos la producción $S \to NP \;\; V \;\; NP$ estaremos expresando las dependencias entre sujeto, verbo y objeto en una sóla producción, pero entonces no podremos tener VP en la gramática. En consecuencia, al tomar las las producciones independientes del contexto como especificaciones del dominio de localidad, no es posible expresar localmente la dependencia entre un verbo y sus argumentos y mantener el nodo VP en la gramática.

En las gramáticas de adjunción de árboles el dominio de localidad es más amplio, puesto que viene especificado no ya por producciones independientes del contexto sino por árboles. En la gramática de adjunción de árboles lexicalizada de la figura 2.8 se observa que el árbol anclado por *likes* especifica la dependencia entre dicho verbo, su sujeto y su complemento, manteniendo el nodo VP en la gramática. ¶

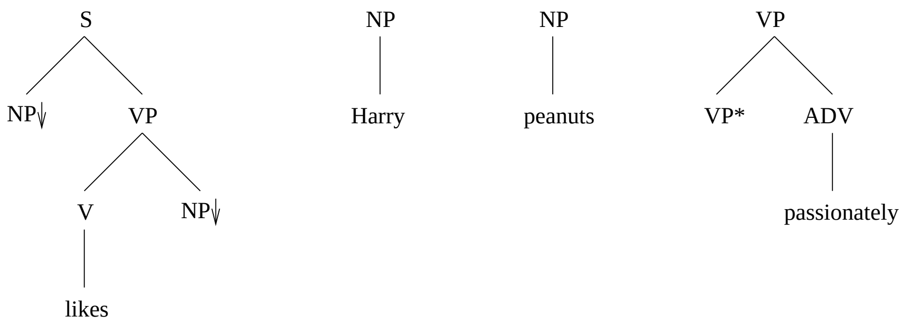

Figura 2.8: Dominio extendido de localidad de las TAG

- La *factorización de la recursión del dominio de dependencias.* Los árboles elementales, estructuras elementales de las gramáticas de adjunción de árboles, son los dominios sobre los cuales se establecen dependencias tales como la concordancia, subcategorización y relleno de huecos. La operación de adjunción, mediante la inserción de árboles auxiliares dentro de árboles elementales, permite que tales dependencias sean de larga distancia, aunque hayan sido especificadas localmente en un sólo árbol elemental [101].

Ejemplo 2.6 Consideremos la TAG lexicalizada mostrada en la figura 2.9. Mediante la adjunción del árbol auxiliar anclado por *tell* en el nodo interior etiquetado por S' del árbol inicial anclado por *likes*, obtenemos el árbol derivado de la figura 2.10, correspondiente a la frase who_i *did John tell Sam that Bill likes* ϵ_i. Es importante resaltar que en la gramática la dependencia entre who_i y el hueco ϵ_i (indicada mediante una línea discontinua en la

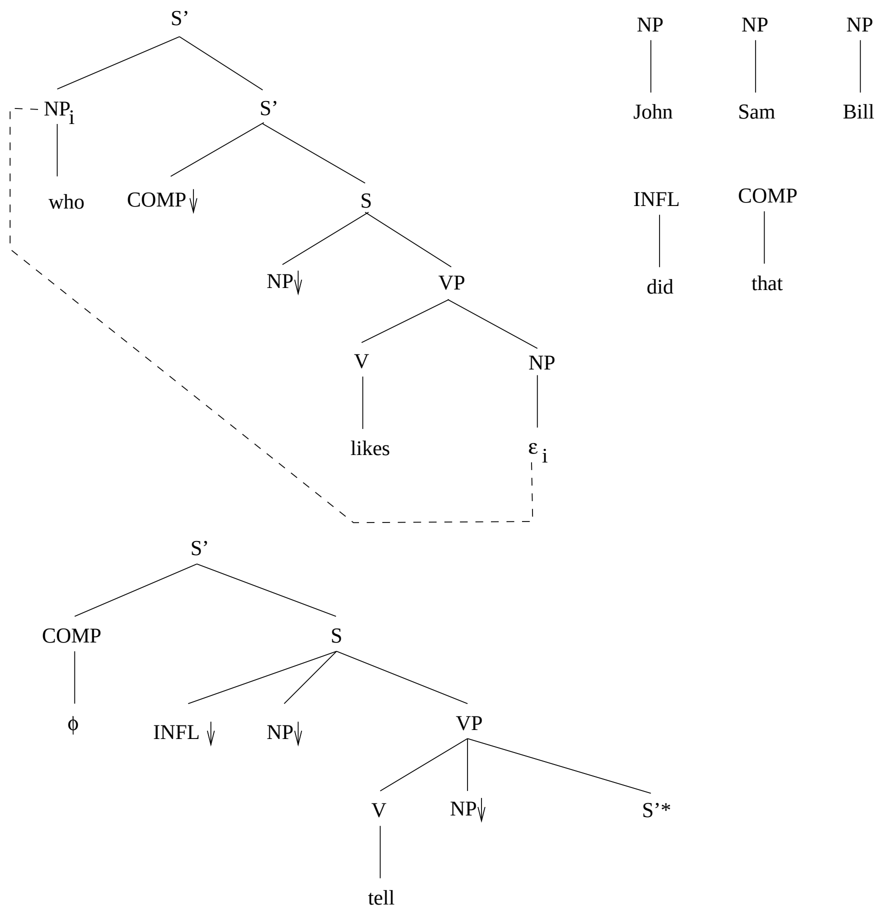

Figura 2.9: Dominio de localidad de la dependencia entre who_i y ϵ_i

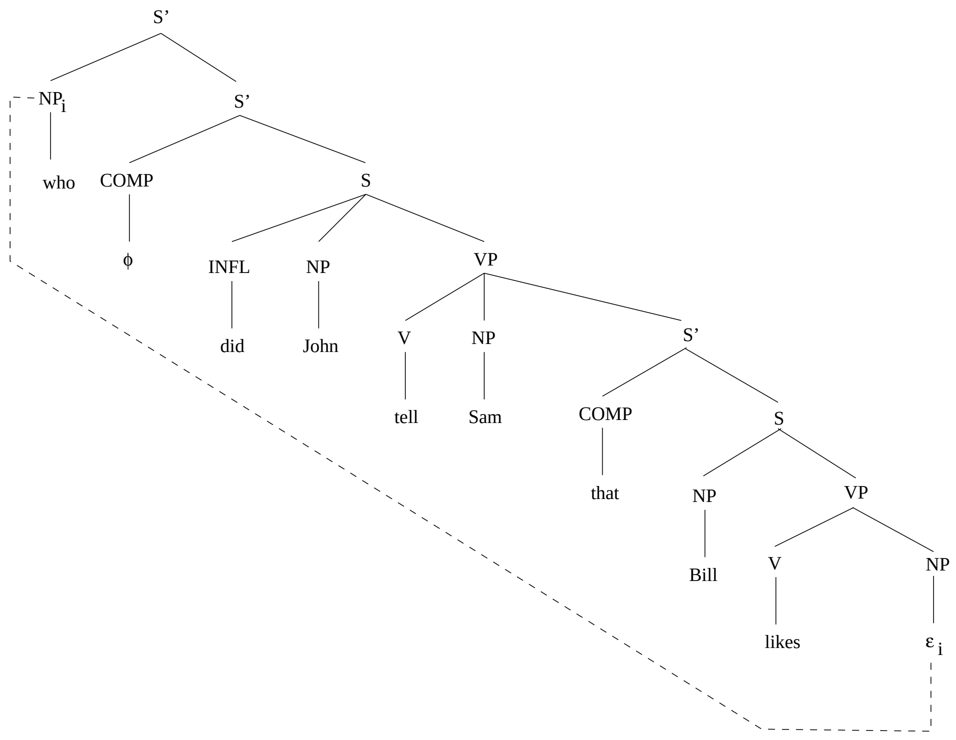

Figura 2.10: Dependencia de larga distancia entre who_i y ϵ_i

figura 2.10) aparece en el dominio de localidad definido por el árbol anclado por *likes*. En el árbol derivado de la figura 2.10 ambos elementos se han alejado como resultado de una adjunción y por tanto dicha dependencia se ha convertido en una dependencia de larga distancia. ¶

2.3 Formalismos derivados de TAG

Las gramáticas de adjunción de árboles han sido tomadas como formalismo base para la creación de varios formalismos gramaticales. A continuación comentamos someramente las características principales de los más importantes.

2.3.1 TAG basadas en unificación

Una *estructura de rasgos* [40] está formada por un conjunto de pares atributo-valor tal que un valor puede ser atómico u otra estructura de rasgos. Vijay-Shanker y Joshi presentan en [211] las TAG basadas de unificación (*Unification-based Tree Adjoining Grammars*, UTAG), una variante de las gramáticas de adjunción de árboles en la cual los nodos de los árboles elementales pueden estar decorados con estructuras de rasgos que describen el nodo y su relación con otros nodos del mismo árbol. Las operaciones de adjunción y sustitución se definen en términos de la unificación de estruturas de rasgos, por lo que las restricciones de adjunción pueden ser modeladas a través del éxito o del fallo de la unificación entre las estruturas de rasgos de los nodos.

Una UTAG en la cual no se permiten adjunciones en el nodo pie y en la que las estructuras de rasgos tienen siempre un tamaño finito, se denomina *gramática de adjunción de árboles basada en estructuras de rasgos* (*Feature structures based Tree Adjoining Grammars*, FTAG) [210]. La prohibición de adjunción en el nodo pie viene motivada lingüísticamente con el fin de no alterar las relaciones gramaticales definidas por los árboles. La utilización de estructuras de rasgos de tamaño finito viene motivada lingüísticamente por el hecho de que los fenómenos de subcategorización, que en otros formalismos se especifican mediante apilamientos especificados en alguno de los componentes de la estructura de rasgos, en el caso de TAG se pueden definir directamente en los árboles elementales, por lo que no es preciso realizar ningún tipo de apilamientos en las estructuras de rasgos.

2.3.2 TAG estocásticas

Las producciones de una gramática independiente del contexto se pueden anotar con probabilidades de tal modo que la probabilidad de una derivación es calculable mediante la realización de sumas y productos [197], puesto que al ser las producciones independientes del contexto las probabilidades asociadas también son independientes.

Carrol y Weir estudian en [45] la asignación de probabilidades a gramáticas de adjunción de árboles lexicalizadas, proponiendo cuatro modelos diferentes de TAG estocásticas que enumeramos según la capacidad creciente para describir fenómenos derivacionales:

1. El primer modelo sólo asocia probabilidades a los árboles elementales de tal modo que la la suma de las probabilidades de todos los árboles auxiliares con la misma etiqueta en la raíz sume 1 y la suma de las probabilidades de los árboles iniciales con la misma etiqueta en la raíz sume también 1.

2. El segundo modelo es equivalente a las TAG probabilísticas (*Probabilistic Tree Adjoining Grammars*, PTAG) definidas por Resnik en [158] y a las TAG lexicalizadas estocásticas (*Stochastic Lexicalized Tree Adjoining Grammars*, SLTAG) propuestas por Schabes en [170], en las que la suma de las probabilidades de que una derivación comience por

un árbol inicial debe ser 1, la suma de las probabilidades de adjunción en un nodo debe ser igual a 1 y la suma de las probabilidades de sustitución en un nodo deber ser también igual a 1. Neumann propone en [133] un método para la extracción automática de SL-TAG a partir de un banco de árboles. Nederhof et al. proponen en [128] un método para calcular la probabilidad de los prefijos de las cadenas de un lenguaje a partir de una TAG estocástica, no necesariamente lexicalizada.

3. En el tercer modelo se asocian probabilidades con una meta-gramática independiente del contexto que codifica las posibles derivaciones de la gramática. La suma de las probabilidades de las meta-producciones asociadas a un árbol dado debe ser 1.

4. El cuarto modelo considera una gramática de sustitución de árboles estocástica obtenida a partir de un banco de árboles (*treebank*) en la que cada árbol tiene una probabilidad asociada. Las probabilidades de todos los árboles con el mismo símbolo no-terminal deben sumar 1.

Asumiendo que la probabilidad asociada a cada derivación de una cadena de entrada está bien definida, la probabilidad de una cadena es igual a la suma de las probabilidades de todas las derivaciones de dicha cadena. Se dice que una gramática probabilística es *consistente* si las probabilidades asociadas a todas las cadenas del lenguaje suman 1. Sarkar estudia en [164] las condiciones que debe satisfacer una PTAG para garantizar que es consistente.

2.3.3 TAG con dominación local y precedencia lineal

Las gramáticas de adjunción de árboles con dominación local y precedencia lineal [95] TAG(LD/LP) son una variante de las TAG en las cuales los árboles elementales pasan a ser *estructuras elementales* que especifican únicamente relaciones de dominación local sobre las cuales se pueden establecer relaciones de precedencia lineal. Quiere esto decir que las estructuras elementales son como árboles pero no establecen un orden a priori entre nodos hermanos, sino que dicho orden se establece mediante relaciones de precedencia que indican si un nodo debe aparecer antes o después de alguno de los otros nodos del árbol. Las relaciones de precedencia pueden establecerse entre nodos que no son hermanos pero que pertenecen al mismo dominio de localidad (estructura elemental).

Las TAG(LD/TLP) son una versión restringida de las TAG(LD/LP) que preservan los árboles elementales puesto que fuerzan el cumplimiento de la siguiente *condición de consistencia*: dada una estructura elemental sobre la cual se establece que un nodo N precede a un nodo M, si N domina al nodo P y M domina al nodo Q entonces P debe preceder a Q. Minnen [119] y Poller [148] estudian la extensión del algoritmo sintáctico de Earley al caso de las gramáticas TAG(LD/TLP).

2.3.4 TAG síncronas

Las gramáticas de adjunción de árboles síncronas [187] son una variante de TAG que caracterizan correspondencias entre lenguajes, por lo que son frecuentemente utilizadas en traducción automática [3, 46]. Tanto el lenguaje de partida, denominado *lenguaje fuente*, como el lenguaje al que se desea traducir, denominado *lenguaje objetivo*, se define mediante gramáticas de adjunción de árboles. Ambas gramáticas están sincronizadas en el sentido de que las operaciones de adjunción y sustitución se aplican simultáneamente a nodos relacionados de pares de árboles, uno de cada lenguaje. Chiang et al. presentan en [48] las gramáticas de adjunción de árboles de dos niveles en forma regular (*Regular Form–2 Level Tree Adjoining Grammars*, RF-2LTAG) como una extensión de las TAG síncronas que permite coordinar árboles de derivación que no son isomórfos.

2.3.5 TAG multicomponente

Las gramáticas de adjunción de árboles multicomponente (*Multicomponent Tree Adjoining Grammars*, MCTAG) son una extensión de TAG propuesta inicialmente en [93] y posteriormente refinada en [91] en la cual la operación de adjunción se aplica a secuencias de árboles. El objetivo de MCTAG es extender el dominio de localidad de TAG de árboles a secuencias de árboles.

Una MCTAG está formada por un conjunto finito de secuencias de árboles. Habitualmente se consideran las cuatro versiones siguientes de MCTAG en función de cómo se defina la operación de adjunción con respecto a las secuencias de árboles [230, 152]:

MCTAG de árbol local: la adjunción se realiza mediante la adjunción de cada uno de los árboles de una secuencia en diferentes nodos de un único árbol elemental. El formalismo resultante es fuertemente equivalente a las gramáticas de adjunción de árboles estándar [94].

MCTAG de conjunto local: la adjunción se realiza mediante la adjunción de cada uno de los árboles de una secuencia en diferentes nodos de árboles elementales que pertenecen a una misma secuencia. El formalismo resultante es débilmente equivalente a los sistemas de reescritura independientes del contexto lineales (*Linear Context-Free Rewriting Systems*, LCFRS) [217] .

MCTAG de árbol no local: la adjunción se realiza mediante la adjunción de cada uno de los árboles de una secuencia en diferentes nodos de un único árbol derivado.

MCTAG de conjunto no local: la adjunción se realiza mediante la adjunción de cada uno de los árboles de una secuencia en diferentes nodos de árboles elementales que pertenecen a una misma secuencia de árboles derivados.

Las MCTAG no locales generan lenguajes que no son semilineales y, por lo tanto, quedan fuera de la clase de los lenguajes suavemente dependientes del contexto.

2.3.6 Gramáticas de descripción de árboles

Las gramáticas de descripción de árboles (*D-Tree Grammars*, DTG) [155] se derivan de TAG, pero a diferencia de estas en vez de manipular árboles elementales manipulan *árboles de descripción* (árboles-D). Los árboles-D contienen dos tipos de aristas, *aristas de dominación* (aristas-d) y *aristas de dominación inmediata* (aristas-i). Dado un nodo de un árbol-D, todos sus hijos deben estar enlazados mediante aristas-i o bien debe tener un único hijo enlazado por una arista-d.

Los lenguajes generados por DTG y TAG son incomparables, pues algunos lenguajes generados por TAG no pueden ser generados por DTG y viceversa. Por ejemplo, el lenguaje de copia wcw con $w \in V_T^*$ es un lenguaje de adjunción de árboles pero no puede ser generado por ninguna DTG, mientras que el lenguaje $a^n b^n c^n d^n f e^n$ no es un lenguaje de adjunción de árboles pero puede ser generado por una DTG [208].

Vijay-Shanker discute la relación entre DTG y TAG en [207]. Carrol et al. describen en [44] la utilización de DTG lexicalizadas en el proyecto LexSys. Smet compara en [192] la TAG lexicalizada utilizada en el proyecto XTAG [198] con la DTG utilizada en el proyecto LexSys. Smets y Evans describen en [193] un método para representar de modo compacto una DTG. Carrol et al. estudian en [43] un método para compactar las DTG basado en la codificación de los recorridos de los árboles-D en un autómata finito con el fin de aumentar la eficiencia de los analizadores sintácticos para DTG. Rambow et al. proponen en [156] una extensión del algoritmo de Earley para realizar el análisis sintáctico de DTG.

Frank et al. estudian en [74] las relaciones de dominancia en análisis sintáctico basado en descripciones. Hepple estudia en [84] las relaciones entre DTG y las gramáticas lógicas de tipos. Candito y Kahane definen en [39] un formalismo derivado de DTG denominado GAG cuyas derivaciones inducen grafos de dependencias semánticas.

2.3.7 Gramáticas de inserción de árboles

Las gramáticas independientes del contexto no están en general lexicalizadas. Basta con que alguna de sus producciones (estructuras elementales) no contenga ningún símbolo terminal para que la gramática no se encuentre lexicalizada. Las gramáticas independientes del contexto en forma normal de Greibach [85] están lexicalizadas puesto que dicha forma normal impone la existencia de un terminal en el lado derecho de las producciones. Toda gramática independiente del contexto puede ser normalizada, sin embargo la gramática en forma normal de Greibach obtenida no preserva la forma de los árboles derivados por la gramática original. Decimos en este caso que se trata de un proceso de *lexicalización débil*.

Para conservar la forma de los árboles derivados (*lexicalización fuerte*) es preciso que el formalismo gramatical objetivo de la lexicalización manipule árboles en lugar de producciones. Las TAG lexicalizadas (LTAG) se presentan de forma natural como un formalismo objetivo adecuado. Siguiendo este enfoque, Carrillo Montero y Díaz Madrigal en [42] y Joshi y Schabes en [94] presentan métodos de conversión de CFG en LTAG.

La desventaja de utilizar LTAG como formalismo objetivo en el proceso de lexicalización de gramáticas independientes del contexto es que la complejidad temporal del análisis sintáctico se incrementa de $\mathcal{O}(n^3)$ a $\mathcal{O}(n^6)$, donde n es la longitud de la cadena de entrada. Surge entonces el reto de diseñar un formalismo gramatical similar a LTAG que permita la lexicalización de gramáticas independientes del contexto y que sea analizable en tiempo $\mathcal{O}(n^3)$.

Schabes [171] y Schabes y Waters [177] definen una forma restringida de LTAG que da lugar a las *gramáticas independientes del contexto lexicalizadas* (*Lexicalized Context-Free Grammars*, LCFG). Al igual que LTAG, las LCFG vienen definidas por una tupla $(V_T, V_N, \boldsymbol{I}, \boldsymbol{A}, S)$, donde V_T es un conjunto finito de símbolos terminales, V_N es un conjunto finito de símbolos no-terminales, $\boldsymbol{I}$ y $\boldsymbol{A}$ son conjuntos finitos de árboles iniciales y auxiliares y S es el axioma de la gramática. La diferencia con LTAG radica en los siguientes puntos:

- Dado un árbol auxiliar, el conjunto de nodos de la frontera a la izquierda (resp. a la derecha) del nodo pie debe ser vacío o bien cada uno de los nodos en dicho conjunto está etiquetado por la palabra vacía ϵ, dando lugar a árboles auxiliares izquierdos (resp. árboles auxiliares derechos).

- La operación de adjunción se restringe de modo que se prohibe que un árbol auxiliar izquierdo sea adjuntado en la espina de un árbol auxiliar derecho y viceversa, un árbol auxiliar derecho no puede ser adjuntado en la espina de un árbol auxiliar izquierdo. Adicionalmente, queda prohibida la adjunción en los nodos situados a la derecha de la espina de un árbol auxiliar izquierdo y en los nodos que están a la izquierda de la espina de un árbol auxiliar derecho.

- Se permite una forma restringida de adjunción múltiple, de tal modo que a lo sumo un árbol auxiliar izquierdo y un árbol auxiliar derecho pueden ser adjuntados en un mismo nodo.

Las gramáticas independientes del contexto lexicalizadas generan únicamente lenguajes independientes del contexto, pero la utilización del árbol como estructura elemental de representación y la utilización de una forma restringida de adjunción permite que una LCFG lexicalice una gramática independiente del contexto que no sea cíclica manteniendo la forma de los árboles derivados en la gramática original. En [177] se describe un algoritmo de análisis sintáctico de tipo CYK [6] para LCFG y en [171, 177] se describe un analizador sintáctico de tipo Earley [69] para LCFG. En ambos casos el análisis sintáctico de LCFG se realiza en tiempo $\mathcal{O}(n^3)$ en el peor caso. Schabes y Waters definen en [178] una versión estocástica de LCFG.

Schabes y Waters definen en [179] una versión más elaborada de LCFG denominada *gramáticas de inserción de árboles* (*Tree Insertion Grammars*, TIG). La principal diferencia de TIG con respecto a LCFG es que se relajan las restricciones sobre la adjunción múltiple, de tal modo que en TIG se permite la adjunción de un número arbitrario de árboles auxiliares en un único nodo elemental, con la salvedad de que el conjunto de adjunciones en dicho nodo se especifica en términos de dos secuencias, una de árboles auxiliares izquierdos y otra de árboles auxiliares derechos. En [179] se describe un algoritmo de análisis sintáctico de tipo Earley para TIG que presenta una complejidad temporal $\mathcal{O}(n^3)$ en el peor de los casos. Schabes y Waters definen en [180] una versión estocástica de TIG. Neumann propone en [133] un método para la extracción automática de TIG lexicalizadas estocásticas a partir de un banco de árboles. Hwa realiza en [87] una evaluación empírica de TIG lexicalizadas probabilísticas (*Probabilistic Lexicalized Tree Insertion Grammars*, PLTIG).

Ejemplo 2.7 Sea $\mathcal{G} = (V_N, V_T, P, A_1)$ una gramática indpendiente del contexto donde $V_N = \{A_1, A_2\}$, $V_T = \{a\}$, A_1 es el axioma de la gramática y P contiene las producciones:

$$A_1 \to A_2 A_2$$

$$A_2 \to A_1 A_2$$

$$A_2 \to A_2 A_1$$

$$A_2 \to a$$

El lenguaje generado por $\mathcal{G}$ es $\{(aa)^+\}$. La gramática de inserción de árboles de la figura 2.11 lexicaliza $\mathcal{G}$ preservando la forma de los árboles derivados. ¶

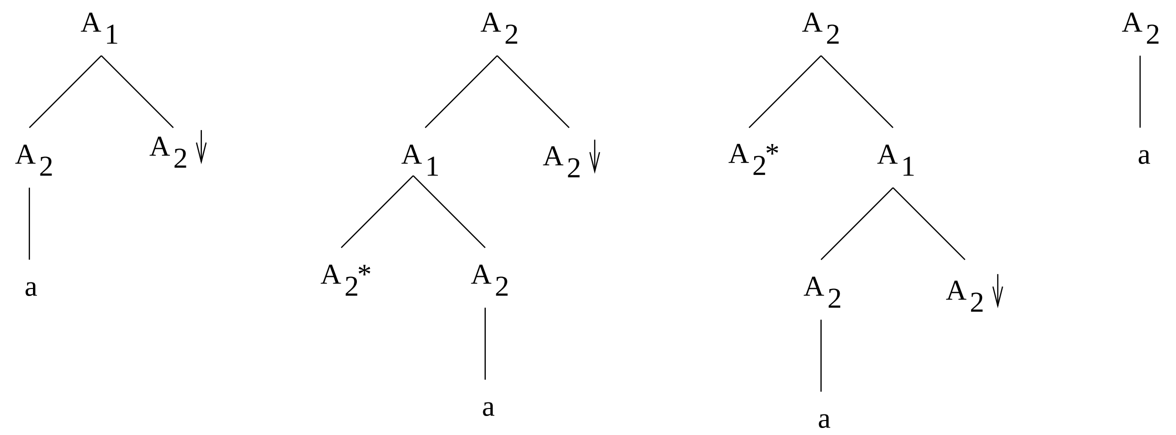

Figura 2.11: Gramática de inserción de árboles

2.3.8 TAG en forma regular

Rogers estudia en [160] el problema de la lexicalización de gramáticas independientes del contexto siguiendo un enfoque distinto al de Schabes y Waters. La idea de Rogers se basa en restringir la operación de adjunción de tal modo que solo se puedan generar árboles derivados cuyos conjuntos de caminos sean un lenguaje regular. En concreto, las restricciones a aplicar son las siguientes:

- Un árbol auxiliar puede ser adjuntado en cualquier nodo de un árbol inicial o bien en cualquier nodo de un árbol auxiliar excepto en los nodos de la espina.
- Sea un *árbol auxiliar propio* aquel en el cual ningún nodo de la espina comparte etiqueta con el nodo raíz salvo el nodo pie. Un árbol auxiliar propio puede ser adjuntado en la raíz o en el pie de cualquier árbol auxiliar.
- Un árbol auxiliar β_1 puede ser adjuntado en cualquier nodo de la espina de un árbol auxiliar β_2 siempre que ningún ejemplar de β_2 pueda ser adjuntada en la espina de β_1.

El problema de determinar si una TAG arbitraria está en forma regular es decidible [160].

2.4 Gramáticas lineales de índices

Las Gramáticas de Índices (*Indexed Grammars*, IG) [4] son una extensión de las gramáticas independientes del contexto en las cuales cada símbolo no-terminal tiene asociado una pila de índices[8]. Denotaremos mediante $A[\alpha]$ al símbolo formado por el no-terminal A y la pila de índices α, la cual puede estar vacía, en cuyo caso se representa por $[\ ]$. Si restringimos la forma de las producciones de tal modo que la pila asociada al no-terminal del lado izquierdo de una producción (denominado *padre*) sólo pueda transmitirse a un no-terminal del lado derecho (denominado *hijo dependiente*) y los demás no-terminales estén asociados con pilas de tamaño acotado, obtenemos las Gramáticas Lineales de Índices (*Linear Indexed Grammars*, LIG) [75]. Formalmente, una LIG es una quíntupla (V_T, V_N, V_I, S, P) donde:

- V_T es un conjunto finito de símbolos terminales.
- V_N es un conjunto finito de símbolos no-terminales. Se cumple que $V_T \cap V_N = \epsilon$.
- V_I es un conjunto finito de símbolos índice, elementos que se almacenan en las pilas asociadas a los no-terminales.
- $S \in V_N$ es el axioma de la gramática.
- P es un conjunto finito de producciones que tienen alguna de las formas siguientes:

$$A[\circ\circ\gamma] \rightarrow \Upsilon_1\ B[\circ\circ]\ \Upsilon_2$$

$$A[\circ\circ] \rightarrow \Upsilon_1\ B[\circ\circ]\ \Upsilon_2$$

$$A[\circ\circ] \rightarrow \Upsilon_1\ B[\circ\circ\gamma]\ \Upsilon_2$$

$$A[\ ] \rightarrow a$$

[8] No debemos confundir el formalismo gramatical tratado en esta sección con los Lenguajes Lineales de Índices definidos por Duske y Parchmann en [68], pues estos últimos se refieren a los lenguajes generados por gramáticas de índices que son lineales en el sentido de que el lado derecho de cada producción puede contener a lo sumo un no-terminal.

donde $A, B \in V_N$, A es el padre, B es el hijo dependiente, $\gamma \in V_I$, $\circ\circ$ representa la parte de la pila transmitida del padre al hijo dependiente, $a \in V_T \cup \{\epsilon\}$ y $\Upsilon_1, \Upsilon_2 \in (V_N[\,])^*$. Por conveniencia, habitualmente se permiten terminales en Υ_1 y Υ_1, por lo que $\Upsilon_1, \Upsilon_2 \in (V_N[\,] \cup V_T)^*$, sin que este hecho afecte a la capacidad generativa ni a la forma de los árboles producidos.

A continuación definimos la relación de derivación $\Rightarrow$ en LIG. Decimos que $\Upsilon \Rightarrow \Upsilon'$ si:

- $\Upsilon = \Upsilon_1 A[\alpha\gamma]\Upsilon_4$, existe una producción $A[\circ\circ\gamma] \to \Upsilon_2 B[\circ\circ\gamma']\,\Upsilon_3$ y $\Upsilon' = \Upsilon_1\Upsilon_2 B[\alpha\gamma']\,\Upsilon_3\Upsilon_4$, con $A, B \in V_N$, $\alpha \in V_I^*$, $\gamma, \gamma' \in V_I \cup \{\epsilon\}$ y $\Upsilon_1, \Upsilon_2, \Upsilon_3, \Upsilon_4 \in (V_N[V_I^*] \cup V_T)^*$. En este caso decimos que $B[\alpha\gamma']$ es el *descendiente dependiente* de $A[\alpha\gamma]$ en la derivación.

- O bien $\Upsilon = \Upsilon_1 A[\,]\,\Upsilon_4$ y existe una producción $A[\,] \to a$ de modo que $\Upsilon' = \Upsilon_1\ a\ \Upsilon_4$, donde $A \in V_N$ y $\Upsilon_1, \Upsilon_4 \in (V_N[V_I^*] \cup V_T)^*$.

Denotaremos mediante $\stackrel{*}{\Rightarrow}$ el cierre reflexivo y transitivo de $\Rightarrow$. El lenguaje generado por una LIG queda definido por todos los $w \in V_T^*$ tales que $S[\,] \stackrel{*}{\Rightarrow} w$.

Otro concepto importante es el de *espina*. Sea $A[\,]$ un hijo no dependiente en una producción o bien $A = S$. Definimos la espina de una derivación $A[\,] \stackrel{*}{\Rightarrow} \Upsilon$ que comienza en $A[\,]$ como la secuencia de los descendientes dependientes de $A[\,]$. Decimos que la espina está completa si el último descendiente dependiente tiene la forma $B[\,]$ y la producción a él aplicada es de la forma $B[\,] \to a$, donde $a \in V_T \cup \{\epsilon\}$. En una derivación pueden existir múltiples espinas. La espina principal de una derivación es aquella que comienza en $S[\,]$. La noción de espina es fundamental en la teoría de las gramáticas lineales de índices puesto que representa el camino en el cual se evalúan las pilas de índices.

Ejemplo 2.8 Consideremos la gramática lineal de índices definida por la quíntupla (V_N, V_T, V_I, P, S), donde $V_N = \{S, A, B, C\}$, $V_T = \{a, b, c, d\}$, $V_I = \{x, y\}$, S es el axioma y P es el conjunto de producciones:.

$$S[\circ\circ] \to a\ A[\circ\circ x]$$

$$A[\circ\circ] \to a\ A[\circ\circ x]$$

$$A[\circ\circ] \to b\ B[\circ\circ y]$$

$$B[\circ\circ] \to b\ B[\circ\circ y]$$

$$B[\circ\circ y] \to C[\circ\circ]\ d$$

$$C[\circ\circ y] \to C[\circ\circ]\ d$$

$$C[\circ\circ x] \to C[\circ\circ]\ c$$

$$C[\,] \to \epsilon$$

Esta gramática genera el lenguaje $\{a^n b^m c^n d^m \mid n, m \geq 1\}$, que coincide con el generado por la gramática de adjunción de árboles de la figura 2.3. En la figura 2.12 se muestra el árbol derivado para la cadena *aabbbccddd*, que posee la misma estructura que el árbol derivado mostrado en la figura 2.4 y exhibe las mismas relaciones cruzadas de la figura 2.5. La espina principal de la derivación $S[\,] \stackrel{*}{\Rightarrow}$ *aabbbccddd* está constituida por la secuencia de elementos alineados verticalmente con $S[\,]$ en la figura 2.12. Mediante ligeras modificaciones en las producciones de la gramática lineal de índices podríamos obtener árboles derivados isomorfos a los de la TAG de la figura 2.3. Un poco más adelante en esta misma sección se muestra un método que permite transformar una TAG en una LIG fuertemente equivalente. ¶

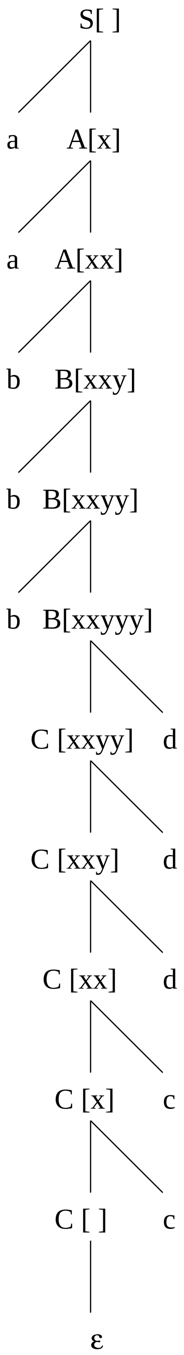

Figura 2.12: Árbol derivado en LIG para la cadena *aabbbccddd*

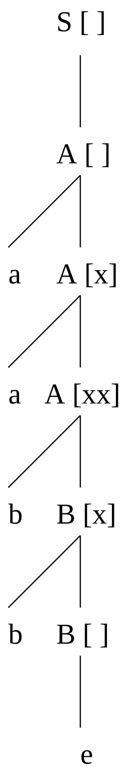

Figura 2.13: Árbol derivado en LIG para la cadena *aabbe*

Ejemplo 2.9 La gramática lineal de índices definida por la quíntupla (V_N, V_T, V_I, P, S), donde $V_N = \{S, A, B\}$, $V_T = \{a, b, e\}$, $V_I = \{x\}$, S es el axioma y P es el conjunto de producciones:

$$S[\circ\circ] \to B[\circ\circ]$$
$$S[\circ\circ] \to A[\circ\circ]$$
$$A[\circ\circ] \to a\ A[\circ\circ x]$$
$$A[\circ\circ x] \to b\ B[\circ\circ]$$
$$B[\circ\circ x] \to b\ B[\circ\circ]$$
$$B[\,] \to e$$

genera el lenguaje independiente del contexto $\{a^n b^n e \mid n \geq 0\}$, pero asigna a la cadena de entrada *aabbe* la estructura de la figura 2.13, que no puede ser creada por ninguna gramática independiente del contexto. Esta gramática genera la misma clase de lenguajes y genera árboles derivados con la misma estructura que los producidos por la gramática de adjunción de árboles de la figura 2.7. ¶

A diferencia de lo que ocurría en el caso de las gramáticas de adjunción de árboles, en las gramáticas lineales de índices podemos identificar árbol derivado y árbol de derivación, pues de la observación del primero se obtienen las posibles secuencias de producciones utilizadas para generarlo.

2.4.1 Propiedad de independencia del contexto de LIG

Una producción $A[\circ\circ\gamma] \to \Upsilon_1\ B[\circ\circ\gamma']\ \Upsilon_2$ puede ser aplicada a cualquier símbolo LIG $A[\alpha\gamma]$ formado por un no-terminal A más una pila de índices $\alpha\gamma$ asociada que tiene el elemento γ en su cima. Al igual que en el caso de las producciones independientes del contexto, dicha aplicación es independiente de cualquier otro símbolo LIG que aparezca a derecha o izquierda de $A[\alpha\gamma]$. La diferencia fundamental radica en el hecho de que existe una cierta dependencia del contexto derivada del examen del contenido de la cima de la pila de índices, puesto que γ puede ser la marca dejada para indicar que cierta producción ha sido previamente aplicada. Sin embargo, la aplicación de la producción es independiente del contenido de la parte restante de la pila de índices.

Podemos observar que es la misma clase de *suave* dependencia del contexto presente en las gramáticas de adjunción de árboles, en las que existe una dependencia entre el reconocimiento de los nodos raíz y pie de un árbol auxiliar. En el caso de las TAG la independencia con respecto al entorno del nodo de adjunción se captura en la propia definición de la operación de adjunción. En el caso de las gramáticas lineales de índices queda definida por la siguiente propiedad de independencia del contexto de LIG.

Definición 2.1 *La propiedad de independencia del contexto de LIG establece que si*

$$A[\gamma] \overset{*}{\Rightarrow} uB[\,]w$$

donde $u, v, w \in V_T^*$, $A, B \in V_N$, $B[\,]$ *es el descendiente dependiente de* $A[\gamma]$ *y* $\gamma \in V_I \cup \{\epsilon\}$, *entonces para cualquier* $\beta \in V_I^*$ *se cumple que*

$$A[\beta\gamma] \overset{*}{\Rightarrow} uB[\beta]w$$

y para cualquier $\Upsilon_1, \Upsilon_2 \in (V_N[V_I^*] \cup V_T)^*$ *se cumple que*

$$\Upsilon_1 A[\beta\gamma]\Upsilon_2 \stackrel{*}{\Rightarrow} \Upsilon_1 u B[\beta] w \Upsilon_2$$

Análogamente, si $B[\gamma]$ *es el descendiente dependiente de* $A[\,]$ *en la derivación*

$$A[\,] \stackrel{*}{\Rightarrow} uB[\gamma]w$$

entonces para cualquier Υ_1, Υ_2 *y* β *se cumple que*

$$\Upsilon_1 A[\beta]\Upsilon_2 \stackrel{*}{\Rightarrow} \Upsilon_1 u B[\beta\gamma] w \Upsilon_2$$

2.4.2 Extensiones a la notación

Una gramática independiente del contexto se puede transformar de modo inmediato en una gramática lineal de índices convirtiendo cada producción independiente del contexto

$$A_{r,0} \rightarrow A_{r,1} \ldots A_{r,m}$$

en una producción de una gramática lineal de índices

$$A_{r,0}[\circ\circ] \rightarrow A_{r,1}[\,] \ldots A_{r,d}[\circ\circ] \ldots A_{r,m}$$

Para una producción independiente del contexto con m elementos en el lado derecho existen m diferentes formas de situar el hijo dependiente. Debemos escoger una sola de esas opciones pues de lo contrario introduciríamos ambigüedades en la nueva gramática que no existirían en la original. Un modo de evitar este problema consiste en elegir sistemáticamente el primer hijo como hijo dependiente. Las producciones de tipo $A \rightarrow a$, donde $a \in V_T$, se transforman en $A[\,] \rightarrow a$.

Ejemplo 2.10 La gramática independiente del contexto $\mathcal{G} = (V_N, V_T, P, A)$, donde $V_N = \{A\}$, $V_T = \{a, b\}$, T es el axioma de la gramática y P es el conjunto de producciones

$$A \rightarrow aAb$$

$$A \rightarrow \epsilon$$

genera el lenguaje $\{a^n b^n \mid n \geq 1\}$. Podemos definir una gramática lineal de índices equivalente $\mathcal{L} = (V_N, V_T, V_I, A, P')$, donde $V_I = \emptyset$ y P' contiene el siguiente conjunto de producciones:

$$A[\circ\circ] \rightarrow a\ A[\circ\circ]\ b$$

$$A[\,] \rightarrow \epsilon$$

¶

El inconveniente de esta transformación es que las producciones expresan transmisiones de pilas y creaciones de espinas cuando realmente se están transmitiendo siempre pilas vacías que no son manipuladas en ningún momento, por lo que no intervienen en el proceso de derivación de las cadenas del lenguaje.

Teniendo en cuenta que una parte importante de las construcciones de las lenguas naturales son independientes del contexto resultaría interesante poder incorporar directamente producciones independientes del contexto en una gramática lineal de índices sin tener que definir arbitrariamente caminos para la transmisión de las pilas vacías. Esto se puede lograr de forma sencilla permitiendo que las producciones de una LIG tengan también la forma

$$A_{r,0}[\,] \to A_{r,1}[\,] \ldots A_{r,m}[\,]$$

La introducción de este tipo de producciones no modifica la capacidad generativa de las gramáticas lineales de índices, pues es fácilmente demostrable que cada una de dichas producciones es equivalente la siguiente conjunto de 3 producciones:

$$A_{r,0}[\circ\circ] \to A'_{r,0}[\circ\circ\gamma_r]$$
$$A'_{r,0}[\circ\circ\gamma_r] \to A''_{r,0}[\circ\circ]\; A_{r,1}[\,] \ldots A_{r,m}[\,]$$
$$A''_{r,0}[\,] \to \epsilon$$

donde $A'_{r,0}$ y $A''_{r,0}$ son no-terminales nuevos y γ_r es un índice nuevo, tal que ninguno de ellos es utilizado en ninguna otra parte de la gramática. La primera y la última producción evitan la utilización de la producción cuando la pila asociada a $A_{r,0}$ no está vacía.

Podemos pensar en la incorporación de reglas independientes del contexto en LIG como un fenómeno equivalente a la incorporación de la operación de sustitución, la cual es independiente del contexto, en TAG. No incrementan la capacidad generativa, pero facilitan la legibilidad de la gramática.

Ejemplo 2.11 La gramática lineal de índices $\mathcal{L} = (V_N, V_T, V_I, P, S)$, donde $V_N = \{S, C, D\}$, $V_T = \{c, d, e\}$, $V_I = \{\gamma\}$, S es el axioma y P contiene las producciones

$$S[\circ\circ] \to c\; C[\circ\circ\gamma]\; e$$
$$C[\circ\circ] \to c\; C[\circ\circ\gamma]\; e$$
$$C[\circ\circ] \to D[\circ\circ]$$
$$D[\circ\circ\gamma] \to d\; D[\circ\circ]$$
$$D[\,] \to \epsilon$$

genera el lenguaje $\{c^m d^m e^m \mid m \geq 1\}$. Para diseñar una gramática lineal de índices que genere el lenguaje $\{a^n b^n c^m d^m e^m \mid n, m \geq 1\}$ únicamente precisamos incorporar la gramática independiente del contexto definida en el ejemplo 2.10. Como resultado, obtendremos una LIG $\mathcal{L}' = (V'_N, V'_T, V'_I, P', S')$, donde $V'_N = \{S', S, A, C, D\}$, $V_T = \{a, b, c, d, e\}$, S' es el axioma de la nueva gramática y P' contiene las producciones

$$S'[\circ\circ] \to A[\,]\; S[\circ\circ]$$
$$S[\circ\circ] \to c\; C[\circ\circ\gamma]\; e$$
$$C[\circ\circ] \to c\; C[\circ\circ\gamma]\; e$$
$$C[\circ\circ] \to D[\circ\circ]$$
$$D[\circ\circ\gamma] \to d\; D[\circ\circ]$$
$$D[\,] \to \epsilon$$
$$A[\,] \to a\; A[\,]\; b$$
$$A[\,] \to \epsilon$$

¶

2.4.3 Conversión de TAG a LIG

Las gramáticas lineales de índices generan la misma clase de lenguajes que las gramáticas de adjunciones de árboles. Dada una cadena perteneciente al lenguaje generado por una gramática de adjunción de árboles, existe una gramática lineal de índices que construye la misma estructura derivada sobre dicha cadena.

A continuación mostramos un método para transformar una gramática de adjunción de árboles en una gramática lineal de índices fuertemente equivalente que está basado en los descritos por Vijay-Shanker y Weir en [213, 214]. La base del método consiste en ir descomponiendo los árboles elementales en un conjunto de producciones LIG teniendo cuidado de hacer corresponder la espina de los árboles auxiliares con la relación padre-hijo dependiente de dichas producciones. Adicionalmente utilizaremos los superíndices t (*top*) y b (*bottom*) para indicar si nos estamos refiriendo a un nodo η de un árbol elemental antes de que se haya realizado ninguna adjunción, en cuyo caso denotaremos dicho nodo como η^t, o después de la realización de una adjunción, en cuyo caso nos referiremos a dicho nodo como η^b. Respecto a la forma de los árboles elementales, únicamente asumiremos que los nodos etiquetados por símbolos terminales no tienen hermanos. En caso de que los tengan, reemplazaremos el nodo etiquetado por el terminal por un nuevo nodo interior del que cuelga el nodo etiquetado por el terminal.

Una vez aplicada la transformación, de una gramática de adjunción de árboles $(V_T, V_N, \boldsymbol{I}, \boldsymbol{A}, S)$ obtendremos una gramática lineal de índices (V_T, V_N', V_I, S', P), donde $V_N' = \{\eta^t\} \cup \{\eta^b\}$ y $V_I = \{\eta\}$ tal que η es un nodo de un árbol elemental. El conjunto de producciones P se determina aplicando las siguientes reglas de transformación a cada uno de los nodos de los árboles elementales de la gramática de adjunción de árboles:

1. Sea η un nodo que es padre de un único nodo etiquetado por $a \in V_T \cup \{\epsilon\}$. En este caso crearemos la producción
$$\eta^b[\,] \rightarrow a$$

2. Sea η_0 un nodo de la espina de un árbol auxiliar con hijos $\eta_1 \ldots \eta_d \ldots \eta_m$, de los cuales η_d está también en la espina. En tal caso crearemos la producción
$$\eta_0^b[\circ\circ] \rightarrow \eta_1^t[\,] \ldots \eta_d^t[\circ\circ] \ldots \eta_m^t$$

3. Sea η_0 un nodo que no está en la espina de un árbol auxiliar y sean $\eta_1 \ldots \eta_m$ sus hijos. En tal caso crearemos la producción
$$\eta_0^b[\,] \rightarrow \eta_1^t[\,] \ldots \eta_m^t[\,]$$
En el caso de no desear incluir producciones de este tipo podemos suponer arbitrariamente que el primer hijo es el hijo dependiente, resultando entonces en una producción
$$\eta_0^b[\circ\circ] \rightarrow \eta_1^t[\circ\circ]\, \eta_2^t[\,] \ldots \eta_m^t[\,]$$

4. Sea η un nodo en el que no es obligatorio realizar ninguna adjunción. En ese caso añadiremos la producción
$$\eta^t[\circ\circ] \rightarrow \eta^b[\circ\circ]$$

5. Sea η un nodo en el que se puede realizar la adjunción del árbol auxiliar β. En este caso añadiremos la producción
$$\eta^t[\circ\circ] \rightarrow \eta_r^t[\circ\circ\eta]$$
donde η_r es el nodo raíz de β.

6. Sea η_f el nodo pie de un árbol auxiliar β que puede ser adjuntado en un nodo η de un árbol elemental. En tal caso, para cada uno de los posibles nodos de adjunción η añadiremos la producción
$$\eta_f^b[\circ\circ\eta] \to \eta^b[\circ\circ]$$

7. En el caso de una gramática de adjunción de árboles lexicalizada, un nodo η puede ser un nodo de sustitución. En tal caso, para todos los árboles iniciales α que pueden ser sustituidos en dicho nodo crearemos la producción
$$\eta^t[\circ\circ] \to \eta_r^t[\circ\circ]$$
donde η_r es el nodo raíz de α.

8. Para cada nodo η_r que sea raíz de un árbol inicial y que esté etiquetado por S, crearemos una producción
$$S'[\circ\circ] \to \eta_r^t[\circ\circ]$$

Ejemplo 2.12 Consideremos la gramática de adjunción de árboles de la figura 2.3. Para referirnos al nodo de un árbol γ que ocupa la posición g según el direccionamiento de Gorn utilizaremos la notación $\langle\gamma, g\rangle$. La gramática lineal de índices obtenida a partir de dicha TAG contiene las siguientes producciones correspondientes

- al árbol α:
$$S'[\circ\circ] \to \langle\alpha, 0\rangle^t[\circ\circ]$$
$$\langle\alpha, 0\rangle^t[\circ\circ] \to \langle\alpha, 0\rangle^b[\circ\circ]$$
$$\langle\alpha, 0\rangle^b[\,] \to \langle\alpha, 1\rangle^t[\,] \;\; \langle\alpha, 2\rangle^t[\,]$$
$$\langle\alpha, 1\rangle^t[\circ\circ] \to \langle\alpha, 1\rangle^b[\circ\circ]$$
$$\langle\alpha, 1\rangle^b[\,] \to a$$
$$\langle\alpha, 2\rangle^t[\circ\circ] \to \langle\beta_1, 0\rangle^t[\circ\circ\langle\alpha, 2\rangle]$$
$$\langle\alpha, 2\rangle^t[\circ\circ] \to \langle\beta_2, 0\rangle^t[\circ\circ\langle\alpha, 2\rangle]$$
$$\langle\alpha, 2\rangle^b[\circ\circ] \to \langle\alpha, 2.1\rangle^t[\circ\circ]$$
$$\langle\alpha, 2.1\rangle^t[\circ\circ] \to \langle\alpha, 2.1\rangle^b[\circ\circ]$$
$$\langle\alpha, 2.1\rangle^b[\,] \to c$$

- al árbol β_1:
$$\langle\beta_1, 0\rangle^t[\circ\circ] \to \langle\beta_1, 0\rangle^b[\circ\circ]$$
$$\langle\beta_1, 0\rangle^b[\circ\circ] \to \langle\beta_1, 1\rangle^t[\,] \;\; \langle\beta_1, 2\rangle^t[\circ\circ]$$
$$\langle\beta_1, 1\rangle^t[\circ\circ] \to \langle\beta_1, 1\rangle^b[\circ\circ]$$
$$\langle\beta_1, 1\rangle^b[\,] \to a$$
$$\langle\beta_1, 2\rangle^t[\circ\circ] \to \langle\beta_1, 0\rangle^t[\circ\circ\langle\beta_1, 2\rangle]$$
$$\langle\beta_1, 2\rangle^t[\circ\circ] \to \langle\beta_2, 0\rangle^t[\circ\circ\langle\beta_1, 2\rangle]$$
$$\langle\beta_1, 2\rangle^b[\circ\circ] \to \langle\beta_1, 2.1\rangle^t[\circ\circ] \;\; \langle\beta_1, 2.2\rangle^t[\,]$$

$$\langle \beta_1, 2.1\rangle^t[\circ\circ] \to \langle \beta_1, 2.1\rangle^b[\circ\circ]$$

$$\langle \beta_1, 2.1\rangle^b[\circ\circ\langle \alpha, 2\rangle] \to \langle \alpha, 2\rangle^b[\circ\circ]$$

$$\langle \beta_1, 2.1\rangle^b[\circ\circ\langle \beta_1, 2\rangle] \to \langle \beta_1, 2\rangle^b[\circ\circ]$$

$$\langle \beta_1, 2.2\rangle^t[\circ\circ] \to \langle \beta_1, 2.2\rangle^b[\circ\circ]$$

$$\langle \beta_1, 2.2\rangle^b[\,] \to c$$

- al árbol β_2:

$$\langle \beta_2, 0\rangle^t[\circ\circ] \to \langle \beta_2, 0\rangle^b[\circ\circ]$$

$$\langle \beta_2, 0\rangle^b[\circ\circ] \to \langle \beta_2, 1\rangle^t[\circ\circ]$$

$$\langle \beta_2, 1\rangle^t[\circ\circ] \to \langle \beta_3, 0\rangle^t[\circ\circ\langle \beta_2, 1\rangle]$$

$$\langle \beta_2, 1\rangle^b[\circ\circ] \to \langle \beta_2, 1.1\rangle^t[\circ\circ]$$

$$\langle \beta_2, 1.1\rangle^t[\circ\circ] \to \langle \beta_2, 1.1\rangle^b[\circ\circ]$$

$$\langle \beta_2, 1.1\rangle^b[\circ\circ\langle \alpha, 2\rangle] \to \langle \alpha, 2\rangle^b[\circ\circ]$$

$$\langle \beta_2, 1.1\rangle^b[\circ\circ\langle \beta_1, 2\rangle] \to \langle \beta_1, 2\rangle^b[\circ\circ]$$

- y al árbol β_3:

$$\langle \beta_3, 0\rangle^t[\circ\circ] \to \langle \beta_3, 0\rangle^b[\circ\circ]$$

$$\langle \beta_3, 0\rangle^b[\circ\circ] \to \langle \beta_3, 1\rangle^t[\,] \quad \langle \beta_3, 2\rangle^t[\circ\circ]$$

$$\langle \beta_3, 1\rangle^t[\circ\circ] \to \langle \beta_3, 1\rangle^b[\circ\circ]$$

$$\langle \beta_3, 1\rangle^b[\,] \to b$$

$$\langle \beta_3, 2\rangle^t[\circ\circ] \to \langle \beta_3, 0\rangle^t[\circ\circ\langle \beta_3, 2\rangle]$$

$$\langle \beta_3, 2\rangle^t[\circ\circ] \to \langle \beta_3, 2\rangle^b[\circ\circ]$$

$$\langle \beta_3, 2\rangle^b[\circ\circ] \to \langle \beta_3, 2.1\rangle^t[\circ\circ] \quad \langle \beta_3, 2.2\rangle^t[\,]$$

$$\langle \beta_3, 2.1\rangle^t[\circ\circ] \to \langle \beta_3, 2.1\rangle^b[\circ\circ]$$

$$\langle \beta_3, 2.1\rangle^b[\circ\circ\langle \beta_3, 2\rangle] \to \langle \beta_3, 2\rangle^b[\circ\circ]$$

$$\langle \beta_3, 2.2\rangle^t[\circ\circ] \to \langle \beta_3, 2.2\rangle^b[\circ\circ]$$

$$\langle \beta_3, 2.2\rangle^b[\,] \to d$$

¶

Puesto que las gramáticas lineales de índices presentan una forma más cercana a las gramáticas independientes del contexto que las gramáticas de adjunción de árboles, en ciertos casos es más fácil adaptar técnicas existentes para el tratamiento de gramáticas independientes del contexto al caso de LIG que al de TAG. Es por ello que aparte del interés que las gramáticas lineales de índices presentan por sí mismas como formalismo descriptivo, en lingüística computacional presentan una relevancia especial puesto que son habitualmente utilizadas como formalismo intermedio a través del cual se realiza el análisis sintáctico de las gramáticas de adjunción de árboles [213, 170, 175, 214].

2.5 Formalismos derivados de LIG

2.5.1 LIG estocásticas

Schabes propone en [170] anotar con probabilidades las producciones de las gramáticas lineales de índices, con objeto de definir un formalismo gramatical que facilite el tratamiento de las gramáticas de adjunción de árboles estocásticas. El formalismo resultante recibe el nombre de LIG estocásticas (*Stochastic Linear Indexed Grammars*, SLIG). La función de distribución de probabilidades debe satisfacer que la suma de las probabilidades de todas las producciones que puedan ser aplicadas a un no-terminal con un determinado índice en la cima de su pila sea igual a 1.

La probabilidad de una derivación se define como el producto de las probabilidades de todas las producciones individuales involucradas en dicha derivación (teniendo en cuenta las repeticiones). La probabilidad de una cadena de entrada es la suma de las probabilidades de todas las derivaciones de dicha cadena. La gramática es consistente si las probabilidades asociadas a todas las cadenas del lenguaje suman 1, aunque Schabes no investiga las condiciones bajo las cuales se satisface dicha condición de consistencia.

Nederhof et al. proponen en [129] un método para calcular la probabilidades de los prefijos de las cadenas de un lenguaje a partir de una LIG estocástica, equivalente al método propuesto por los mismos autores para el cálculo de las probabilidades de los prefijos de las cadenas de un lenguaje a partir de una TAG estocástica [128].

2.5.2 Gramáticas parcialmente lineales de índices

Keller y Weir proponen en [98] relajar la condición de linealidad en la transmisión de la pila de índices permitiendo que dos no-terminales del lado derecho de una misma producción compartan la pila de índices, pero manteniendo la restricción de que la pila del no-terminal del lado izquierdo de una producción sólo puede ser compartida con un no-terminal del lado derecho de dicha producción.

El formalismo resultante recibe el nombre de gramáticas parcialmente lineales de índices (*Partially Linear Indexed Grammars*, PLIG). La capacidad generativa se amplía considerablemente con respecto a LIG puesto que las PLIG pueden generar el lenguaje de k-copias $\{w^k \mid w \in R, k \geq 0\}$, donde R es un lenguaje regular, y pueden *contar* hasta cualquier k, por lo que pueden generar lenguajes de la forma $\{a_1^n \ldots a_k^n \mid k \geq 1, n \geq 0\}$.

Puesto que las pilas compartidas entre no-terminales hermanos no puede ser compartidas por el no-terminal del lado izquierdo de la producción, el número de espinas dependientes en una derivación está acotado por la longitud del lado derecho de las producciones.

2.5.3 Gramáticas parcialmente lineales de árboles

Keller y Weir también proponen en [98] aumentar la capacidad generativa de las gramáticas parcialmente lineales de índices reemplazando las pilas por árboles. Se establece la restricción de que cada subárbol del no-terminal del lado izquierdo de una producción puede ser compartido con, a lo sumo, un no-terminal del lado derecho. No obstante, aquellos subárboles que no son compartidos con el lado izquierdo de una producción pueden ser compartidos entre los no-terminales del lado derecho de dicha producción.

El formalismo tal cual ha sido descrito permite simular una máquina de Turing arbitraria. Para evitar este inconveniente se establece una nueva limitación: supongamos que el no-terminal en el lado izquierdo comparte un subárbol con un no-terminal del lado derecho de la producción; supongamos también que dicho no-terminal del lado derecho comparte algún subárbol con otro no-terminal; en tal caso los subárboles que el no-terminal del lado izquierdo comparta con esos dos no-terminales del lado derecho deben ser hermanos.

El formalismo resultante recibe el nombre de *gramáticas parcialmente lineales de árboles* (*Partially Linear Tree Grammars*, PLTG). La capacidad generativa se amplía considerablemente con respecto a PLIG puesto que las PLTG pueden generar el lenguaje de k-copias sobre cualquier lenguaje L independiente del contexto: $\{w^k \mid w \in L, k \geq 0\}$.

Las estructuras de rasgos acíclicas no reentrantes son árboles cuyas ramas están etiquetadas por nombres de rasgos y cuyas hojas pueden estar etiquetadas por valores atómicos mientras que los nodos interiores carecen de etiquetas. Basándonos en este hecho podemos considerar que las restricciones formuladas para PLTG son restricciones en un formalismo gramatical basado en unificación. Keller y Weir denominan a tal formalismo *PATR parcialmente lineal* (PLPATR).

2.5.4 LIG multiconjunto

Rambow argumenta en [153, 152] que las estructuras de rasgos que toman valores en multiconjuntos tienen relevancia lingüística pero no pueden ser descritas mediante gramáticas lineales de índices o formalismos equivalentes. Rambow propone un nuevo formalismo denominado LIG multiconjunto ($\{\}$-LIG) para solventar esta carencia. La clase de lenguajes definida por $\{\}$-LIG es incomparable con los lenguajes de adjunción de árboles, puesto que incluye lenguajes como $\{a^n b^n c^n d^n e^n \mid n \geq 0\}$, que no pueden ser generados por una gramática lineal de índices, mientras que no pueden generar el lenguaje copia $\{ww \mid w \in \{a,b\}^*\}$.

2.5.5 Gramáticas pila-lineales independientes del contexto

Wartena define en [228] una jerarquía de gramáticas que denomina gramáticas S-lineales independientes del contexto (CFL-S-G, *Context-Free Linear-S Grammars*). El primer nivel en esta jerarquía lo constituyen las gramáticas independientes del contexto. El segundo nivel lo constituyen las gramáticas pila-lineales independientes del contexto (CFL-pila-G), que no son más que gramáticas independientes del contexto en las que cada elemento gramatical tiene asociada una pila, por lo que son equivalentes a las gramáticas lineales de índices. Utilizando estructuras de almacenamiento más complejas (por ejemplo tuplas de pilas) se pueden definir formalismos gramaticales más potentes que las gramáticas lineales de índices.

2.6 Otros formalismos gramaticales que generan lenguajes de adjunción de árboles

2.6.1 Gramáticas categoriales combinatorias

La base de las gramáticas categoriales combinatorias (*Combinatory Categorial Grammars*, CCG) [194] son las *categorías*. El conjunto de categorías generado a partir de un conjunto V_N de categorías atómicas se define como el conjunto más pequeño tal que todos los miembros de V_N son categorías y si c_1 y c_2 son categorías entonces (c_1/c_2) y $(c_1\backslash c_2)$ también lo son.

Formalmente, una gramática categorial combinatoria es una quíntupla (V_T, V_N, S, f, R) donde:

- V_T es un conjunto finito de símbolos terminales (ítems léxicos).
- V_N es un conjunto finito de símbolos no-terminales (categorías atómicas).
- $S \in V_N$ es el axioma de la gramática.

- f es una función que relaciona cada elemento de V_T con un conjunto finito de categorías.
- R es un conjunto finito de reglas de cancelación. Los diferentes tipos de reglas son:
  - *Reglas hacia adelante* de la forma:

$$(x/y)\ (y|_1 z_1|_2 \dots |_m z_m) \rightarrow (x|_1 z_1|_2 \dots |_m z_m)$$

  - *Reglas hacia atrás* de la forma

$$(y|_1 z_1|_2 \dots |_m z_m)\ (x\backslash y)\ \rightarrow (x|_1 z_1|_2 \dots |_m z_m)$$

  donde $m \geq 0$, $x, y, z_1, \dots, z_m$ son metavariables de categorías y $|_1, \dots, |_m \in \{\backslash, /\}$. En dichas reglas (x/y) y $(x\backslash y)$ reciben el nombre de *constituyentes primarios* mientras que $(y|_1 z_1|_2 \dots |_m z_m)$ recibe el nombre de *constituyente secundario*. En el caso de $m = 0$ estas reglas corresponden a la aplicación de funciones y para $m > 0$ corresponden a la composición de funciones.

Se puede restringir la aplicación de las reglas especificando ciertas condiciones para la instanciación de las metavariables. Dichas condiciones pueden ser aplicables a la categoría completa o bien únicamente al primer componente.

La relación de derivación $\Rightarrow$ se define como sigue:

- Si $c_1 c_2 \rightarrow c$ es una instancia de una regla en R, entonces $\Upsilon_1 c \Upsilon_2 \Rightarrow \Upsilon_1 c_1 c_2 \Upsilon_2$, donde Υ_1 y Υ_2 son cadenas de categorías y símbolos terminales. El componente primario de la regla $c_1 c_2 \rightarrow c$ se denomina *hijo dependiente* de c con respecto a esta derivación.
- Si $c \in f(a)$ para algún $a \in V_T$ y c es una categoría entonces $\Upsilon_1 c \Upsilon_2 \Rightarrow \Upsilon_1 a \Upsilon_2$.

El lenguaje generado por una CCG queda definido por todos los $w \in V_T^*$ tal que $S \overset{*}{\Rightarrow} w$.

Ejemplo 2.13 Consideremos la gramática categorial combinatoria definida por la quíntupla (V_T, V_N, S, f, R), donde $V_T = \{a, b, c, d\}$, $V_N = \{S, A, B, C, D\}$, S es el axioma, f es como sigue:

$$\begin{aligned} f(a) &= \{S/(S/C), S/(B/C)\} \\ f(b) &= \{B/D\} \\ f(c) &= \{(B/C)\backslash B\} \\ f(d) &= \{D\backslash B\} \\ f(\epsilon) &= \{B\} \end{aligned}$$

y R no establece ninguna restricción. Esta gramática genera el lenguaje $\{a^n b^m c^n d^m \mid n, m \geq 1\}$, que coincide con el generado por la gramática de adjunción de árboles de la figura 2.3 y con el generado por la gramática lineal de índices del ejemplo 2.8. El árbol derivado, que coincide con el de derivación, para la cadena *aabbbccddd* se muestra en la figura 2.14. ¶

Weir muestra en [230] la inclusión de los lenguajes de adjunción de árboles en los lenguajes categoriales combinatorios y de éstos en los lenguajes lineales de índices. Joshi et al. muestran en [95] la equivalencia de los lenguajes lineales de índices y de los lenguajes categoriales combinatorios. Vijay-Shanker y Weir muestran en [216] la inclusión de los lenguajes categoriales combinatorios en los lenguajes lineales de índices y la de los lenguajes de adjunción de árboles en los lenguajes categoriales combinatorios.

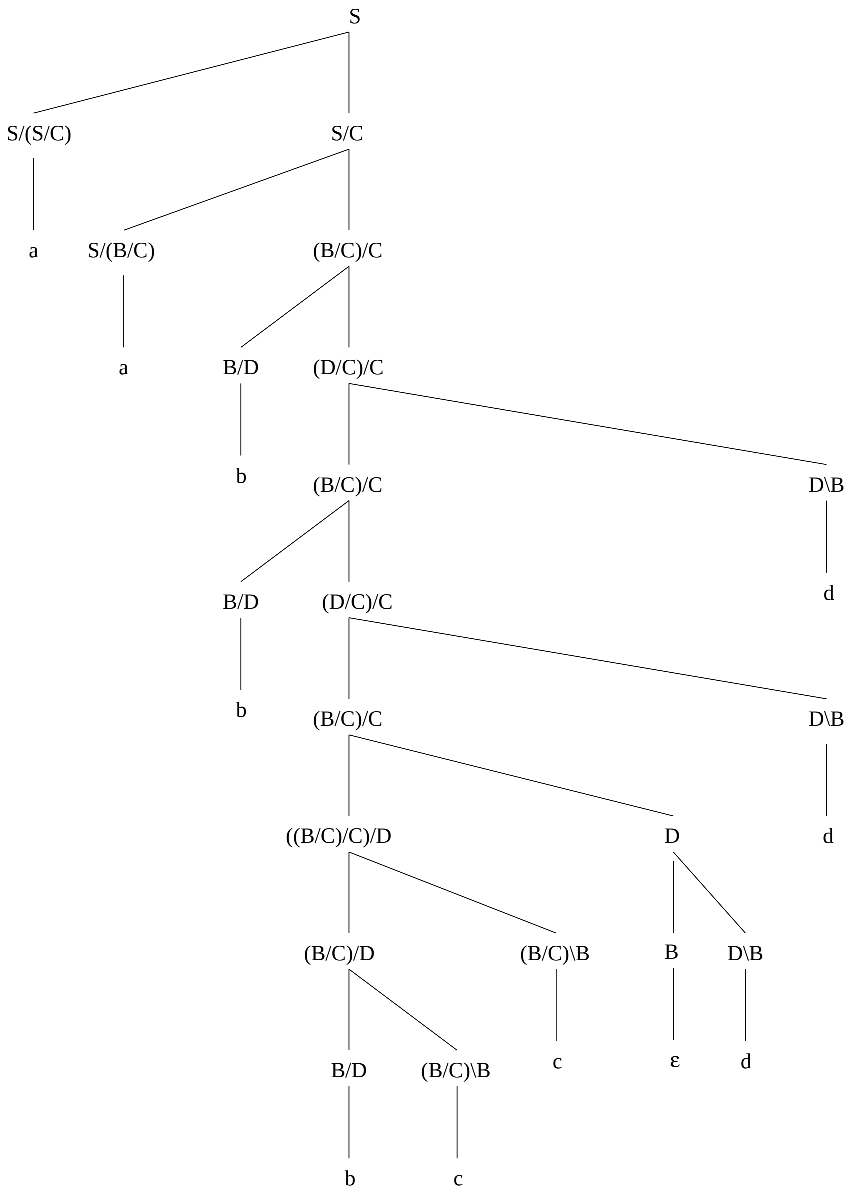

Figura 2.14: Árbol derivado en CCG para la cadena *aabbbccddd*

2.6.2 Gramáticas de núcleo

Las gramáticas de núcleo (*Head Grammars*, HG) [146, 159] pueden considerase como una generalización de las gramáticas independientes del contexto que, además de la operación de sustitución, utilizan una nueva operación denominada *wrapping*. Mientras que los no-terminales de las CFG derivan cadenas de terminales, los no-terminales de las gramáticas de núcleo derivan cadenas con núcleo. En la definición original de las gramáticas de núcleo se establecía que el núcleo de una cadena debía ser o bien el primer símbolo terminal o bien el último símbolo terminal. Siguiendo la notación de Vijay-Shanker y Weir en [216] consideraremos una definición equivalente según la cual una cadena con núcleo es un par de cadenas de terminales (u, v) que denotaremos $u_{\uparrow}v$. Con ello evitaremos que la presencia de cadenas vacías provoque problemas notacionales.

Formalmente, una gramática de núcleo es una cuádrupla (V_N, V_T, S, P) donde:

- V_N es un conjunto finito de símbolos no-terminales.
- V_T es un conjunto finito de símbolos terminales.
- $S \in V_N$ es el axioma de la gramática.
- P es un conjunto finito de producciones de la forma $A \to f(\sigma_1, \dots, \sigma_m)$, con $A \in V_N$, $m \geq 1$, $f \in \{W, C_{1,m}, C_{2,m}, \dots, C_{m,m}\}$, $\sigma_1, \dots, \sigma_m \in V_N \cup (V_T^* \times V_T^*)$ y si $f = W$ entonces $m = 2$.

  $C_{i,m} : (V_T^* \times V_T^*)^m \to (V_T^* \times V_T^*)$ es la operación de concatenación:

$$C_{i,m}(u_{1\uparrow}v_1, \dots, u_{i\uparrow}v_i, \dots, u_{m\uparrow}v_m) = u_1 v_1 \dots u_{i\uparrow}v_i \dots u_m v_m$$

  $W : (V_T^* \times V_T^*)^2 \to (V_T^* \times V_T^*)$ es la operación de wrapping:

$$W(u_{1\uparrow}v_1, u_{2\uparrow}v_2) = u_1 u_{2\uparrow}v_2 v_1$$

La relación de derivación $\Rightarrow$ se define como sigue:

- $u_\uparrow v \overset{0}{\Rightarrow} u_\uparrow v$ para todo $u_\uparrow v \in V_T^* \times V_T^*$.
- Si $A \to f(\sigma_1, \dots, \sigma_m) \in P$ entonces $A \overset{k}{\Rightarrow} f(u_{1\uparrow}v_1, \dots, u_{m\uparrow}v_m)$, donde $\sigma_i \overset{k_i}{\Rightarrow} u_{i\uparrow}v_i$, con $1 \leq i \leq m$ y $k = 1 + \sum_{1 \leq i \leq m} k_i$.

El lenguaje generado por una gramática de núcleo queda definido por todos los $uv \in V_T^*$ tales que $S \overset{*}{\Rightarrow} u_\uparrow v$, donde $\overset{*}{\Rightarrow} = \cup_{k \geq 0} \overset{k}{\Rightarrow}$.

Ejemplo 2.14 Consideremos la gramática de núcleo definida por la cuádrupla V_N, V_T, S, P, donde $V_T = \{a, b, c, d\}$, $V_N = \{S, A, B, D\}$, S es el axioma y P es el conjunto de producciones:

$$S \to W(A, c)$$

$$A \to C_{2,2}(a_\uparrow \epsilon, S)$$

$$A \to C_{2,2}(a_\uparrow \epsilon, B)$$

$$B \to C_{2,2}(b_\uparrow \epsilon, D)$$

$$D \to C_{1,2}(B, d_\uparrow \epsilon)$$

$$B \to C_{1,2}(b_\uparrow \epsilon, d_\uparrow \epsilon)$$

Esta gramática genera el lenguaje $\{a^n b^m c^n d^m\}$ con $n, m \geq 1$, que coincide con el generado por la gramática de adjunción de árboles de la figura 2.3, por la gramática lineal de índices del ejemplo 2.8 y por la gramática categorial combinatoria del ejemplo 2.13. Las gramáticas de núcleo no proporcionan una estructura derivada a las cadenas del lenguaje. Sin embargo, podemos construir un árbol de derivación[230] en el que cada nodo interno es anotado por un no-terminal y la operación utilizada para combinar los pares de cadenas con núcleo que son derivados por los nodos hijo. En la figura 2.15 se muestra el árbol de derivación para la cadena *aabbbccddd*. En dicha derivación primero se concatena el mismo número de c y d y después, por cada a que se concatena se inserta una c entre b^m y d^m mediante la operación de wrapping:

$$
\begin{array}{rl}
 & B \stackrel{1}{\Rightarrow} C_{1,2}(b_\uparrow \epsilon, d_\uparrow \epsilon) \ = \ b_\uparrow d \\
\text{de aquí} & D \stackrel{2}{\Rightarrow} C_{1,2}(b_\uparrow d, d_\uparrow \epsilon) \ = \ b_\uparrow dd \\
\text{de aquí} & B \stackrel{3}{\Rightarrow} C_{2,2}(b_\uparrow \epsilon, b_\uparrow dd) \ = \ bb_\uparrow dd \\
\text{de aquí} & D \stackrel{4}{\Rightarrow} C_{1,2}(bb_\uparrow dd, d_\uparrow \epsilon) \ = \ bb_\uparrow ddd \\
\text{de aquí} & B \stackrel{5}{\Rightarrow} C_{2,2}(b_\uparrow \epsilon, bb_\uparrow ddd) \ = \ bbb_\uparrow ddd \\
\text{de aquí} & A \stackrel{6}{\Rightarrow} C_{2,2}(a_\uparrow \epsilon, bbb_\uparrow ddd) \ = \ abbb_\uparrow ddd \\
\text{de aquí} & S \stackrel{7}{\Rightarrow} W(abbb_\uparrow ddd, c_\uparrow \epsilon) \ = \ abbbc_\uparrow ddd \\
\text{de aquí} & A \stackrel{8}{\Rightarrow} C_{2,2}(a_\uparrow \epsilon, abbbc_\uparrow ddd) \ = \ aabbbc_\uparrow ddd \\
\text{de aquí} & S \stackrel{9}{\Rightarrow} W(aabbbc_\uparrow ddd, c_\uparrow \epsilon) \ = \ aabbbcc_\uparrow ddd
\end{array}
$$

¶

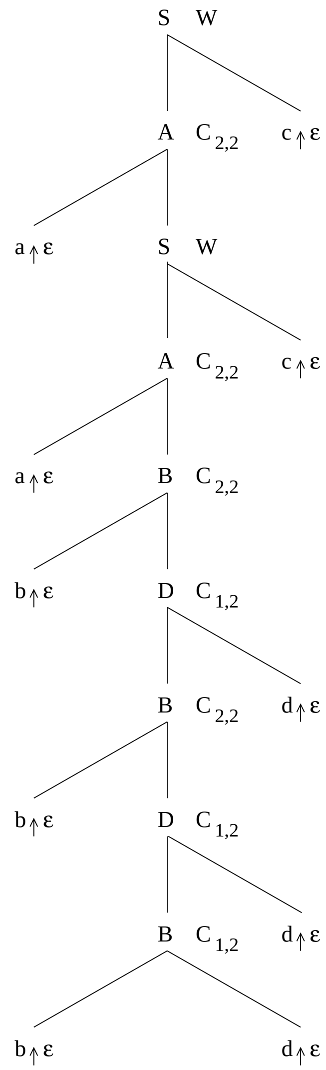

Figura 2.15: Árbol de derivación en HG para la cadena *aabbbccddd*

Vijay-Shanker muestra en [206] la equivalencia entre los lenguajes de adjunción de árboles y los lenguajes de núcleo y la equivalencia entre estos últimos y los lenguajes lineales de índices. Joshi et al. muestran en [95] la equivalencia de los lenguajes de adjunción de árboles y de los lenguajes de núcleo. Vijay-Shanker y Weir muestran en [216] la inclusión de los lenguajes lineales de índices en los lenguajes de núcleo y la de estos en los lenguajes de adjunción de árboles.

2.6.3 Sistemas de matrices recurrentes independientes del contexto de índice 2

Una *matriz recurrente* [81] es una matriz finita cuyos elementos son símbolos terminales o matrices recurrentes. Una *interpretación* indica la forma en la que se obtiene la cadena derivada por una matriz recurrente mediante su lectura línea a línea, bien de izquierda a derecha o bien de derecha a izquierda, descendiendo recursivamente en aquellos elementos que son a su vez matrices recurrentes.

Becker y Heckmann definen en [26] los sistemas de matrices recurrentes independientes del contexto (*context-free Recursive Matrix System*, cf-RMS) de índice 2 y muestran que son débilmente equivalentes a las gramáticas de adjunción de árboles. Un cf-RMS es un par (G, I) donde G es una gramática independiente del contexto que genera matrices recurrentes e I es una interpretación que permite leer una cadena a partir de una matriz recurrente. $L(G)$ es el conjunto de todas las matrices recurrentes derivadas por la gramática G. $L(G, I)$ es el conjunto de todas las cadenas derivadas a partir de las matrices recurrentes en $L(G)$ mediante la interpretación I. La gramática G es una tupla (V_N, V_{ec}, P, S) donde el conjunto V_{ec} de terminales contiene vectores de tamaño n, un parámetro del sistema de matrices recurrentes independiente del contexto denominado *índice* que especifica el número de filas en todas las matrices y submatrices. Los vectores contienen elementos en $V_T \cup V_N$, donde V_T es el conjunto finito de terminales del sistema de matrices recurrentes.

La relación de derivación $\Rightarrow$ se define en relación a G sobre *matrices recurrentes extendidas*, concatenaciones de vectores y no-terminales donde los elementos de un vector son bien terminales de V_T, bien no-terminales de V_N, o bien matrices recurrentes extendidas. Cada paso de una derivación reescribe exactamente un no-terminal de acuerdo con una regla en P. Denotaremos mediante $\overset{*}{\Rightarrow}$ el cierre reflexivo y transitivo de $\Rightarrow$. El lenguaje $L(G)$ generado por G es el conjunto de matrices recurrentes r tal que $S \overset{*}{\Rightarrow} r$.

La interpretación de una matriz recurrente se deriva a partir de un vector de direcciones para cada fila de la matriz. Cada uno de los elementos de dicho vector indica si la correspondiente fila se lee de derecha a izquierda o de izquierda a derecha.

Becker y Heckmann presentan en [27] algoritmos ascendentes para el análisis sintáctico de diversos tipos de sistemas de matrices recurrentes.

Ejemplo 2.15 Consideremos el siguiente sistema de matrices recurrentes independiente del contexto de índice 2, que genera el lenguaje $\{a^n b^m c^n d^m\}$ con $n, m \geq 1$ y está definido por el par (G, I), donde G es la gramática independiente del contexto (V_N, V_{ec}, P, S) con $V_N = \{S, A, B\}$, $V_{ec} = \{a, b, c, d\}$, S el axioma de G y P el conjunto de producciones que se muestra en la figura 2.16. La interpretación del sistema de matrices recurrentes viene determinada por el vector

$$I = \begin{bmatrix} \rightarrow \\ \rightarrow \end{bmatrix}$$

El reconocimiento de la cadena *aabbbccddd* involucra la derivación en G que se muestra en la figura 2.17. Aplicando la interpretación I a la última matriz de dicha derivación obtenemos la cadena *aabbbccddd*. ¶

$$S \to \begin{bmatrix} a \\ c \end{bmatrix} A \begin{bmatrix} b \\ d \end{bmatrix} B$$

$$A \to \begin{bmatrix} a \\ c \end{bmatrix} A$$

$$B \to \begin{bmatrix} b \\ d \end{bmatrix} B$$

$$A \to \begin{bmatrix} \epsilon \\ \epsilon \end{bmatrix}$$

$$B \to \begin{bmatrix} \epsilon \\ \epsilon \end{bmatrix}$$

Figura 2.16: cf-RMS que genera el lenguaje $\{a^n b^m c^n d^m\}$

$$\begin{array}{|c|}\hline \\ S \\ \hline\end{array} \Rightarrow \begin{array}{|c|c|c|c|}\hline a & & b & \\ \hline c & A & b & B \\ \hline\end{array} \Rightarrow \begin{array}{|c|c|c|c|c|}\hline a & a & & b & \\ \hline c & c & A & d & B \\ \hline\end{array} \Rightarrow \begin{array}{|c|c|c|c|c|}\hline a & a & \epsilon & b & \\ \hline a & c & \epsilon & d & B \\ \hline\end{array}$$

$$\Rightarrow \begin{array}{|c|c|c|c|c|c|}\hline a & a & \epsilon & b & b & \\ \hline c & c & \epsilon & d & d & B \\ \hline\end{array} \Rightarrow \begin{array}{|c|c|c|c|c|c|c|}\hline a & a & \epsilon & b & b & b & \\ \hline c & c & \epsilon & d & d & d & B \\ \hline\end{array} \Rightarrow \begin{array}{|c|c|c|c|c|c|c|}\hline a & a & \epsilon & b & b & b & \epsilon \\ \hline c & c & \epsilon & d & d & d & \epsilon \\ \hline\end{array}$$

Figura 2.17: Derivación de la cadena *aabbbccddd* en cf-RMS

2.6.4 Gramáticas de concatenación de rangos positivas simples de aridad 2

Boullier define en [37] las gramáticas de concatenación de rangos positivas (*Positive Range Concatenation Grammars*, PRCG) como una quíntupla (V_N, V_T, V, P, S) donde V_N es un conjunto finito de predicados, V_T es un conjunto finito de terminales, V es un conjunto finito de variables, $S \in V_N$ es el axioma de la gramática y P es un conjunto finito de cláusulas de la forma

$$\psi_0 \to \psi_1 \ldots \psi_m$$

donde $m \geq 0$ y cada ψ_i es un predicado de la forma

$$A(\alpha_1, \ldots, \alpha_p)$$

donde $p \geq 1$ es la aridad, $A \in V_N$ y cada $\alpha_i \in (V_T \cup V_N)^*, 1 \leq i \leq p$ es un argumento. La aridad k de una gramática es el máximo de la aridad de sus predicados. Obtenemos de este modo diferentes subclases de PRCG denominadas k-PRCG en función de la aridad de la gramática.

Una cláusula, y por extensión una PRCG, puede ser:

- *no combinatoria* si cada uno de los argumentos que aparecen en su lado derecho está constituido por una única variable.
- *lineal* si toda variable aparece a lo sumo una vez en el lado derecho y una vez en el lado izquierdo.

- *no-borradora* si toda variable que aparece en el lado izquierdo también aparece en el lado derecho y viceversa.
- *propia* si es no-combinatoria y no-borradora.
- *simple* si es propia y lineal.

El lenguaje definido por una PRCG se basa en la noción de *rango*. Para una cadena de entrada $w = a_1 \dots a_n$ un rango es un par $(i, j), 0 \leq i \leq j \leq n$ de enteros que denotan la ocurrencia de alguna subcadena $a_{i+1} \dots a_j$ en w. El número i es el límite inferior, j es el límite superior y $j - i$ es el tamaño del rango. Una forma alternativa de denotar un rango (i, j) es $w_1 \bullet w_2 \bullet w_3$, donde $w_2 = a_{i+1} \dots a_j$. Se pueden crear nuevos rangos mediante la concatenación de varios rangos consecutivos.

En una PRCG, los terminales, variables y argumentos de las cláusulas están ligados a rangos mediante un mecanismo de sustitución. Una cláusula instanciada es una cláusula en la cual las variables y los argumentos son reemplazados consistentemente (con respecto a la operación de concatenación) por rangos. Los componentes de una cláusula instanciada son predicados instanciados.

La relación de derivación $\Rightarrow$ se define sobre cadenas de predicados instanciados: si un predicado instanciado aparece en el lado izquierdo de una cláusula instanciada entonces puede ser reemplazado por el lado derecho de dicha cláusula. Denotaremos mediante $\stackrel{+}{\Rightarrow}$ el cierre transitivo de $\Rightarrow$.

El lenguaje generado por una PRCG es el conjunto de cadenas $w \in V_T^*$ tal que $S(\bullet w \bullet) \stackrel{+}{\Rightarrow} \epsilon$.

Los argumentos de un predicado pueden denotar rangos continuos, discontinuos o solapados. Básicamente un predicado define una propiedad, estructura o dependencia entre sus argumentos, cuyos rangos pueden abarcar partes arbitrarias de la cadena de entrada. Este hecho hace que las PRCG estén bien adaptadas para describir dependencias de larga distancia.

Una gramática de concatenación de rangos negativa (*Negative Range Concatenation Grammar*, NRCG) es una PRCG salvo que algunos predicados que ocurren en la parte derecha de algunas cláusulas pueden ser predicados negativos de la forma $\neg A(\alpha_1, \dots, \alpha_m)$. Dichos predicados definen el lenguaje complementario del definido por $A(\alpha_1, \dots, \alpha_m)$, esto es, $\neg A(\alpha_1, \dots, \alpha_m) \Rightarrow \epsilon$ pertenece a la derivación de una cadena si y sólo si $\neg\exists\, A(\alpha_1, \dots, \alpha_m) \stackrel{+}{\Rightarrow} \epsilon$ para dicha cadena.

Una gramática de concatenación de rangos (*Range Concatenation Grammar*, RCG) es una PRCG o una NRCG. Las RCG son analizables en tiempo polinomial, son cerradas bajo las operaciones de unión, concatenación, cierre de Kleene, intersección y complementación. Estas propiedades permiten tratar fenómenos complejos como el *scrambling* [28] en tiempo lineal [35]. Como inconvenientes principales tenemos que el problema de determinar si el lenguaje definido por una RCG es vacío no es decidible y que las RCG no son cerradas bajo homomorfismo.

Las RCG son incomparables con la jerarquía de Chomky, aunque la clase de lenguajes generados está propiamente incluida en la clase de los lenguajes dependientes del contexto. Sin embargo, existen subclases de RCG que son equivalentes a formalismos gramaticales pertenecientes a dicha jerarquía. Así, las gramáticas independientes del contexto son equivalentes a las gramáticas de concatenación de rangos positivas simples de aridad 1 (1-sPRCG) [36] y las gramáticas de adjunción de árboles son equivalentes a las gramáticas de concatenación de rangos positivas simples de aridad 2 (2-sPRCG) [33].

Ejemplo 2.16 La siguiente gramática de concatenación de rangos positiva simple de aridad 2 genera el lenguaje $\{a^n b^m c^n d^m\}$ con $n, m \geq 1$ y está definida por la quíntupla (V_N, V_T, V, P, S), donde $V_N = \{S, A, B\}$, $V_T = \{a, b, c, d\}$, $V = \{W, X, Y, Z\}$, S es el axioma de la gramática y P

es el conjunto de producciones:

$$S(aWbXcYdZ) \to A(W,Y)\ B(X,Z)$$

$$A(aW,cY) \to A(W,Y)$$

$$B(bX,dZ) \to B(X,Z)$$

$$A(\epsilon,\epsilon) \to \epsilon$$

$$B(\epsilon,\epsilon) \to \epsilon$$

La cadena *aabbbccddd* es reconocida mediante la siguiente derivación:

$$\begin{aligned}
S(\bullet aabbbccddd\bullet) &\Rightarrow A(a\bullet a\bullet bbbccddd, aabbbc\bullet c\bullet ddd)\ B(aab\bullet bb\bullet ddd, aabbbccd\bullet dd\bullet) \\
&\Rightarrow A(aa\bullet\bullet bbbccddd, aabbbcc\bullet\bullet ddd)\ B(aab\bullet bb\bullet ddd, aabbbccd\bullet dd\bullet) \\
&\Rightarrow \epsilon\ B(aab\bullet bb\bullet ddd, aabbbccd\bullet dd\bullet) \\
&\Rightarrow B(aabb\bullet b\bullet ddd, aabbbccdd\bullet d\bullet) \\
&\Rightarrow B(aabbb\bullet\bullet ddd, aabbbccddd\bullet\bullet) \\
&\Rightarrow \epsilon
\end{aligned}$$

¶

2.6.5 Gramáticas independientes del contexto acopladas de rango 2

Las gramáticas independientes del contexto acopladas (*Coupled Context-Free Grammars*, CCFG) son una generalización de las gramáticas independientes del contexto en las que varios no-terminales son sustituidos simultáneamente si se corresponden con un sistema de paréntesis correctamente anidados.

Decimos que $\mathcal{K} = \{(k_i^1, \ldots, k_i^j, \ldots, k_i^{m_i}) \mid i, j, m_i \in \mathbb{N}\}$ es un conjunto de *tuplas de paréntesis* si es de tamaño finito y si satisface que $k_i^j \neq k_l^m$ cuando $i \neq l$ y cuando $j \neq m$. El conjunto de paréntesis se define como $comp(\mathcal{K}) = \{k_i \mid (k_1, \ldots, k_i, \ldots, k_r) \in \mathcal{K})\}$. Los conjuntos $\mathcal{K}[r] = \{(k_i^1, \ldots, k_i^j, \ldots, k_i^{m_i} \mid m_i = r)\}$ contienen las tuplas de paréntesis de longitud r. Asumiremos que $\mathcal{K}[0] = \{\epsilon\}$.

Sea $\mathcal{K}$ un conjunto de tuplas de paréntesis y sea T un conjunto arbitrario no vacío tal que $T \cap \mathcal{K} = T \cap comp(\mathcal{K}) = \emptyset$. El conjunto semi-Dyck extendido sobre $\mathcal{K}$ y T se denota por $ED(\mathcal{K}, T)$ y se define inductivamente como el conjunto más pequeño que satisface las siguientes condiciones:

- $T^* \subseteq ED(\mathcal{K}, T)$.
- $\mathcal{K}[1] \subseteq ED(\mathcal{K}, T)$.
- $u_1, \ldots, u_r \in ED(\mathcal{K}, T), (k_1, \ldots, k_{r+1}) \in \mathcal{K}[r+1] \Longrightarrow k_1 u_1 \ldots k_r u_r k_{r+1} \in ED(\mathcal{K}, T)$.
- $u, v \in ED(\mathcal{K}, T) \Longrightarrow uv \in ED(\mathcal{K}, T)$.

Un *sistema de reescritura de paréntesis sobre* $ED(\mathcal{K},T)$ es un conjunto finito y no vacío P de producciones de la forma $(k_1,\ldots,k_r) \rightarrow (\alpha_1,\ldots,\alpha_r)$ tal que $(k_1,\ldots,k_r) \in \mathcal{K}$ y $\alpha_1,\ldots,\alpha_r \in ED(\mathcal{K},T)$.

Definimos una gramática independiente del contexto acoplada como una cuádrupla $(\mathcal{K},T,P,S)$ donde $\mathcal{K}$ es un conjunto de paréntesis y P es un sistema de reescritura de paréntesis sobre $ED(\mathcal{K},T)$ y $S \in \mathcal{K}[1]$. El término *acoplada* expresa el hecho de que en un paso dado varias producciones independientes del contexto son aplicadas en paralelo y que dicha aplicación es controlada por $\mathcal{K}$. En consecuencia, $\mathcal{K}$ puede verse como un conjunto de no-terminales acoplados que deben ser reescritos simultáneamente.

La relación de derivación $\Rightarrow$ en CCFG establece que $\Phi \Rightarrow \Psi$, donde $\Phi, \Psi \in (comp(\mathcal{K}) \cup T)^*$, si y sólo si existen $u_1, u_{r+1} \in V^*$, $u_2,\ldots,u_r \in ED(\mathcal{K},T)$ y $(k_1,\ldots,k_r) \rightarrow (\alpha_1,\ldots,\alpha_r) \in P$ tal que $\Phi = u_1 k_1 u_2 k_2 \ldots u_r k_r u_{r+1}$ y $\Psi = u_1 \alpha_1 u_2 \alpha_2 \ldots u_r \alpha_r u_{r+1}$.

El lenguaje definido por una gramática independiente del contexto acoplada es el conjunto de cadenas $w \in T^*$ tal que $S \stackrel{*}{\Rightarrow} w$.

El *rango* de una gramática es igual a $\max\{r \mid (k_1,\ldots,k_r) \in \mathcal{K}\}$ e indica el número de elementos que pueden ser reescritos simultáneamente. La capacidad generativa de las gramáticas independientes del contexto acopladas viene determinada por el rango. Las CCFG de rango 1 son equivalentes a las gramáticas independientes del contexto mientras que las CCFG de rango 2 son débilmente equivalentes a las gramáticas de adjunción de árboles [145].

Ejemplo 2.17 La siguiente gramática independiente del contexto acoplada de rango 2 genera el lenguaje $\{a^n b^m c^n d^m\}$ con $n, m \geq 1$ y está definida por la cuádrupla $(\mathcal{K},T,P,(S))$ donde $\mathcal{K} = \{S,(A,B),(C,D)\}$, $T = \{a,b,c,d\}$, S es el axioma de la gramática y P es el sistema de reescritura de paréntesis que contiene las producciones:

$$(S) \rightarrow (aAbBcCdD)$$

$$(A,C) \rightarrow (aA,cC)$$

$$(B,D) \rightarrow (bB,dD)$$

$$(A,C) \rightarrow (\epsilon,\epsilon)$$

$$(B,D) \rightarrow (\epsilon,\epsilon)$$

La cadena *aabbbccddd* es reconocida mediante la derivación:

$$\begin{aligned} S &\Rightarrow aAbBcCdD \\ &\Rightarrow aaAbBccCdD \\ &\Rightarrow aabBccdD \\ &\Rightarrow aabbBccddD \\ &\Rightarrow aabbbBccdddD \\ &\Rightarrow aabbbccddd \end{aligned}$$

¶

2.6.6 Gramáticas de dos niveles

Las gramáticas de dos niveles [218] son un tipo de gramáticas de derivaciones controladas [50] en las que una gramática independiente del contexto restringe las derivaciones de otra gramática independiente del contexto.

Una gramática de dos niveles se define mediante una *gramática distinguida etiquetada* (*Labeled Distinguished Grammar*, LDG) [230] G_1 y una gramática independiente del contexto G_2. Una gramática distinguida etiquetada es una gramática independiente del contexto con dos especificaciones adicionales: las producciones están etiquetadas y uno de los símbolos del lado derecho de cada producción tiene una marca que lo distingue de los demás. El conjunto de etiquetas de G_1 es el conjunto de terminales de G_2. La gramática G_2 controla las derivaciones de G_1 de la siguiente manera: cada no-terminal de una forma sentencial en G_1 tiene asociada una palabra de control. La palabra de control es un prefijo de una sentencia generada por G_2. Al final de una derivación en G_1, cada símbolo terminal en la sentencia derivada es asociado con una palabra de control derivada por G_2 o bien por ϵ. De este modo, sólo aquellas derivaciones de G_1 que satisfacen las restricciones impuestas por las palabras de control son derivaciones válidas.

Formalmente, una LDG es un quíntupla (V_N, V_T, V_L, P, S) donde:

- V_N es un conjunto finito de no-terminales.
- V_T es un conjunto finito de terminales.
- V_L es un conjunto finito de etiquetas.
- $S \in V_N$ es el axioma de la gramática.
- P es un conjunto finito de producciones distinguidas de la forma

  $$l : A_0 \to A_1 A_2 \ldots \bar{A}_i \ldots A_m$$

  donde $l \in V_L$ es la etiqueta de la producción y $\bar{A}_i \in V_N \cup V_T \cup \{\epsilon\}$ es el símbolo distinguido de la producción.

Dada una gramática de dos niveles $G = (G_1, G_2)$, una forma sentencial derivada en G tiene la forma $\langle Y_1, w_1\rangle \ldots \langle Y_m, w_m\rangle$, donde los Y_i son terminales y no-terminales de G_1 y los w_i son palabras de control, prefijos de las cadenas generadas por G_2. Dada una forma sentencial $\alpha_1\langle A, w\rangle\alpha_2$, si $l : A \to A_1 A_2 \ldots \bar{A}_i \ldots A_m$ es una producción de G_1 entonces

$$\alpha_1\langle A, w\rangle\alpha_2 \Rightarrow \alpha_1\langle A_1, \epsilon\rangle \ldots \langle A_{i-1}, \epsilon\rangle\langle A_i, wl\rangle\langle A_{i+1}, \epsilon\rangle \ldots \langle A_m, \epsilon\rangle\alpha_2$$

El lenguaje generado por G es el conjunto de cadenas $a_1 \ldots a_n \in V_T^*$ tal que $\langle S, \epsilon\rangle \stackrel{*}{\Rightarrow} \langle a_1, w_1\rangle \ldots \langle a_n, w_n\rangle$ donde $w_i \in L(G_2) \cup \{\epsilon\}$.

Ejemplo 2.18 La siguiente gramática de dos niveles genera el lenguaje $\{a^n b^m c^n d^m\}$ con $n, m \geq 1$ y está definida por la gramática distinguida etiquetada $G_1 = (V_{N_1}, V_{T_1}, V_L, P_1, S_1)$ y la gramática independiente del contexto $G_2 = (V_{N_2}, V_{T_2}, P_2, S_2)$, donde $V_{N_1} = \{S_1, A, B, C, D\}$, $V_{T_1} = \{a, b, c, d\}$, $V_L = \{l_1, l_2, l_3, l_4, l_5, l_6, l_7, l_8, l_9\}$, S_1 es el axioma de G_1 y P_1 contiene el siguiente conjunto de producciones:

$$l_1 : S_1 \to \bar{A}$$

$$l_2 : A \to a\bar{A}$$

$$l_3 : A \to \bar{B}$$
$$l_4 : B \to b\bar{B}$$
$$l_5 : B \to \bar{D}$$
$$l_6 : D \to \bar{D}d$$
$$l_7 : D \to \bar{C}$$
$$l_8 : C \to \bar{C}c$$
$$l_9 : C \to \epsilon$$

Con respecto a G_2, $V_{N_2} = \{S_2, T, X\}$, $V_{T_2} = V_L$, S_2 es el axioma y P_2 contiene las siguientes producciones:

$$S_2 \to l_1\, T\, l_9$$
$$T \to l_2\, T\, l_8$$
$$T \to l_3\, X\, l_7$$
$$X \to l_4\, X\, l_6$$
$$X \to l_5$$

La cadena *aabbbccddd* es reconocida como resultado de la asiguiente derivación:

$$\begin{aligned}
\langle S_1, \epsilon\rangle &\Rightarrow \langle A, l_1\rangle \\
&\Rightarrow \langle a,\epsilon\rangle\langle A, l_1l_2\rangle \\
&\Rightarrow \langle a,\epsilon\rangle\langle a,\epsilon\rangle\langle A, l_1l_2l_2\rangle \\
&\Rightarrow \langle a,\epsilon\rangle\langle a,\epsilon\rangle\langle B, l_1l_2l_2l_3\rangle \\
&\Rightarrow \langle a,\epsilon\rangle\langle a,\epsilon\rangle\langle b,\epsilon\rangle\langle B, l_1l_2l_2l_3l_4\rangle \\
&\Rightarrow \langle a,\epsilon\rangle\langle a,\epsilon\rangle\langle b,\epsilon\rangle\langle b,\epsilon\rangle\langle B, l_1l_2l_2l_3l_4l_4\rangle \\
&\Rightarrow \langle a,\epsilon\rangle\langle a,\epsilon\rangle\langle b,\epsilon\rangle\langle b,\epsilon\rangle\langle b,\epsilon\rangle\langle B, l_1l_2l_2l_3l_4l_4l_4\rangle \\
&\Rightarrow \langle a,\epsilon\rangle\langle a,\epsilon\rangle\langle b,\epsilon\rangle\langle b,\epsilon\rangle\langle b,\epsilon\rangle\langle D, l_1l_2l_2l_3l_4l_4l_4l_5\rangle \\
&\Rightarrow \langle a,\epsilon\rangle\langle a,\epsilon\rangle\langle b,\epsilon\rangle\langle b,\epsilon\rangle\langle b,\epsilon\rangle\langle D, l_1l_2l_2l_3l_4l_4l_4l_5l_6\rangle\langle d,\epsilon\rangle \\
&\Rightarrow \langle a,\epsilon\rangle\langle a,\epsilon\rangle\langle b,\epsilon\rangle\langle b,\epsilon\rangle\langle b,\epsilon\rangle\langle D, l_1l_2l_2l_3l_4l_4l_4l_5l_6l_6\rangle\langle d,\epsilon\rangle\langle d,\epsilon\rangle \\
&\Rightarrow \langle a,\epsilon\rangle\langle a,\epsilon\rangle\langle b,\epsilon\rangle\langle b,\epsilon\rangle\langle b,\epsilon\rangle\langle D, l_1l_2l_2l_3l_4l_4l_4l_5l_6l_6l_6\rangle\langle d,\epsilon\rangle\langle d,\epsilon\rangle\langle d,\epsilon\rangle \\
&\Rightarrow \langle a,\epsilon\rangle\langle a,\epsilon\rangle\langle b,\epsilon\rangle\langle b,\epsilon\rangle\langle b,\epsilon\rangle\langle C, l_1l_2l_2l_3l_4l_4l_4l_5l_6l_6l_6l_7\rangle\langle d,\epsilon\rangle\langle d,\epsilon\rangle\langle d,\epsilon\rangle \\
&\Rightarrow \langle a,\epsilon\rangle\langle a,\epsilon\rangle\langle b,\epsilon\rangle\langle b,\epsilon\rangle\langle b,\epsilon\rangle\langle C, l_1l_2l_2l_3l_4l_4l_4l_5l_6l_6l_6l_7l_8\rangle\langle c,\epsilon\rangle\langle d,\epsilon\rangle\langle d,\epsilon\rangle\langle d,\epsilon\rangle \\
&\Rightarrow \langle a,\epsilon\rangle\langle a,\epsilon\rangle\langle b,\epsilon\rangle\langle b,\epsilon\rangle\langle b,\epsilon\rangle\langle C, l_1l_2l_2l_3l_4l_4l_4l_5l_6l_6l_6l_7l_8l_8\rangle\langle c,\epsilon\rangle\langle c,\epsilon\rangle\langle d,\epsilon\rangle\langle d,\epsilon\rangle\langle d,\epsilon\rangle \\
&\Rightarrow \langle a,\epsilon\rangle\langle a,\epsilon\rangle\langle b,\epsilon\rangle\langle b,\epsilon\rangle\langle b,\epsilon\rangle\langle C, l_1l_2l_2l_3l_4l_4l_4l_5l_6l_6l_6l_7l_8l_8\rangle\langle c,\epsilon\rangle\langle c,\epsilon\rangle\langle d,\epsilon\rangle\langle d,\epsilon\rangle\langle d,\epsilon\rangle \\
&\Rightarrow \langle a,\epsilon\rangle\langle a,\epsilon\rangle\langle b,\epsilon\rangle\langle b,\epsilon\rangle\langle b,\epsilon\rangle\langle \epsilon, l_1l_2l_2l_3l_4l_4l_4l_5l_6l_6l_6l_7l_8l_8l_9\rangle\langle c,\epsilon\rangle\langle c,\epsilon\rangle\langle d,\epsilon\rangle\langle d,\epsilon\rangle\langle d,\epsilon\rangle
\end{aligned}$$

¶

Kulkarni y Shankar investigan en [102] la extensión de las técnicas de análisis sintáctico determinista LL(1) y LR(1) al caso de las gramáticas de dos niveles.

2.7 Complejidad del análisis de los lenguajes de adjunción de árboles

2.7.1 Complejidad temporal

Durante mucho tiempo se ha discutido acerca del valor de la cota superior del tiempo mínimo requerido para realizar el análisis sintáctico de una gramática de adjunción de árboles. Satta estudia detalladamente esta cuestión en [166]. Los algoritmo de análisis presentados hasta entonces establecían una complejidad mínima de $\mathcal{O}(|G|\, n^6)$, donde $|G|$ es el tamaño de la gramática[9] y n es el tamaño de la cadena de entrada. Con el fin de determinar si es posible obtener una cota máxima más pequeña, Satta reduce el problema del análisis sintáctico de TAG a un problema algorítmico bien estudiado como es el de la *multiplicación de matrices booleanas* (MMB), de tal modo que la cota superior para el problema del análisis de TAG pueda ser transferida al problema de MMB. El resultado que obtiene es que un algoritmo para el análisis sintáctico de TAG que mejore la cota superior de tiempo mínimo dada por $\mathcal{O}(|G|\, n^6)$ puede ser convertido automáticamente en un algoritmo para MMB con una cota superior de tiempo mínimo inferior a $\mathcal{O}(m^3)$, donde m es el orden de las matrices de entrada.

Como resultado de las investigaciones llevadas a cabo en el campo de la MMB, se conocen algoritmos que son asintóticamente más rápidos que $\mathcal{O}(m^3)$, pero cuanto más considerable es la mejora, más complejas son las operaciones involucradas en el cálculo, de tal modo que los algoritmos asintóticamente más rápidos para MMB presentan un interés meramente teórico, puesto que el enorme valor de las constantes involucradas en tales cálculos vuelve prohibitiva su aplicación a casos prácticos. Pero lo que es más importante, el diseño de algoritmos prácticos para MMB que mejoren considerablemente la cota superior cúbica es considerada una tarea extremadamente difícil.

Como ejemplo de algoritmo para análisis sintáctico de TAG que hace uso de MMB tenemos el propuesto por Rajasekaran en [151], de tipo Earley ascendente para TAG binarias, que presenta una complejidad temporal $\mathcal{O}(n^3 M(n))$, donde $M(k)$ es el tiempo necesario para multiplicar dos matrices booleanas de tamaño $k \times k$. Puesto que $\mathcal{O}(k^{2.376})$ es la complejidad mínima alcanzada hasta el momento para $M(k)$, el algoritmo de análisis sintáctico de TAG presenta una complejidad $\mathcal{O}(n^{5.376})$.

De los resultados mostrados en [166] se deduce que el estado actual del problema del análisis sintáctico de TAG, en cuanto a límites de complejidad se refiere, es un problema "difícil de mejorar" y que hay suficiente evidencia para pensar que aquellos algoritmos de análisis que se comporten asintóticamente mejor que $\mathcal{O}(|G|\, n^6)$ probablemente carezcan de interés práctico debido a los costosos cálculos que llevarán aparejados.

Satta y Schuler estudian en [167] diversas restricciones a las gramáticas de adjunción de árboles que permitan la construcción de algoritmos de análisis sintáctico de utilidad práctica con una complejidad temporal inferior a $\mathcal{O}(n^6)$ y que al mismo tiempo retengan la potencia generativa necesaria para expresar las construcciones sintácticas del lenguaje natural que pueden ser modeladas en TAG. De su estudio deducen que permitiendo una única adjunción en la espina de un árbol auxiliar en el que la frontera del árbol adjuntado se extienda a derecha e izquierda del nodo pie, es posible construir algoritmos de análisis sintáctico con complejidad $\mathcal{O}(n^5)$ y la capacidad generativa del formalismo resultante es lo suficientemente potente como para dar cabida a la práctica totalidad de los fenómenos lingüísticos que pueden ser analizados mediante TAG.

Eisnert y Satta estudian en [71] el impacto que produce en la complejidad con respecto a la

[9]En el caso de TAG el tamaño de la gramática se define habitualmente como la suma del número total de nodos en todos los árboles de $\boldsymbol{I} \cup \boldsymbol{A}$.

longitud de la cadena de entrada el fenómeno de la lexicalización presente en las gramáticas de adjunción de árboles lexicalizadas.

2.7.2 Complejidad espacial

La complejidad espacial de los algoritmos de análisis sintáctico asociada a los formalismos gramaticales que generan lenguajes de adjunción de árboles varía entre $\mathcal{O}(|G|\,n^4)$ y $\mathcal{O}(|G|\,n^5)$. Esta última se da en el caso de los algoritmos que preservan al propiedad del prefijo válido [125, 53, 14, 126, 19]. Mientras que el mantenimiento de dicha propiedad no supone un incremento de la complejidad temporal, sí implica un incremento de la complejidad espacial.

Capítulo 3

Algoritmos de análisis sintáctico para TAG

Se describen varios de los algoritmos de análisis sintáctico que han sido propuestos para las gramáticas de adjunción de árboles desde que éstas fueron por descritas por primera vez en [93]. La mayor parte de estos algoritmos están basados en algoritmos de análisis sintáctico independientes del contexto extendidos para el tratamiento de la adjunción. Asimismo es de destacar que prácticamente todos estos algoritmos utilizan técnicas de programación dinámica, obteniendo una complejidad polinómica de orden $\mathcal{O}(n^6)$, donde n es la longitud de la cadena de entrada. La aportación de este capítulo es múltiple: se proporciona un camino evolutivo que relaciona los algoritmos de análisis de TAG más populares; se describen por vez primera algunos algoritmos de análisis de TAG, como por ejemplo, las diferentes versiones de los algoritmos de tipo Earley ascendente y las versiones propuestas de los algoritmos de tipo Earley sin la propiedad del prefijo válido; por último, se realiza una descripción conjunta de la mayor parte de los algoritmos de análisis existentes para TAG. Este capítulo está basado en [8, 9].

3.1 Introducción

A la hora de describir los algoritmos de análisis para TAG, tenemos que elegir una representación adecuada para indicar el reconocimiento parcial de los árboles elementales. En las gramáticas independientes del contexto se utilizan habitualmente producciones con punto para separar la parte de la producción que ya ha sido procesada de la que no lo ha sido aún. En los algoritmos que proceden unidireccionalmente, un solo punto es suficiente, mientras que en aquellos que proceden bidireccionalmente se necesitan dos puntos [189]. En el caso de TAG, al no ser las estructuras elementales producciones sino árboles, deberemos representar árboles con punto. Existen varias alternativas:

1. Al igual que en las gramáticas independientes del contexto se escriben producciones completas con punto, en TAG se podría escribir cada árbol elemental completo con su punto correspondiente. Un ejemplo de utilización de este tipo de notación puede encontrarse en [176].

2. Si identificamos unívocamente todo elemento en una producción independiente del contexto, para lo cual podemos utilizar subíndices correspondientes al número de la producción y a la posición que cada elemento ocupa en al producción de tal modo que la producción $k : N \rightarrow MPQ$ se representaría como $N_{k,0} \rightarrow N_{k,1}N_{k,2}N_{k,3}$, podemos representar una producción con punto simplemente indicando un elemento de la producción contiguo al

punto e indicando si este se encuentra situado a derecha o izquierda de dicho elemento. Del mismo modo, podemos indicar la posición del punto en un árbol elemental de una TAG indicando simplemente un nodo del árbol adyacente al punto y la posición relativa del punto respecto a dicho nodo: arriba-izquierda, abajo-izquierda, abajo-derecha o arriba-derecha. Este es el tipo de notación utilizada en [168].

3. Como un árbol elemental γ puede considerarse constituido por un conjunto de producciones independientes del contexto $\mathcal{P}(\gamma)$, podemos indicar la posición del punto en el árbol simplemente indicando la posición en la producción correspondiente. Este tipo de representación recibe el nombre de *multicapa* en [64].

4. Aplicar un aplanamiento a los árboles, esto es, una conversión de los mismos a cadenas de caracteres parentizadas en las cuales cada nuevo nivel de paréntesis indica un nivel adicional de profundidad en el árbol elemental. Este tipo de representación, que recibe el nombre de *plana* en [64], implica que cada árbol inicial se representa únicamente por una cadena de caracteres mientras que cada árbol auxiliar se representa por dos cadenas correspondientes a las partes situadas a la izquierda y a la derecha de la espina. Dichas cadenas pueden ser utilizadas como partes derechas de las producciones de una gramática independiente del contexto que permitiría representar de modo conciso los árboles elementales de una gramática de adjunción [64].

La primera alternativa presenta el inconveniente de que las representaciones lineales de árboles suelen dificultar su comprensión. Alternativamente podrían utilizarse representaciones pictóricas de los árboles, que si bien son incómodas a lo hora de describir los pasos deductivos del algoritmo pueden ser útiles para representar gráficamente el comportamiento del algoritmo. La segunda alternativa, aunque muy manejable y concisa tiene el inconveniente de que es necesario recurrir a las descripciones de los árboles elementales para poder ver el contexto en el que se está aplicando cada paso del algoritmo. La tercera alternativa resuelve este problema al proporcionar un contexto formado por los hermanos del nodo considerado, que si bien es limitado, proporciona al lector la información suficiente la mayor parte de las veces. La desventaja es que normalmente deberemos añadir pasos deductivos extra que realizan el recorrido del árbol en un orden determinado. La cuarta alternativa, aunque en principio supone una reducción del número de producciones independientes al contexto, en la práctica su utilización no implica generalmente una reducción del número de pasos deductivos utilizados para describir el procesamiento realizado por los algoritmos de análisis sintáctico[1] y contribuye a oscurecer la notación.

En nuestro caso hemos preferido utilizar la representación multicapa, por lo que indicaremos la posición del punto en un árbol mediante una producción $N \to \delta \bullet \nu$, donde $\delta\nu$ son los hijos de N. En el caso de los elementos del lado derecho de las producciones cuyas etiquetas pertenecen a $V_T \cup \epsilon$, puesto que no pueden tener descendientes ni son susceptibles de ser nodos de adjunción, consideraremos directamente su etiqueta a la hora de describir las producciones, en lugar del nodo propiamente dicho[2]. Adicionalmente y con el fin de simplificar los esquemas de análisis de los algoritmos más complejos, seguiremos el enfoque de [125] de considerar la producción adicional $\top \to \mathbf{R}^\alpha$ para cada árbol inicial α y las dos producciones adicionales siguientes para cada árbol auxiliar β: $\top \to \mathbf{R}^\beta$ y $\mathbf{F}^\beta \to \bot$, donde $\mathbf{R}^\beta$ y $\mathbf{F}^\beta$ se refieren a los nodos raíz y pie de β, respectivamente. Con el fin de no modificar la capacidad generativa de las gramáticas,

[1] Como ejemplo, podemos comparar el algoritmo descrito en [64] con el representado por el esquema 3.7.

[2] Con esta convención, que se adopta porque simplifica considerablemente la notación, la única forma de distinguir diferentes hojas de un árbol elemental con la misma etiqueta es referenciando los respectivos padres o la producción independiente del contexto de la que forman parte.

los nuevos nodos $\top$ y $\perp$ no pueden ser nodos de adjunción, por lo cual la nueva gramática se asemeja a una TAG *limpia*[3].

Con respecto a las restricciones de adjunción, $\beta \in \mathrm{adj}(N^\gamma)$ si $\beta \in \boldsymbol{A}$ puede ser adjuntado en N^γ. Si $\beta = \mathbf{nil}$ entonces la adjunción en el nodo N^γ es opcional. Si **nil** es el único valor que retorna la función para un nodo N^γ, entonces no es posible realizar ninguna adjunción en dicho nodo.

Ejemplo 3.1 El árbol inicial α de la figura 3.1 se representa en notación multicapa mediante el siguiente conjunto de producciones independientes del contexto:

$$\top \to \langle \alpha, 0 \rangle$$

$$\langle \alpha, 0 \rangle \to a\ \langle \alpha, 2 \rangle$$

$$\langle \alpha, 2 \rangle \to c$$

El árbol auxiliar β de la misma figura se representa mediante el siguiente conjunto de producciones:

$$\top \to \langle \beta, 0 \rangle$$

$$\langle \beta, 0 \rangle \to a\ \langle \beta, 2 \rangle$$

$$\langle \beta, 2 \rangle \to \langle \beta, 2.1 \rangle\ c$$

$$\langle \beta, 2.1 \rangle \to \perp$$

En ambos casos $\langle \alpha, g \rangle$ denota el nodo de α que ocupa la posición g según el direccionamiento de Gorn[4]. ¶

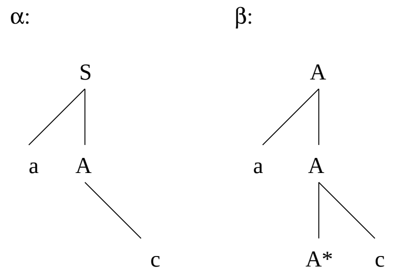

Figura 3.1: Ejemplos de árbol inicial y auxiliar

La relación $\Rightarrow$ de derivación sobre $\mathcal{P}(\gamma)$ se define como $\delta \Rightarrow \nu$ si existen $\delta', \delta'', M^\gamma, \upsilon$ tal que $\delta = \delta' M^\gamma \delta''$, $\nu = \delta' \upsilon \delta''$ y existe una producción $M^\gamma \to \upsilon \in \mathcal{P}(\gamma)$. Mediante $\stackrel{*}{\Rightarrow}$ denotaremos el cierre reflexivo y transitivo de $\Rightarrow$.

Sea $\mathcal{P}(\mathcal{T}) = \bigcup_{\gamma \in \boldsymbol{I} \cup \boldsymbol{A}} \mathcal{P}(\gamma)$ el conjunto de todas las producciones independientes del contexto presentes en los árboles de una gramática de adjunción de árboles $\mathcal{T}$. Con el fin de ser capaces de representar derivaciones que incluyan adjunciones extendemos la relación $\Rightarrow$ a $\mathcal{P}(\mathcal{T})$, de tal modo que $\delta \stackrel{*}{\Rightarrow} \nu$ si existen $\delta', \delta'', M^\gamma, \upsilon$ tal que $\delta = \delta' M^\gamma \delta''$, $\mathbf{R}^\beta \stackrel{*}{\Rightarrow} \upsilon_1 \mathbf{F}^\beta \upsilon_3$, $\beta \in \mathrm{adj}(M^\gamma)$, $M^\gamma \to \upsilon_2$ y $\nu = \delta' \upsilon_1 \upsilon_2 \upsilon_3 \delta''$.

[3] Poller y Becker [149] denominan TAG *limpias* a las gramáticas de adjunción en las que los nodos raíz de los árboles elementales y los nodos pie de los árboles auxiliares tienen restricciones de adjunción nulas.

[4] En el direccionamiento de Gorn se utiliza 0 para referirse al nodo raíz, k para referirse al k-ésimo hijo del nodo raíz y $p.q$ para referirse al q-ésimo hijo del nodo con dirección p.

La mayoría de los algoritmos de análisis sintáctico diseñados para TAG se corresponden con adaptaciones de algoritmos para el análisis de gramáticas independientes del contexto y por lo tanto recibirán el mismo nombre en ambos casos. Solamente en aquellos casos en los que pueda existir confusión diferenciaremos explícitamente entre unos y otros[5]. Para describir los diferentes algoritmos de análisis sintáctico haremos uso de *esquemas de análisis*, una estructura para realizar descripciones de alto nivel de algoritmos de análisis desarrollada por Sikkel en [189] y que se describe en el apéndice A.

3.2 Algoritmo de tipo CYK

A continuación mostramos una extensión para TAG del algoritmo CYK de análisis sintáctico (ver sección B.1) basado en el algoritmo descrito en [209, 206] pero con algunas correcciones. Asumiremos que todo nodo de un árbol elemental de la gramática tiene a lo sumo dos descendientes. Este condicionante puede verse como una trasposición de la forma normal de Chomsky [85] al caso de las gramáticas de adjunción.

El reconocimiento de los nodos de un árbol elemental se realiza aplicando casi literalmente el algoritmo para el caso independiente del contexto. Sin embargo ahora es preciso definir cómo reconocer la operación de adjunción de forma totalmente ascendente. Para ello se definen ítems de la forma

$$\left\{ \begin{array}{ll} [N^\gamma, i, j \mid p, q \mid adj] \mid & N^\gamma \stackrel{*}{\Rightarrow} a_{i+1} \ldots a_p \, \mathbf{F}^\gamma \, a_{q+1} \ldots a_j \stackrel{*}{\Rightarrow} a_{i+1} \ldots a_j \quad \text{sii } (p,q) \neq (-,-) \\ & N^\gamma \stackrel{*}{\Rightarrow} a_{i+1} \ldots a_j \quad \text{sii } (p,q) = (-,-) \end{array} \right\}$$

que contienen un nuevo componente *adj* con respecto a los ítems definidos en [209, 206], que toma su valor del conjunto $\{\text{true}, \text{false}\}$ y que indica

- si su valor es *true* significa que el ítem es el resultado de una operación de adjunción ya totalmente realizada;
- si valor es *false* significa que el ítem no es el resultado de una adjunción.

Con ello podemos limitar a una sola la cantidad de adjunciones que se pueden realizar sobre cada nodo de un árbol elemental y podemos también asegurar el cumplimiento de las restricciones de adjunción obligatoria, ajustándonos de este modo a lo establecido en el formalismo de las gramáticas de adjunción [175].

Esquema de análisis sintáctico 3.1 El sistema de análisis $\mathbb{P}_{\text{CYK}}$ que se corresponde con el algoritmo CYK para una gramática de adjunción de árboles $\mathcal{T}$ y una cadena de entrada $a_1 \ldots a_n$ se define como sigue:

$$\mathcal{I}_{\text{CYK}} = \left\{ \begin{array}{ll} [N^\gamma, i, j \mid p, q \mid adj] \mid & \text{etiqueta}(N^\gamma) \in V_N, \ \gamma \in \boldsymbol{I} \cup \boldsymbol{A}, \ 0 \leq i \leq j, \\ & (p,q) \leq (i,j), \ adj \in \{\text{true}, \text{false}\} \end{array} \right\}$$

$$\mathcal{H}_{\text{CYK}} = \left\{ \ [a, i-1, i] \mid a = a_i, 1 \leq i \leq n \ \right\}$$

$$\mathcal{D}_{\text{CYK}}^{\text{Scan}} = \frac{[a, i, i+1]}{[N^\gamma, i, i+1 \mid -, - \mid \text{false}]} \quad N^\gamma \to a \in \mathcal{P}(\gamma)$$

[5] En el apéndice B se describen sucintamente los algoritmos de análisis sintántico para gramáticas independientes del contexto que son relevantes en este capítulo

$$\mathcal{D}^{\epsilon}_{\mathrm{CYK}} = \frac{}{[N^\gamma, i, i \mid -, - \mid \mathrm{false}]} \quad N^\gamma \to \epsilon \in \mathcal{P}(\gamma)$$

$$\mathcal{D}^{\mathrm{Foot}}_{\mathrm{CYK}} = \frac{}{[\mathbf{F}^\gamma, i, j \mid i, j \mid \mathrm{false}]}$$

$$\mathcal{D}^{\mathrm{LeftDom}}_{\mathrm{CYK}} = \frac{[M^\gamma, i, k \mid p, q \mid adj1],\ [P^\gamma, k, j \mid -, - \mid adj2]}{[N^\gamma, i, j \mid p, q \mid \mathrm{false}]} \quad \begin{array}{l} N^\gamma \to M^\gamma P^\gamma \in \mathcal{P}(\gamma), \\ M^\gamma \in \mathrm{espina}(\gamma), \\ adj1 = \mathrm{false\ sii\ } \mathbf{nil} \in \mathrm{adj}(M^\gamma), \\ adj2 = \mathrm{false\ sii\ } \mathbf{nil} \in \mathrm{adj}(P^\gamma) \end{array}$$

$$\mathcal{D}^{\mathrm{RightDom}}_{\mathrm{CYK}} = \frac{[M^\gamma, i, k \mid -, - \mid adj1],\ [P^\gamma, k, j \mid p, q \mid adj2]}{[N^\gamma, i, j \mid p, q \mid \mathrm{false}]} \quad \begin{array}{l} N^\gamma \to M^\gamma P^\gamma \in \mathcal{P}(\gamma), \\ P^\gamma \in \mathrm{espina}(\gamma), \\ adj1 = \mathrm{false\ sii\ } \mathbf{nil} \in \mathrm{adj}(M^\gamma), \\ adj2 = \mathrm{false\ sii\ } \mathbf{nil} \in \mathrm{adj}(P^\gamma) \end{array}$$

$$\mathcal{D}^{\mathrm{NoDom}}_{\mathrm{CYK}} = \frac{[M^\gamma, i, k \mid -, - \mid adj1],\ [P^\gamma, k, j \mid -, - \mid adj2]}{[N^\gamma, i, j \mid -, - \mid \mathrm{false}]} \quad \begin{array}{l} N^\gamma \to M^\gamma P^\gamma \in \mathcal{P}(\gamma), \\ N^\gamma \notin \mathrm{espina}(\gamma), \\ adj1 = \mathrm{false\ sii\ } \mathbf{nil} \in \mathrm{adj}(M^\gamma), \\ adj2 = \mathrm{false\ sii\ } \mathbf{nil} \in \mathrm{adj}(P^\gamma) \end{array}$$

$$\mathcal{D}^{\mathrm{Unary}}_{\mathrm{CYK}} = \frac{[M^\gamma, i, j \mid p, q \mid adj]}{[N^\gamma, i, j \mid p, q] \mid \mathrm{false}]} \quad \begin{array}{l} N^\gamma \to M^\gamma \in \mathcal{P}(\gamma), \\ adj = \mathrm{false\ sii\ } \mathbf{nil} \in \mathrm{adj}(M^\gamma) \end{array}$$

$$\mathcal{D}^{\mathrm{Adj}}_{\mathrm{CYK}} = \frac{[\mathbf{R}^\beta, i', j' \mid i, j \mid adj],\ [N^\gamma, i, j \mid p, q \mid \mathrm{false}]}{[N^\gamma, i', j' \mid p, q \mid \mathrm{true}]} \quad \beta \in \boldsymbol{A},\ \beta \in \mathrm{adj}(N^\gamma)$$

$$\mathcal{D}_{\mathrm{CYK}} = \mathcal{D}^{\mathrm{Scan}}_{\mathrm{CYK}} \cup \mathcal{D}^{\epsilon}_{\mathrm{CYK}} \cup \mathcal{D}^{\mathrm{Foot}}_{\mathrm{CYK}} \cup \mathcal{D}^{\mathrm{LeftDom}}_{\mathrm{CYK}} \cup \mathcal{D}^{\mathrm{RightDom}}_{\mathrm{CYK}} \cup \mathcal{D}^{\mathrm{NoDom}}_{\mathrm{CYK}} \cup \mathcal{D}^{\mathrm{Unary}}_{\mathrm{CYK}} \cup \mathcal{D}^{\mathrm{Adj}}_{\mathrm{CYK}}$$

$$\mathcal{F}_{\mathrm{CYK}} = \left\{\ [\mathbf{R}^\alpha, 0, n \mid -, - \mid adj] \mid \alpha \in \boldsymbol{I},\ adj = \mathrm{false\ sii\ } \mathbf{nil} \in \mathrm{adj}(\mathbf{R}^\alpha)\ \right\}$$

donde $(p, q) \leq (i, j)$ se cumple si $i \leq p \leq q \leq j$ en el caso de que $(p, q) \neq (-, -)$ y si $i \leq j$ en el caso de que $(p, q) = (-, -)$. §

La definición de las hipótesis realizada en este sistema de análisis sintáctico se corresponde con la estándar y es la misma que se utilizará en los restantes sistemas de análisis del capítulo. Por consiguiente, no nos volveremos a referir explícitamente a ellas.

Los pasos clave en el sistema de análisis $\mathbb{P}_{\mathrm{CYK}}$ son $\mathcal{D}^{\mathrm{Foot}}_{\mathrm{CYK}}$ y $\mathcal{D}^{\mathrm{Adj}}_{\mathrm{CYK}}$, puesto que son los que definen el mecanismo de adjunción. Los demás pasos se encargan de recorrer de forma ascendente los árboles elementales, propagando la información relativa a la porción de la cadena de entrada cubierta por el pie en el caso de los árboles auxiliares. El paso deductivo $\mathcal{D}^{\mathrm{Scan}}_{\mathrm{CYK}}$ permite resolver la limitación planteada en [209, 206] con respecto a nodos frontera etiquetados con ϵ.

El paso deductivo $\mathcal{D}^{\mathrm{Foot}}_{\mathrm{CYK}}$ hace posible el análisis ascendente de los árboles auxiliares, puesto que predice todas las posibles porciones de la cadena de entrada que puede cubrir el pie. Si no fuese así, sería necesario esperar al reconocimiento total del pie antes de poder continuar el reconocimiento del árbol auxiliar, con lo cual el algoritmo de análisis no sería totalmente ascendente.

Una de las consecuencias de la utilización del paso $\mathcal{D}_{\mathrm{CYK}}^{\mathrm{Foot}}$ es que existirán múltiples análisis diferentes para cada árbol auxiliar, diferentes únicamente en las posición de la cadena de entrada que se *supone* en cada caso que cubre el pie. Pero finalmente sólo uno o unos pocos de esos árboles podrán formar parte del árbol de derivación, aquellos que hayan predicho correctamente el pie. La realización de esta comprobación le corresponde al paso $\mathcal{D}_{\mathrm{CYK}}^{\mathrm{Adj}}$. Efectivamente, cuando se ha alcanzado la raíz de un árbol auxiliar, se comprueba si existe un subárbol de un árbol elemental cuya raíz pueda ser un nodo de adjunción para dicho árbol auxiliar y que además cubra la porción de la cadena de entrada que espera ser cubierta por el pie. En caso de que así sea, se eliminará la posibilidad de realizar otras adjunciones en ese nodo y el análisis continuará ascendiendo por el nuevo árbol elemental. En la figura 3.2 se muestra gráficamente la aplicación de este paso, donde los dos árboles de la izquierda se corresponden con las descripciones de los ítems antecedentes y el árbol de la derecha con la descripción del ítem consecuente. Se utilizan líneas punteadas para indicar partes de árboles elementales que no han sido aún analizadas, mientras que las espinas se resaltan mediante líneas gruesas.

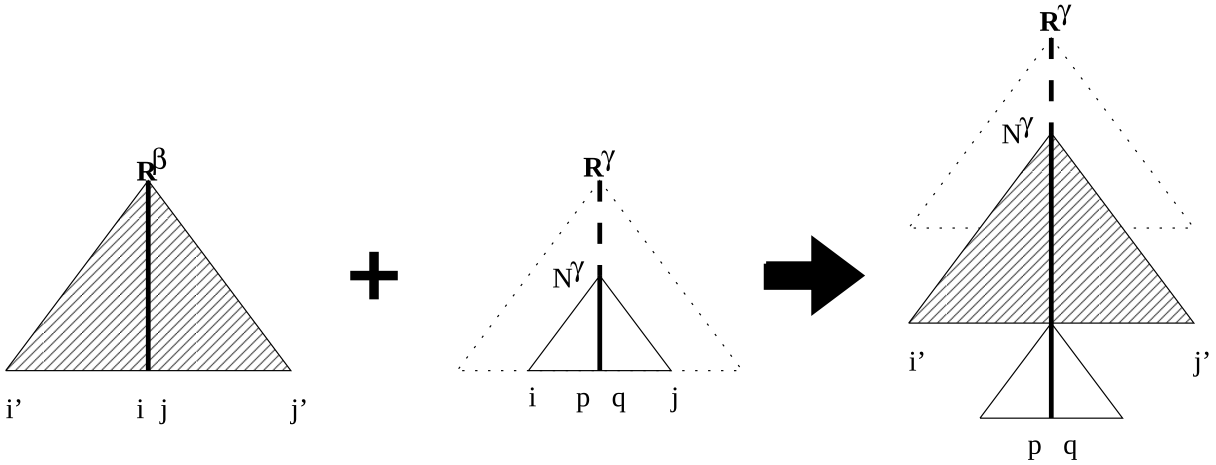

Figura 3.2: Descripción gráfica de un paso $\mathcal{D}_{\mathrm{CYK}}^{\mathrm{Adj}}$

La complejidad temporal del algoritmo con respecto a la longitud n de la cadena de entrada es $\mathcal{O}(n^6)$ y viene dada por el paso deductivo encargado de la adjunción, que debe combinar seis posiciones de la cadena de entrada. La complejidad espacial con respecto a la cadena de entrada es $\mathcal{O}(n^4)$, puesto que cada ítem almacena cuatro posiciones de la cadena de entrada.

3.3 Algoritmos de tipo Earley ascendente

El algoritmo CYK presenta un limitación muy importante: sólo es aplicable a gramáticas en las cuales un nodo no tiene más de dos descendentes. Nuestro objetivo ahora es extender el esquema **CYK** a la clase general de TAG, obteniendo lo que podríamos llamar un esquema Earley ascendente (ver sección B.2) extendido a gramáticas de adjunción de árboles. Debemos reseñar que no conocemos ninguna adaptación anterior para TAG de este algoritmo.

Como primer paso para la realización de un algoritmo de tipo Earley ascendente para gramáticas de adjunción de árboles precisamos introducir puntos en las producciones que representan los árboles elementales, por lo que los ítems que utilizaremos tendrán la forma

$$\left\{ \begin{array}{lll} [N^\gamma \to \delta \bullet \nu, i, j \mid p, q \mid adj] \mid & \delta \stackrel{*}{\Rightarrow} a_{i+1} \ldots a_p \, \mathbf{F}^\gamma \, a_{q+1} \ldots a_j \stackrel{*}{\Rightarrow} a_{i+1} \ldots a_j & \text{sii } (p,q) \neq (-,-) \\ & \delta \stackrel{*}{\Rightarrow} a_{i+1} \ldots a_j & \text{sii } (p,q) = (-,-) \end{array} \right\}$$

Los ítems del nuevo esquema de análisis sintáctico, que denominaremos **buE₁**, son por tanto un refinamiento de los ítems de **CYK**. Sobre los pasos deductivos aplicaremos también un *refinamiento* puesto que los pasos de tipo LeftDom, RightDom y NoDom serán separados en diferentes pasos de tipo Init y Comp, mientras que los pasos de tipo Unary y ϵ ya no serán necesarios puesto que todas las producciones serán tratadas uniformemente con independencia de su longitud. Finalmente, se realizará una extensión del dominio de las producciones permitiendo árboles con un grado arbitrario de ramificación.

Esquema de análisis sintáctico 3.2 El sistema de análisis $\mathbb{P}_{\text{buE}_1}$ que se corresponde con una primera extensión del algoritmo de Earley ascendente para TAG, dada una gramática de adjunción de árboles $\mathcal{T}$ y una cadena de entrada $a_1 \ldots a_n$ se define como sigue:

$$\mathcal{I}_{\text{buE}_1} = \left\{ \begin{array}{ll} [N^\gamma \to \delta \bullet \nu, i, j \mid p, q \mid adj] \mid & N^\gamma \to \delta\nu \in \mathcal{P}(\gamma), \ \gamma \in \boldsymbol{I} \cup \boldsymbol{A}, \\ & 0 \leq i \leq j, \ (p,q) \leq (i,j), \ adj \in \{\text{true}, \text{false}\} \end{array} \right\}$$

$$\mathcal{D}^{\text{Init}}_{\text{buE}_1} = \frac{}{[N^\gamma \to \bullet \delta, i, i \mid -, - \mid \text{false}]}$$

$$\mathcal{D}^{\text{Foot}}_{\text{buE}_1} = \frac{}{[\mathbf{F}^\beta \to \bot \bullet, i, j \mid i, j \mid \text{false}]}$$

$$\mathcal{D}^{\text{Scan}}_{\text{buE}_1} = \frac{[N^\gamma \to \delta \bullet a\nu, i, j \mid p, q \mid \text{false}], \ [a, j, j+1]}{[N^\gamma \to \delta a \bullet \nu, i, j+1 \mid p, q \mid \text{false}]}$$

$$\mathcal{D}^{\text{Comp}}_{\text{buE}_1} = \frac{\begin{array}{l} [N^\gamma \to \delta \bullet M^\gamma \nu, i, k \mid p', q' \mid \text{false}], \\ [M^\gamma \to \upsilon \bullet, i, j \mid p, q \mid adj] \end{array}}{[N^\gamma \to \delta M^\gamma \bullet \nu, i, j \mid p' \cup p, q' \cup q \mid \text{false}]} \quad adj = \text{false sii } \mathbf{nil} \in \text{adj}(M^\gamma)$$

$$\mathcal{D}^{\text{Adj}}_{\text{buE}_1} = \frac{[\top \to \mathbf{R}^\beta \bullet, k, j \mid k', j' \mid \text{false}], \ [M^\gamma \to \upsilon \bullet, k', j' \mid p, q \mid \text{false}]}{[M^\gamma \to \upsilon \bullet, k, j \mid p, q \mid \text{true}]} \quad \beta \in \boldsymbol{A}, \ \beta \in \text{adj}(\gamma)$$

$$\mathcal{D}_{\text{buE}_1} = \mathcal{D}^{\text{Init}}_{\text{buE}_1} \cup \mathcal{D}^{\text{Foot}}_{\text{buE}_1} \cup \mathcal{D}^{\text{Scan}}_{\text{buE}_1} \cup \mathcal{D}^{\text{Comp}}_{\text{buE}_1} \cup \mathcal{D}^{\text{Adj}}_{\text{buE}_1}$$

$$\mathcal{F}_{\text{buE}_1} = \left\{ \ [\top \to \mathbf{R}^\alpha \bullet, 0, n \mid -, - \mid false] \mid \alpha \in \boldsymbol{I} \ \right\}$$

y donde $p \cup q$ se refiere a la operación de unión de índices, función parcial de enteros a enteros definida como

$$p \cup q = \left\{ \begin{array}{ll} p & \text{si } q = - \\ q & \text{si } p = - \\ - & \text{si } p = q = - \end{array} \right.$$

donde $-$ se utiliza para indicar que el valor de uno de los parámetros está indefinido. §

Los pasos $\mathcal{D}^{\text{Init}}_{\text{buE}_1}$ son los encargados de lanzar el análisis ascendente. Los ítems generados por estos pasos son siempre válidos, puesto que no expanden ninguna porción la cadena de entrada. Tan sólo expresan la expectativa de que una determinada producción pueda ser aplicada para reconocer una porción de la cadena que comienza en una determinada posición.

Los pasos $\mathcal{D}^{\text{Foot}}_{\text{buE}_1}$ son utilizados, al igual que en el caso CYK, para predecir la porción de la cadena de entrada cubierta por el pie de un árbol auxiliar.

El algoritmo procede a reconocer ascendentemente los árboles auxiliares mediante la aplicación de pasos $\mathcal{D}^{\text{Comp}}_{\text{buE}_1}$, propagando también la información correspondiente a la porción de la cadena de entrada cubierta por el pie en el caso de los árboles auxiliares.

El paso deductivo $\mathcal{D}^{\text{Adj}}_{\text{buE}_1}$ se comporta de modo idéntico al paso homónimo del algoritmo CYK, comprobando que las predicciones respecto al pie se corresponden con las de una adjunción que se ha realizado realmente.

Proposición 3.1 $\mathbf{CYK} \stackrel{\text{ir}}{\Longrightarrow} \mathbf{CYK'} \stackrel{\text{sr}}{\Longrightarrow} \mathbf{ECYK} \stackrel{\text{ext}}{\Longrightarrow} \mathbf{buE_1}$.

Demostración:

Como primer paso definiremos el sistema de análisis $\mathbb{P}_{\text{CYK}'}$ para una gramática de adjunción $\mathcal{T}$ y una cadena de entrada $a_1 \dots a_n$.

$$\mathcal{I}_{\text{CYK}'} = \left\{ \begin{array}{ll} [N^\gamma \to \delta \bullet \nu, i, j \mid p, q \mid adj] \mid & N^\gamma \to \delta\nu \in \mathcal{P}(\gamma),\ \gamma \in \boldsymbol{I} \cup \boldsymbol{A} \\ & 0 \le i \le j,\ (p,q) \le (k,j),\ adj \in \{\text{true}, \text{false}\} \end{array} \right\}$$

$$\mathcal{D}^{\text{Scan}}_{\text{CYK}'} = \frac{[a, i, i+1]}{[N^\gamma \to a\bullet, i, i+1 \mid -,- \mid \text{false}]}$$

$$\mathcal{D}^{\epsilon}_{\text{CYK}'} = \frac{}{[N^\gamma \to \bullet, i, i \mid -,- \mid \text{false}]} \quad N^\gamma \to \epsilon$$

$$\mathcal{D}^{\text{Foot}}_{\text{CYK}'} = \frac{}{[\mathbf{F}^\gamma \to \bot\bullet, i, j \mid i, j \mid \text{false}]}$$

$$\mathcal{D}^{\text{LeftDom}}_{\text{CYK}'} = \frac{\begin{array}{c}[M^\gamma \to \delta\bullet, i, k \mid p, q \mid adj1],\\ [P^\gamma \to \nu\bullet, k, j \mid -,- \mid adj2]\end{array}}{[N^\gamma \to M^\gamma P^\gamma\bullet, i, j \mid p, q \mid \text{false}]} \quad \begin{array}{l} N^\gamma \to M^\gamma P^\gamma \in \mathcal{P}(\gamma), \\ M^\gamma \in \text{espina}(\gamma), \\ adj1 = \text{false sii } \mathbf{nil} \in \text{adj}(M^\gamma), \\ adj2 = \text{false sii } \mathbf{nil} \in \text{adj}(P^\gamma) \end{array}$$

$$\mathcal{D}^{\text{RightDom}}_{\text{CYK}'} = \frac{\begin{array}{c}[M^\gamma \to \delta\bullet, i, k \mid -,- \mid adj1],\\ [P^\gamma \to \nu\bullet, k, j \mid p, q \mid adj2]\end{array}}{[N^\gamma \to M^\gamma P^\gamma\bullet, i, j \mid p, q \mid \text{false}]} \quad \begin{array}{l} N^\gamma \to M^\gamma P^\gamma \in \mathcal{P}(\gamma), \\ P^\gamma \in \text{espina}(\gamma), \\ adj1 = \text{false sii } \mathbf{nil} \in \text{adj}(M^\gamma), \\ adj2 = \text{false sii } \mathbf{nil} \in \text{adj}(P^\gamma) \end{array}$$

$$\mathcal{D}^{\text{NoDom}}_{\text{CYK}'} = \frac{\begin{array}{c}[M^\gamma \to \delta\bullet, i, k \mid -,- \mid adj1],\\ [P^\gamma \to \nu\bullet, k, j \mid -,- \mid adj2]\end{array}}{[N^\gamma \to M^\gamma P^\gamma\bullet, i, j \mid -,- \mid \text{false}]} \quad \begin{array}{l} N^\gamma \to M^\gamma P^\gamma \in \mathcal{P}(\gamma), \\ N^\gamma \notin \text{espina}(\gamma), \\ adj1 = \text{false sii } \mathbf{nil} \in \text{adj}(M^\gamma), \\ adj2 = \text{false sii } \mathbf{nil} \in \text{adj}(P^\gamma) \end{array}$$

$$\mathcal{D}^{\text{Unary}}_{\text{CYK}'} = \frac{[M^\gamma \to \delta\bullet, i, j \mid p, q \mid adj]}{[N^\gamma \to M^\gamma\bullet, i, j \mid p, q] \mid \text{false}]} \quad \begin{array}{l} N^\gamma \to M^\gamma \in \mathcal{P}(\gamma), \\ adj = \text{false sii } \mathbf{nil} \in \text{adj}(M^\gamma) \end{array}$$

$$\mathcal{D}^{\text{Adj}}_{\text{CYK}'} = \frac{[\top \to \mathbf{R}^{\beta}\bullet, i', j' \mid i, j \mid adj],\ [N^{\gamma} \to \delta\bullet, i, j \mid p, q \mid \text{false}]}{[N^{\gamma} \to \delta\bullet, i', j' \mid p, q \mid \text{true}]} \quad \beta \in \boldsymbol{A},\ \beta \in \text{adj}(N^{\gamma})$$

$$\mathcal{D}_{\text{CYK}'} = \mathcal{D}^{\text{Scan}}_{\text{CYK}'} \cup \mathcal{D}^{\epsilon}_{\text{CYK}'} \cup \mathcal{D}^{\text{Foot}}_{\text{CYK}'} \cup \mathcal{D}^{\text{LeftDom}}_{\text{CYK}'} \cup \mathcal{D}^{\text{RightDom}}_{\text{CYK}'} \cup \mathcal{D}^{\text{NoDom}}_{\text{CYK}'} \cup \mathcal{D}^{\text{Unary}}_{\text{CYK}'} \cup \mathcal{D}^{\text{Adj}}_{\text{CYK}'}$$

$$\mathcal{F}_{\text{CYK}'} = \left\{\ [\top \to \mathbf{R}^{\alpha}\bullet, 0, n \mid -, - \mid adj] \mid \alpha \in \boldsymbol{I},\ adj = \text{false sii } \mathbf{nil} \in \text{adj}(\mathbf{R}^{\alpha})\ \right\}$$

Para demostrar que $\mathbf{CYK} \overset{\mathbf{ir}}{\Longrightarrow} \mathbf{CYK}'$, definiremos la siguiente función

$$f([N^{\gamma} \to \delta\bullet, i, j \mid p, q \mid adj]) = [N^{\gamma}, i, j \mid p, q \mid adj]$$

de la cual se obtiene directamente que $\mathcal{I}_{\text{CYK}} = f(\mathcal{I}_{\text{CYK}'})$ y que $\Delta_{\text{CYK}} = f(\Delta_{\text{CYK}'})$ por inducción en la longitud de las secuencias de derivación. En consecuencia, $\mathbb{P}_{\text{CYK}} \overset{\mathbf{ir}}{\longrightarrow} \mathbb{P}_{\text{CYK}'}$, con lo que hemos probado lo que pretendíamos.

Definiremos ahora el sistema de análisis sintáctico $\mathbb{P}_{\text{ECYK}}$ para una gramática de adjunción de árboles $\mathcal{T}$ en la que ningún nodo puede tener más de dos descendientes y una cadena de entrada $a_1 \dots a_n$ dada:

$$\mathcal{I}_{\text{ECYK}} = \mathcal{I}_{\text{CYK}'} = \mathcal{I}_{\text{buE}_1}$$

$$\mathcal{D}^{\text{Init}}_{\text{ECYK}} = \mathcal{D}^{\text{Init}}_{\text{buE}_1}$$

$$\mathcal{D}^{\text{Foot}}_{\text{ECYK}} = \mathcal{D}^{\text{Foot}}_{\text{buE}_1}$$

$$\mathcal{D}^{\text{Scan}}_{\text{ECYK}} = \mathcal{D}^{\text{Scan}}_{\text{buE}_1}$$

$$\mathcal{D}^{\text{Comp}}_{\text{ECYK}} = \mathcal{D}^{\text{Comp}}_{\text{buE}_1}$$

$$\mathcal{D}^{\text{Adj}}_{\text{ECYK}} = \mathcal{D}^{\text{Adj}}_{\text{buE}_1}$$

$$\mathcal{D}_{\text{ECYK}} = \mathcal{D}^{\text{Init}}_{\text{ECYK}} \cup \mathcal{D}^{\text{Foot}}_{\text{ECYK}} \cup \mathcal{D}^{\text{Scan}}_{\text{ECYK}} \cup \mathcal{D}^{\text{Comp}}_{\text{ECYK}} \cup \mathcal{D}^{\text{Adj}}_{\text{ECYK}}$$

$$\mathcal{F}_{\text{ECYK}} = \mathcal{F}_{\text{buE}_1}$$

Para demostrar que $\mathbf{CYK}' \overset{\mathbf{sr}}{\Longrightarrow} \mathbf{ECYK}$, deberemos demostrar que para todo sistema de análisis $\mathbb{P}_{\text{CYK}'}$ y $\mathbb{P}_{\text{ECYK}}$ se cumple que $\mathcal{I}_{\text{CYK}'} \subseteq \mathcal{I}_{\text{ECYK}}$ y que $\overset{*}{\vdash}_{\text{CYK}'} \subseteq \overset{*}{\vdash}_{\text{ECYK}}$. Lo primero es cierto por definición, puesto que $\mathcal{I}_{\text{CYK}'} = \mathcal{I}_{\text{ECYK}}$. Para lo segundo debemos mostrar que $\mathcal{D}_{\text{CYK}'} \subseteq \overset{*}{\vdash}_{\text{ECYK}}$. Consideremos caso por caso:

- un paso deductivo $\mathcal{D}^{\text{Scan}}_{\text{CYK}'}$ es equivalente a la secuencia de pasos deductivos constituida por la aplicación de un paso $\mathcal{D}^{\text{Init}}_{\text{ECYK}}$ y un paso $\mathcal{D}^{\text{Scan}}_{\text{ECYK}}$:

$$\frac{}{[N^{\gamma} \to \bullet a, i, i, \mid -, - \mid \text{false}]}$$

$$\frac{[N^{\gamma} \to \bullet a, i, i, \mid -, - \mid \text{false}], [a, i, i+1]}{[N^{\gamma} \to a\bullet, i, i+1, \mid -, - \mid \text{false}]}$$

- un paso deductivo $\mathcal{D}^{\epsilon}_{\text{CYK}'}$ es equivalente a un paso $\mathcal{D}^{\text{Init}}_{\text{ECYK}}$:

$$\frac{}{[N^{\gamma} \to \bullet, i, i, \mid -, - \mid \text{false}]}$$

- $\mathcal{D}^{\text{Foot}}_{\text{ECYK}}$=$\mathcal{D}^{\text{Foot}}_{\text{CYK}'}$.

- un paso deductivo $\mathcal{D}^{\text{LeftDom}}_{\text{CYK}'}$ es equivalente a la secuencia de pasos deductivos constituida por un paso $\mathcal{D}^{\text{Init}}_{\text{ECYK}}$ y dos pasos $\mathcal{D}^{\text{Comp}}_{\text{ECYK}}$:

$$\frac{}{[N^\gamma \to \bullet M^\gamma P^\gamma, i, i, \mid -, - \mid \text{false}]}$$

$$\frac{[N^\gamma \to \bullet M^\gamma P^\gamma, i, i, \mid -, - \mid \text{false}],\ [M^\gamma \to \delta\bullet, i, k \mid p, q \mid adj]}{[N^\gamma \to M^\gamma \bullet P^\gamma, i, k, \mid p, q \mid \text{false}]}$$

$$\frac{[N^\gamma \to M^\gamma \bullet P^\gamma, i, k, \mid p, q \mid \text{false}], [P^\gamma \to \nu\bullet, k, j \mid -, - \mid adj]}{[N^\gamma \to M^\gamma P^\gamma \bullet, i, j, \mid p, q \mid \text{false}]}$$

- un paso $\mathcal{D}^{\text{RightDom}}_{\text{CYK}'}$ es equivalente a la secuencia formada por un paso $\mathcal{D}^{\text{Init}}_{\text{ECYK}}$ y dos pasos $\mathcal{D}^{\text{Comp}}_{\text{ECYK}}$:

$$\frac{}{[N^\gamma \to \bullet M^\gamma P^\gamma, i, i, \mid -, - \mid \text{false}]}$$

$$\frac{[N^\gamma \to \bullet M^\gamma P^\gamma, i, i, \mid -, - \mid \text{false}],\ [M^\gamma \to \delta\bullet, i, k \mid -, - \mid adj]}{[N^\gamma \to M^\gamma \bullet P^\gamma, i, k, \mid -, - \mid \text{false}]}$$

$$\frac{[N^\gamma \to M^\gamma \bullet P^\gamma, i, k, \mid -, - \mid \text{false}],\ [P^\gamma \to \nu\bullet, k, j \mid p, q \mid adj]}{[N^\gamma \to M^\gamma P^\gamma \bullet, i, j, \mid p, q \mid \text{false}]}$$

- un paso $\mathcal{D}^{\text{NoDom}}_{\text{CYK}'}$ es equivalente a la secuencia formada por un paso $\mathcal{D}^{\text{Init}}_{\text{ECYK}}$ y dos pasos $\mathcal{D}^{\text{Comp}}_{\text{ECYK}}$:

$$\frac{}{[N^\gamma \to \bullet M^\gamma P^\gamma, i, i, \mid -, - \mid \text{false}]}$$

$$\frac{[N^\gamma \to \bullet M^\gamma P^\gamma, i, i, \mid -, - \mid \text{false}],\ [M^\gamma \to \delta\bullet, i, k \mid -, - \mid adj]}{[N^\gamma \to M^\gamma \bullet P^\gamma, i, k, \mid -, - \mid \text{false}]}$$

$$\frac{[N^\gamma \to M^\gamma \bullet P^\gamma, i, k, \mid -, - \mid \text{false}],\ [P^\gamma \to \nu\bullet, k, j \mid -, - \mid adj]}{[N^\gamma \to M^\gamma P^\gamma \bullet, i, j, \mid -, - \mid \text{false}]}$$

- un paso $\mathcal{D}^{\text{Unary}}_{\text{CYK}'}$ es equivalente a la secuencia formada por un paso $\mathcal{D}^{\text{Init}}_{\text{ECYK}}$ y un paso $\mathcal{D}^{\text{Comp}}_{\text{ECYK}}$:

$$\frac{}{[N^\gamma \to \bullet M^\gamma, i, i, \mid -, - \mid \text{false}]}$$

$$\frac{[N^\gamma \to \bullet M^\gamma, i, i, \mid -, - \mid \text{false}],\ [M^\gamma \to \delta\bullet, i, j \mid -, - \mid adj]}{[N^\gamma \to M^\gamma \bullet, i, j, \ \mid -, - \mid \text{false}]}$$

- $\mathcal{D}^{\text{Adj}}_{\text{ECYK}} = \mathcal{D}^{\text{adj}}_{\text{CYK}'}$.

El esquema de análisis sintáctico **ECYK** está definido para gramáticas de adjunción de árboles en las cuales ningún nodo puede tener más de dos descendientes mientras que el esquema de análisis $\mathbf{buE_1}$ está definido para cualquier TAG. Es fácil mostrar que $\mathbf{ECYK} \stackrel{\text{ext}}{\Longrightarrow} \mathbf{buE_1}$ puesto que $\mathbf{ECYK}(\mathcal{T}) = \mathbf{buE_1}(\mathcal{T})$ es cierto para toda gramática de adjunción de árboles, ya que por definición $\mathbb{P}_{\text{ECYK}} = \mathbb{P}_{\text{buE}_1}$. □

Se puede eliminar el elemento de los ítems que indica si se ha realizado una adjunción en el nodo situado en el lado izquierdo de la producción contenida en dicho ítem, quedando definido el conjunto de ítems

$$\left\{ \begin{array}{ll} [N^\gamma \to \delta \bullet \nu, i, j \mid p, q] \mid & \delta \stackrel{*}{\Rightarrow} a_{i+1} \ldots a_p \, \mathbf{F}^\gamma \, a_{q+1} \ldots a_j \stackrel{*}{\Rightarrow} a_{i+1} \ldots a_j \ \ \text{sii } (p,q) \neq (-,-) \\ & \delta \stackrel{*}{\Rightarrow} a_{i+1} \ldots a_j \qquad\qquad\qquad\qquad\qquad\quad \text{sii } (p,q) = (-,-) \end{array} \right\}$$

Para ello es necesario aplicar un filtro al esquema de análisis sintáctico $\mathbf{buE_1}$, consistente en la contracción de pasos deductivos: puesto que el ítem generado por un paso deductivo de tipo

Adj sólo podrá ser utilizado en un paso de tipo Comp para avanzar el punto en la producción que predijo el no terminal del lado izquierdo de su producción, podemos crear un nuevo tipo AdjComp de paso deductivo y eliminar los pasos de tipo Adj. Obtendremos así el esquema de análisis **buE** cuyo sistema de análisis $\mathbb{P}_{\text{buE}}$ mostramos a continuación.

Esquema de análisis sintáctico 3.3 El sistema de análisis $\mathbb{P}_{\text{buE}}$ que se corresponde con una nueva versión de la extensión del algoritmo de Earley ascendente para TAG, dada una gramática de adjunción de árboles $\mathcal{T}$ y una cadena de entrada $a_1 \dots a_n$ se define como sigue:

$$\mathcal{I}_{\text{buE}} = \left\{ \begin{array}{ll} [N^\gamma \to \delta \bullet \nu, i, j \mid p, q] \mid & N^\gamma \to \delta\nu \in \mathcal{P}(\gamma),\ \gamma \in \boldsymbol{I} \cup \boldsymbol{A}, \\ & 0 \le i \le j,\ (p, q) \le (i, j) \end{array} \right\}$$

$$\mathcal{D}_{\text{buE}}^{\text{Init}} = \frac{}{[N^\gamma \to \bullet\delta, i, i \mid -, -]}$$

$$\mathcal{D}_{\text{buE}}^{\text{Foot}} = \frac{}{[\mathbf{F}^\beta \to \bot\bullet, i, j \mid i, j]}$$

$$\mathcal{D}_{\text{buE}}^{\text{Scan}} = \frac{[N^\gamma \to \delta \bullet a\nu, i, j \mid p, q],\ [a, j, j+1]}{[N^\gamma \to \delta a \bullet \nu, i, j+1 \mid p, q]}$$

$$\mathcal{D}_{\text{buE}}^{\text{Comp}} = \frac{[N^\gamma \to \delta \bullet M^\gamma\nu, i, k \mid p, q],\ [M^\gamma \to \upsilon\bullet, k, j \mid p', q']}{[N^\gamma \to \delta M^\gamma \bullet \nu, i, j \mid p \cup p', q \cup q']} \quad \mathbf{nil} \in \text{adj}(M^\gamma)$$

$$\mathcal{D}_{\text{buE}}^{\text{AdjComp}} = \frac{\begin{array}{l} [\top \to \mathbf{R}^\beta\bullet, k, j \mid l, m], \\ [M^\gamma \to \upsilon\bullet, l, m \mid p', q'], \\ [N^\gamma \to \delta \bullet M^\gamma\nu, i, k \mid p, q], \end{array}}{[N^\gamma \to \delta M^\gamma \bullet \nu, i, j \mid p \cup p', q \cup q']} \quad \beta \in \boldsymbol{A},\ \beta \in \text{adj}(M^\gamma)$$

$$\mathcal{D}_{\text{buE}} = \mathcal{D}_{\text{buE}}^{\text{Init}} \cup \mathcal{D}_{\text{buE}}^{\text{Foot}} \cup \mathcal{D}_{\text{buE}}^{\text{Scan}} \cup \mathcal{D}_{\text{buE}}^{\text{Comp}} \cup \mathcal{D}_{\text{buE}}^{\text{AdjComp}}$$

$$\mathcal{F}_{\text{buE}} = \{\ [\top \to \mathbf{R}^\alpha\bullet, 0, n \mid -, -] \mid \alpha \in \boldsymbol{I}\ \}$$

§

Como se puede observar, el paso $\mathcal{D}_{\text{buE}}^{\text{AdjComp}}$ solamente permite que un árbol auxiliar sea adjuntado en un nodo de un árbol elemental, pues una vez realizada la adjunción, el punto de la producción correspondiente al padre del nodo de adjunción se mueve a la derecha mientras que la producción del nodo de adjunción permanece sin cambios: si posteriormente otro árbol auxiliar es adjuntado en dicho nodo, representará una ambigüedad en el análisis sintáctico de la cadena de entrada, no la adjunción simultánea de dos árboles auxiliares en un mismo nodo [175]. En la figura 3.3 se muestra una representación gráfica de la aplicación de este paso deductivo para su caso más complejo, aquel en el que γ es un árbol auxiliar y el nodo de adjunción pertenece a su espina.

La complejidad temporal del esquema de análisis **buE** con respecto a la cadena de entrada es aparentemente $\mathcal{O}(n^7)$ puesto que son 7 los índices involucrados en el paso deductivo $\mathcal{D}_{\mathrm{buE}}^{\mathrm{AdjComp}}$: i, j, k, l, m y bien p y q o bien p' y q'. Sin embargo, podemos observar que los índices l y m sólo son necesarios para relacionar los dos primeros antecedentes y son irrelevantes para el resto de los ítems involucrados en el paso deductivo. Por tanto, mediante la aplicación parcial o *currificación* del paso deductivo $\mathcal{D}_{\mathrm{buE}}^{\mathrm{AdjComp}}$ la complejidad del mismo se reduce a $\mathcal{O}(n^6)$. De forma equivalente, también se podría incluir dicha aplicación parcial en el propio esquema de análisis, sustituyendo el paso $\mathcal{D}_{\mathrm{buE}}^{\mathrm{AdjComp}}$ por otros dos, uno que combine los dos primeros ítems antecedentes y genere un ítem intermedio y otro que combine dicho ítem con el tercer antecedente del paso original para obtener el ítem resultado. Sin embargo, con ello oscureceríamos la comprensión del algoritmo descrito en el esquema y estaríamos vulnerando la filosofía de los esquemas de análisis, ya que la aplicación parcial es un detalle de implementación del que nos debemos abstraer.

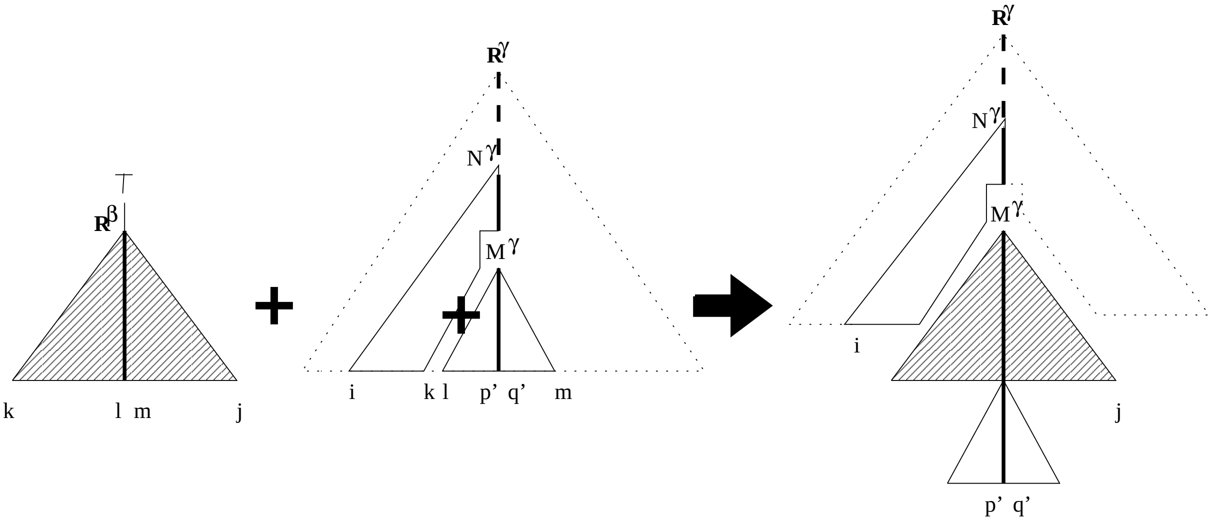

Figura 3.3: Descripción gráfica de un paso $\mathcal{D}_{\mathrm{buE}}^{\mathrm{AdjComp}}$

Proposición 3.2 $\mathbf{buE_1} \overset{\mathbf{df}}{\Longrightarrow} \mathbf{buE'_1} \overset{\mathbf{sc}}{\Longrightarrow} \mathbf{buE_2} \overset{\mathbf{ic}}{\Longrightarrow} \mathbf{buE}$.

Demostración:

Como primer paso definiremos el sistema de análisis $\mathbb{P}_{\mathrm{buE}'_1}$ para una gramática de adjunción $\mathcal{T}$ y una cadena de entrada $a_1 \ldots a_n$, en el cual el paso Comp ha sido desdoblado en Comp^1 y Comp^2, el primero de los cuales se encarga de avanzar el punto en aquellos casos en que el nodo sobre el que se avanza no ha sido utilizado como nodo de adjunción, mientras que el segundo avanza el punto sobre un nodo de adjunción.

$$\mathcal{I}_{\mathrm{buE}'_1} = \mathcal{I}_{\mathrm{buE}_1}$$

$$\mathcal{D}_{\mathrm{buE}'_1}^{\mathrm{Init}} = \mathcal{D}_{\mathrm{buE}_1}^{\mathrm{Init}}$$

$$\mathcal{D}_{\mathrm{buE}'_1}^{\mathrm{Foot}} = \mathcal{D}_{\mathrm{buE}_1}^{\mathrm{Foot}}$$

$$\mathcal{D}_{\mathrm{buE}'_1}^{\mathrm{Scan}} = \mathcal{D}_{\mathrm{buE}_1}^{\mathrm{Scan}}$$

$$\mathcal{D}_{\mathrm{buE}'_1}^{\mathrm{Comp}^1} = \frac{[N^\gamma \to \delta \bullet M^\gamma \nu, i, k \mid p', q' \mid \mathrm{false}],\ [M^\gamma \to \upsilon \bullet, k, j \mid p, q \mid \mathrm{false}]}{[N^\gamma \to \delta M^\gamma \bullet \nu, i, j \mid p' \cup p, q' \cup q \mid \mathrm{false}]} \quad \mathbf{nil} \in \mathrm{adj}(M^\gamma)$$

$$\mathcal{D}_{\mathrm{buE}'_1}^{\mathrm{Comp}^2} = \frac{[N^\gamma \to \delta \bullet M^\gamma \nu, i, k \mid p', q' \mid \mathrm{false}],\ [M^\gamma \to \upsilon \bullet, k, j \mid p, q \mid \mathrm{true}]}{[N^\gamma \to \delta M^\gamma \bullet \nu, i, j \mid p' \cup p, q' \cup q \mid \mathrm{false}]} \quad \exists \beta \in \boldsymbol{A},\ \beta \in \mathrm{adj}(M^\gamma)$$

$$\mathcal{D}^{\text{Adj}}_{\text{buE}_1} = \frac{[\top \to \mathbf{R}^\beta \bullet, k, j \mid k', j' \mid \text{false}],\ [M^\gamma \to \upsilon \bullet, k', j' \mid p, q \mid \text{false}],}{[M^\gamma \to \upsilon \bullet, k, j \mid p, q \mid \text{true}]} \quad \beta \in \boldsymbol{A},\ \beta \in \text{adj}(\gamma)$$

$$\mathcal{D}_{\text{buE}'_1} = \mathcal{D}^{\text{Init}}_{\text{buE}'_1} \cup \mathcal{D}^{\text{Foot}}_{\text{buE}'_1} \cup \mathcal{D}^{\text{Scan}}_{\text{buE}'_1} \cup \mathcal{D}^{\text{Comp}^1}_{\text{buE}'_1} \cup \mathcal{D}^{\text{Comp}^2}_{\text{buE}'_1} \cup \mathcal{D}^{\text{Adj}}_{\text{buE}'_1}$$

$$\mathcal{F}_{\text{buE}'_1} = \mathcal{F}_{\text{buE}_1}$$

Los pasos deductivos $\mathcal{D}^{\text{Comp}^1}_{\text{buE}'_1}$ y $\mathcal{D}^{\text{Comp}^2}_{\text{buE}'_1}$ son el resultado de aplicar un filtro dinámico al paso $\mathcal{D}^{\text{Comp}}_{\text{buE}_1}$. El el caso de $\mathcal{D}^{\text{Comp}^1}_{\text{buE}'_1}$ dicho filtro consiste en comprobar si el último elemento de los ítems contiene el valor false, mientras que en el caso de $\mathcal{D}^{\text{Comp}^2}_{\text{buE}'_1}$ consiste en comprobar que su valor es *true*.

Para demostrar que $\mathbf{buE_1} \stackrel{\mathbf{df}}{\Longrightarrow} \mathbf{buE'_1}$, deberemos demostrar que para todo esquema de análisis $\mathbb{P}_{\text{buE}_1}$ y $\mathbb{P}_{\text{buE}'_1}$ se cumple que $\mathcal{I}_{\text{buE}_1} \supseteq \mathcal{I}_{\text{buE}'_1}$ y $\vdash_{\text{buE}_1} \supseteq \vdash_{\text{buE}'_1}$. Lo primero se cumple puesto que por definición $\mathcal{I}_{\text{buE}_1} = \mathcal{I}_{\text{buE}'_1}$. Para lo segundo debemos mostrar que $\vdash_{\text{buE}_1} \supseteq \mathcal{D}_{\text{buE}'_1}$. Para ello sólo necesitamos considerar aquellos pasos de $\mathbb{P}_{\text{buE}'_1}$ a los que se les ha aplicado el filtro dinámico, pues los restantes pasos permanecen inalterados:

- un paso $\mathcal{D}^{\text{Comp}^1}_{\text{buE}'_1}$ es equivalente a la aplicación del paso $\mathcal{D}^{\text{Comp}}_{\text{buE}_1}$:

$$\frac{[N^\gamma \to \delta \bullet M^\gamma \nu, i, k \mid p', q' \mid \text{false}],\ [M^\gamma \to \upsilon \bullet, k, j \mid p, q \mid \text{false}]}{[N^\gamma \to \delta M^\gamma \bullet \nu, i, j \mid p' \cup p, q' \cup q \mid \text{false}]}$$

- un paso $\mathcal{D}^{\text{Comp}^2}_{\text{buE}'_1}$ es equivalente a la aplicación del paso $\mathcal{D}^{\text{Comp}}_{\text{buE}_1}$:

$$\frac{[N^\gamma \to \delta \bullet M^\gamma \nu, i, k \mid p', q' \mid \text{false}],\ [M^\gamma \to \upsilon \bullet, k, j \mid p, q \mid \text{true}]}{[N^\gamma \to \delta M^\gamma \bullet \nu, i, j \mid p' \cup p, q' \cup q \mid \text{false}]}$$

Sólo se pueden generar ítems que tengan el último componente con valor true como consecuencia de la aplicación de un paso $\mathcal{D}^{\text{Adj}}_{\text{buE}'_1}$ y a su vez estos ítems sólo puede ser utilizados como antecedentes en un paso $\mathcal{D}^{\text{Comp}^2}_{\text{buE}'_1}$. Por tanto podemos juntar ambos pasos pasos en uno sólo. A continuación definimos el sistema de análisis $\mathbb{P}_{\text{buE}_2}$ para una gramática de adjunción $\mathcal{T}$ y una cadena de entrada $a_1 \dots a_n$ que incorpora esta transformación.

$$\mathcal{I}_{\text{buE}_2} = \mathcal{I}_{\text{buE}'_1}$$

$$\mathcal{D}^{\text{Init}}_{\text{buE}_2} = \mathcal{D}^{\text{Init}}_{\text{buE}'_1}$$

$$\mathcal{D}^{\text{Foot}}_{\text{buE}_2} = \mathcal{D}^{\text{Foot}}_{\text{buE}'_1}$$

$$\mathcal{D}^{\text{Scan}}_{\text{buE}_2} = \mathcal{D}^{\text{Scan}}_{\text{buE}'_1}$$

$$\mathcal{D}^{\text{Comp}}_{\text{buE}_2} = \mathcal{D}^{\text{Comp}^1}_{\text{buE}'_1}$$

$$\mathcal{D}^{\text{Adj}}_{\text{buE}_2} = \frac{\begin{array}{l}[\top \to \mathbf{R}^\beta \bullet, k, j \mid k', j' \mid \text{false}],\\ [M^\gamma \to \upsilon \bullet, k', j' \mid p, q \mid \text{false}],\\ [N^\gamma \to \delta \bullet M^\gamma \nu, i, k \mid p', q' \mid \text{false}]\end{array}}{[N^\gamma \to \delta M^\gamma \bullet \nu, i, j \mid p' \cup p, q' \cup q \mid \text{false}]} \quad \beta \in \boldsymbol{A},\ \beta \in \text{adj}(\gamma)$$

$$\mathcal{D}_{\text{buE}_2} = \mathcal{D}^{\text{Init}}_{\text{buE}_2} \cup \mathcal{D}^{\text{Foot}}_{\text{buE}_2} \cup \mathcal{D}^{\text{Scan}}_{\text{buE}_2} \cup \mathcal{D}^{\text{Comp}}_{\text{buE}_2} \cup \mathcal{D}^{\text{Adj}}_{\text{buE}_2}$$

$$\mathcal{F}_{\text{buE}_2} = \mathcal{F}_{\text{buE}_1}$$

Para demostrar que el sistema de análisis sintáctico $\mathbb{P}_{\text{buE}_2}$ es el resultado de aplicar una contracción de pasos al sistema de análisis $\mathbb{P}_{\text{buE}'_1}$ tenemos que demostrar que $\mathcal{I}_{\text{buE}'_1} \supseteq \mathcal{I}_{\text{buE}_2}$ y que $\overset{*}{\vdash}_{\text{buE}'_1} \supseteq \overset{*}{\vdash}_{\text{buE}_2}$. Lo primero es cierto por definición, puesto que $\mathcal{I}_{\text{buE}'_1} = \mathcal{I}_{\text{buE}_2}$. Para lo segundo es suficiente con demostrar que $\overset{*}{\vdash}_{\text{buE}'_1} \supseteq \mathcal{D}_{\text{buE}_2}$. Puesto que los demás pasos deductivos son idénticos en ambos esquemas, debemos mostrar únicamente que $\overset{*}{\vdash}_{\text{buE}'_1} \supseteq \mathcal{D}^{\text{Adj}}_{\text{buE}_2}$, tarea que no presenta complicación alguna puesto que como se ha mencionado anteriormente, un paso $\mathcal{D}^{\text{Adj}}_{\text{buE}_2}$ es equivalente a la aplicación consecutiva de un paso $\mathcal{D}^{\text{Adj}}_{\text{buE}'_1}$ y un paso $\mathcal{D}^{\text{Comp}^2}_{\text{buE}'_1}$:

$$\frac{[\top \to \mathbf{R}^{\beta}\bullet, k, j \mid k', j' \mid \text{false}],\ [M^{\gamma} \to \upsilon\bullet, k', j' \mid p, q \mid \text{false}]}{[M^{\gamma} \to \upsilon\bullet, k, j \mid p, q \mid \text{true}]}$$

$$\frac{[N^{\gamma} \to \delta \bullet M^{\gamma}\nu, i, k \mid p', q' \mid \text{false}],\ [M^{\gamma} \to \upsilon\bullet, k, j \mid p, q \mid \text{true}]}{[N^{\gamma} \to \delta M^{\gamma} \bullet \nu, i, j \mid p' \cup p, q' \cup q \mid \text{false}]}$$

Finalmente observamos que todos los ítems del sistema de análisis $\mathbb{P}_{\text{buE}_2}$ tienen la forma $[N^{\gamma} \to \delta \bullet \nu, i, j \mid p, q \mid \text{false}]$. Puesto que el último elemento tiene un valor constante, su eliminación no producirá ningún efecto con respecto al conjunto de ítems que puedan ser generados. El sistema de análisis $\mathbb{P}_{\text{buE}}$ se obtiene precisamente al aplicar esta transformación al sistema de análisis $\mathbb{P}_{\text{buE}_2}$. Dicha transformación constituye un caso peculiar de contracción de ítems puesto que no se rompe un ítem en varios, sino que se establece una relación biyectiva entre los conjuntos $\mathcal{I}_{\text{buE}_2}$ y $\mathcal{I}_{\text{buE}}$. Vemos que la relación de contracción de ítems se mantiene entre los dos esquemas de análisis puesto que definiendo la función

$$f([N^{\gamma} \to \delta \bullet \nu, i, j \mid p, q \mid \text{false}]) = [N^{\gamma} \to \delta \bullet \nu, i, j \mid p, q]$$

obtenemos que $f(\mathcal{I}_{\text{buE}_2}) = \mathcal{I}_{\text{buE}}$ y $f(\Delta_{\text{buE}_2}) = \Delta_{\text{buE}}$ por inducción en la longitud de las secuencias de derivación. □

3.4 La propiedad del prefijo válido en los algoritmos de análisis

Los analizadores sintácticos que satisfacen la *propiedad del prefijo válido* (VPP) garantizan que, en tanto que leen la cadena de entrada de izquierda a derecha, las subcadenas leídas son prefijos válidos del lenguaje definido por la gramática. A la propiedad del prefijo válido también se la denomina a veces propiedad de detección de errores porque implica que los errores son detectados tan pronto como es posible en una lectura de la cadena de entrada de izquierda a derecha. La carencia de la propiedad del prefijo válido no significa que los errores no sean detectados, sino simplemente que estos lo serán más tarde. En contra de la idea mantenida en ciertos momentos [168], la propiedad del prefijo válido es independiente de la propiedad de análisis *en línea* [169].

Más formalmente, un analizador sintáctico satisface la propiedad del prefijo válido si al leer la subcadena $a_1 \dots a_k$ de la cadena de entrada $a_1 \dots a_k a_{k+1} \dots a_n$ garantiza que hay una cadena $b_1 \dots b_m$, donde b_i no tiene porque formar parte de la cadena de entrada, tal que $a_1 \dots a_k b_1 \dots b_m$ es una cadena válida del lenguaje.

El mantenimiento de la propiedad del prefijo válido exige ir reconociendo los posibles árboles derivados de forma prefija. Este recorrido consta de dos fases, una descendente que dado un nodo expande sus nodos hijos y una ascendente que agrupa los nodos hijos para indicar el reconocimiento del nodo padre. Cuando se desea mantener la propiedad del prefijo válido estas dos fases deben actuar coordinadamente. La fase descendente debe ser además restringida, para evitar expansiones que lleven al reconocimiento de prefijos no válidos [169].

En el caso de gramáticas independientes del contexto, existen numerosos algoritmos de análisis que preservan la propiedad del prefijo válido (por ejemplo, Earley) y que muestran una complejidad en el peor caso igual a aquellos algoritmos que no la preservan (por ejemplo, CYK). Esto se debe a que la operación de sustitución puede ser aplicada de forma totalmente independiente del contexto, lo que conlleva que el conjunto de caminos de una gramática independiente del contexto sea un lenguaje regular. Como consecuencia, el mantenimiento de la propiedad del prefijo válido se puede asegurar sin tener que aplicar restricciones complejas sobre la fase descendente de los algoritmos.

En el caso de las gramáticas de adjunción, la operación de adjunción no es completamente independiente del contexto en el cual se aplica. Durante el reconocimiento de un árbol derivado en forma prefija, la expansión de un nodo puede depender de operaciones de adjunción previamente realizadas en la parte recorrida del árbol. Esta dependencia del contexto conlleva que el conjunto de caminos ya no sea un lenguaje regular sino un lenguaje independiente del contexto [230, 217]. Un algoritmo básicamente ascendente (p.ej.: de tipo CYK, algunas variantes de tipo Earley) puede simplemente hacer uso de una pila para ir almacenando las dependencias indicadas por el lenguaje que define el conjunto de caminos. Con ello se conseguiría un algoritmo de análisis correcto pero sin la propiedad del prefijo válido. Para preservar esta propiedad es necesario disponer de una fase descendente, que también tendría que disponer de una pila para satisfacer las restricciones impuestas por el lenguaje que define el conjunto de caminos. Schabes [169] argumentaba que entonces, al tener que coordinar las pilas de la fase ascendente y descendente, la complejidad del algoritmo de análisis sintáctico resultante aumentaría. Sin embargo, el algoritmo descrito por Nederhof en [125] mantiene la propiedad del prefijo válido con una complejidad $\mathcal{O}(n^6)$, igual a la de aquellos algoritmos que no la mantienen.

3.5 Algoritmos de tipo Earley sin la propiedad del prefijo válido

Mediante la aplicación de un filtrado dinámico a los esquemas de análisis sintácticos anteriores es posible obtener un esquema de análisis sintáctico de un algoritmo al estilo del de Earley pero extendido al caso de gramáticas de adjunción de árboles. Concretamente, el filtrado que se realiza es el siguiente:

- El paso deductivo $\mathcal{D}_{\mathrm{buE}}^{\mathrm{Init}}$ sólo contendrá producciones cuyo lado izquierdo corresponda con la raíz de un árbol inicial.
- Un conjunto de pasos deductivos predictivos controlan la generación de nuevos ítems tratando de limitarla únicamente a aquellos que puedan resultar útiles en el proceso de análisis.

Los algoritmos descritos en esta sección no preservan la propiedad del prefijo válido puesto que la fase predictiva no es lo suficientemente restrictiva como para evitar que las predicciones realizadas durante el análisis del pie de un árbol no conlleven el análisis de subárboles no válidos.

Una primera aproximación a un esquema de análisis sintáctico para un algoritmo de tipo Earley para gramáticas de adjunción consiste en transformar el esquema de análisis sintáctico **buE**. Al nuevo esquema de análisis, que presenta ciertas semejanzas con los algoritmos descritos por Schabes en [168, 169, 172], lo denominaremos **E** y su correspondiente sistema de análisis $\mathbb{P}_{\mathrm{E}}$ se define a continuación.

Esquema de análisis sintáctico 3.4 El sistema de análisis $\mathbb{P}_{\mathrm{E}}$ que se corresponde con una versión del algoritmo de Earley para TAG sin la propiedad del prefijo válido, dada una gramática de adjunción de árboles $\mathcal{T}$ y una cadena de entrada $a_1 \dots a_n$ se define como sigue:

$$\mathcal{I}_{\mathrm{E}} = \mathcal{I}_{\mathrm{buE}}$$

$$\mathcal{D}_{\mathrm{E}}^{\mathrm{Init}} = \frac{}{[\top \to \bullet \mathbf{R}^{\alpha}, 0, 0 \mid -, -]} \quad \alpha \in \boldsymbol{I}$$

$$\mathcal{D}_{\mathrm{E}}^{\mathrm{Scan}} = \mathcal{D}_{\mathrm{buE}}^{\mathrm{Scan}}$$

$$\mathcal{D}_{\mathrm{E}}^{\mathrm{Pred}} = \frac{[N^\gamma \to \delta \bullet M^\gamma \nu, i, j \mid p, q]}{[M^\gamma \to \bullet \upsilon, j, j \mid -, -]} \quad \mathbf{nil} \in \mathrm{adj}(M^\gamma)$$

$$\mathcal{D}_{\mathrm{E}}^{\mathrm{Comp}} = \mathcal{D}_{\mathrm{buE}}^{\mathrm{Comp}}$$

$$\mathcal{D}_{\mathrm{E}}^{\mathrm{AdjPred}} = \frac{[N^\gamma \to \delta \bullet M^\gamma \nu, i, j \mid p, q]}{[\top \to \bullet \mathbf{R}^\beta, j, j \mid -, -]} \quad \beta \in \mathrm{adj}(M^\gamma)$$

$$\mathcal{D}_{\mathrm{E}}^{\mathrm{FootPred}} = \frac{[\mathbf{F}^\beta \to \bullet \bot, k, k \mid -, -]}{[M^\gamma \to \bullet \delta, k, k \mid -, -]} \quad \beta \in \mathrm{adj}(M^\gamma)$$

$$\mathcal{D}_{\mathrm{E}}^{\mathrm{FootComp}} = \frac{[M^\gamma \to \delta \bullet, k, l \mid p, q],\ [\mathbf{F}^\beta \to \bullet \bot, k, k \mid -, -]}{[\mathbf{F}^\beta \to \bot \bullet, k, l \mid k, l]} \quad \beta \in \mathrm{adj}(M^\gamma)$$

$$\mathcal{D}_{\mathrm{E}}^{\mathrm{AdjComp}} = \mathcal{D}_{\mathrm{buE}}^{\mathrm{AdjComp}} = \frac{\begin{array}{l}[\top \to \mathbf{R}^\beta \bullet, j, m \mid k, l],\\ [M^\gamma \to \upsilon \bullet, k, l \mid p, q],\\ [N^\gamma \to \delta \bullet M^\gamma \nu, i, j \mid p', q']\end{array}}{[N^\gamma \to \delta M^\gamma \bullet \nu, i, m \mid p \cup p', q \cup q']} \quad \beta \in \mathrm{adj}(M^\gamma)$$

$$\mathcal{D}_{\mathrm{E}} = \mathcal{D}_{\mathrm{E}}^{\mathrm{Init}} \cup \mathcal{D}_{\mathrm{E}}^{\mathrm{Scan}} \cup \mathcal{D}_{\mathrm{E}}^{\mathrm{Pred}} \cup \mathcal{D}_{\mathrm{E}}^{\mathrm{Comp}} \cup \mathcal{D}_{\mathrm{E}}^{\mathrm{AdjPred}} \cup \mathcal{D}_{\mathrm{E}}^{\mathrm{FootPred}} \cup \mathcal{D}_{\mathrm{E}}^{\mathrm{FootComp}} \cup \mathcal{D}_{\mathrm{E}}^{\mathrm{AdjComp}}$$

$$\mathcal{F}_{\mathrm{E}} = \mathcal{F}_{\mathrm{buE}}$$

§

El paso $\mathcal{D}_{\mathrm{E}}^{\mathrm{AdjComp}}$, aunque es idéntico al del sistema de análisis $\mathbb{P}_{\mathrm{buE}_1}$ se repite aquí para facilitar la comprensión del algoritmo. Como se ha dicho anteriormente, el esquema de análisis **E** presenta ciertas similitudes con los algoritmos descritos por Schabes en [168, 169, 172]. Podemos establecer la siguiente relación entre los pasos deductivos de **E** y los procesos definidos en dichos algoritmos, tal y como se muestra en la tabla 3.1. La razón del cambio de nombre en las pasos encargados de la adjunción se debe a que creemos que los nombres utilizados por Schabes pueden llevar a confusión, pues se tiende a pensar que cada *Predictor* está asociado con el *Completor* correspondiente a su mismo lado, cuando no es así. Es por ello que hemos decidido emparejar los pasos deductivos por las funciones que realizan: predicción-compleción de adjunción y predicción-compleción de pie.

Respecto al comportamiento del algoritmo, diremos que el análisis comienza creando un ítem correspondiente a una producción del nodo raíz de un árbol inicial con el punto situado en el extremo izquierdo. Posteriormente, los pasos $\mathcal{D}_{\mathrm{E}}^{\mathrm{Pred}}$ y $\mathcal{D}_{\mathrm{E}}^{\mathrm{Comp}}$ se encargan de ir recorriendo el árbol de modo descendente y ascendente, respectivamente, de modo similar a como actúa el algoritmo de Earley para gramáticas independientes del contexto. Se puede predecir la adjunción de un árbol β en un nodo de un árbol elemental γ mediante la aplicación de un paso $\mathcal{D}_{\mathrm{E}}^{\mathrm{AdjPred}}$,

Pasos deductivos de E	**Algoritmo de Schabes**
$\mathcal{D}_{\mathrm{E}}^{\mathrm{Init}}$	*Initial item*
$\mathcal{D}_{\mathrm{E}}^{\mathrm{Scan}}$	Scanner
$\mathcal{D}_{\mathrm{E}}^{\mathrm{Pred}}$	Move Dot Down
$\mathcal{D}_{\mathrm{E}}^{\mathrm{Comp}}$	Move Dot Up
$\mathcal{D}_{\mathrm{E}}^{\mathrm{AdjPred}}$	Left Predictor
$\mathcal{D}_{\mathrm{E}}^{\mathrm{FootPred}}$	Left Completor
$\mathcal{D}_{\mathrm{E}}^{\mathrm{FootComp}}$	Right Predictor
$\mathcal{D}_{\mathrm{E}}^{\mathrm{AdjComp}}$	Right Completor

Tabla 3.1: Relación entre $\mathcal{D}_{\mathrm{E}}$ y el algoritmo tipo Earley sin VPP de Schabes

con lo que se comienza el análisis del árbol β. Una vez alcanzado el pie de dicho árbol auxiliar, deberemos retomar el análisis de γ, concretamente del subárbol que pende del nodo de adjunción. El problema es que al no conocer en qué nodo de qué árbol elemental se ha producido la adjunción, deberemos predecir todos los posibles nodos donde esté permitida la adjunción de β, predicción realizada por un paso deductivo $\mathcal{D}_{\mathrm{E}}^{\mathrm{FootPred}}$. Es la predicción que se realiza en los pasos $\mathcal{D}_{\mathrm{E}}^{\mathrm{FootPred}}$ lo que provoca que el algoritmo no posea la propiedad del prefijo válido, puesto que se puede comenzar a analizar una parte de la cadena de entrada que no es gramatical, al predecir un subárbol que no se corresponde con el árbol desde el que se realizó la adjunción.

Una vez terminado de analizar todo el subárbol predicho por un paso $\mathcal{D}_{\mathrm{E}}^{\mathrm{FootPred}}$, deberemos retomar el análisis del árbol auxiliar β a partir del pie, tarea encomendada a los pasos deductivos $\mathcal{D}_{\mathrm{E}}^{\mathrm{FootComp}}$. Una vez terminado de analizar completamente el árbol auxiliar β, deberemos concluir la operación de adjunción aplicando un paso $\mathcal{D}_{\mathrm{E}}^{\mathrm{AdjComp}}$. Es en estos pasos en los que se verifica que el subárbol escindido del nodo de adjunción y el árbol auxiliar han sido correctamente reconstruidos. Las adjunciones simultáneas sobre un mismo nodo [175] son evitadas por los pasos $\mathcal{D}_{\mathrm{E}}^{\mathrm{AdjComp}}$ puesto que cuando se ha terminado de recorrer completamente el árbol auxiliar, se verifica que se ha analizado la parte correspondiente al subárbol del nodo de adjunción y se avanza el punto de la producción que contiene a este, sin cambiar el ítem correspondiente al nodo de adjunción. Si posteriormente otro árbol auxiliar es adjuntado en dicho nodo, representará una ambigüedad en el análisis sintáctico de la cadena de entrada pero no la adjunción simultánea de dos árboles auxiliares en un mismo nodo.

Proposición 3.3 **buE** $\stackrel{\mathrm{df}}{\Longrightarrow}$ **E**.

Demostración:

Para demostrar que el esquema de análisis sintáctico **E** es el resultado de aplicar un filtrado dinámico al esquema de análisis **buE**, debemos demostrar que $\mathcal{I}_{\mathrm{buE}} \supseteq \mathcal{I}_{\mathrm{E}}$ y que $\vdash_{\mathrm{buE}} \supseteq \vdash_{\mathrm{E}}$ para los sistemas de análisis sintáctico $\mathbb{P}_{\mathrm{buE}}$ y $\mathbb{P}_{\mathrm{E}}$. Lo primero es cierto por definición puesto que $\mathcal{I}_{\mathrm{buE}} = \mathcal{I}_{\mathrm{E}}$. Respecto a lo segundo, es suficiente con mostrar que $\vdash_{\mathrm{buE}} \supseteq \mathcal{D}_{\mathrm{E}}$.

Los pasos $\mathcal{D}_{\mathrm{E}}^{\mathrm{Scan}}$, $\mathcal{D}_{\mathrm{E}}^{\mathrm{Comp}}$ y $\mathcal{D}_{\mathrm{E}}^{\mathrm{Adjcomp}}$ son idénticos a sus homónimos del sistema $\mathbb{P}_{\mathrm{buE}}$ y los pasos $\mathcal{D}_{\mathrm{E}}^{\mathrm{Init}}$ generan un subconjunto de los ítems generados por $\mathcal{D}_{\mathrm{buE}}^{\mathrm{Init}}$. Respecto a los otros pasos:

- Dado un paso $\frac{[N^\gamma\to\delta\bullet M^\gamma\nu,i,j\mid p,q]}{[M^\gamma\to\bullet\upsilon,j,j\mid -,-]} \in \mathcal{D}_{\mathrm{E}}^{\mathrm{Pred}}$ existe un paso $\frac{}{[M^\gamma\to\bullet\upsilon,j,j\mid -,-]} \in \mathcal{D}_{\mathrm{buE}}^{\mathrm{Init}}$ y por tanto existe la inferencia
$$[N^\gamma \to \delta \bullet M^\gamma\nu, i, j \mid p, q] \vdash_{\mathrm{buE}} [M^\gamma \to \bullet\upsilon, j, j \mid -, -]$$
- Dado un paso $\frac{[N^\gamma\to\delta\bullet M^\gamma\nu,i,j\mid p,q]}{[\top\to\bullet\mathbf{R}^\beta,j,j\mid -,-]} \in \mathcal{D}_{\mathrm{E}}^{\mathrm{AdjPred}}$ existe un paso $\frac{}{[\top\to\bullet\mathbf{R}^\beta,j,j\mid -,-]} \in \mathcal{D}_{\mathrm{buE}}^{\mathrm{Init}}$ y por tanto existe la inferencia
$$[N^\gamma \to \delta \bullet M^\gamma\nu, i, j \mid p, q] \vdash_{\mathrm{buE}} [\top \to \bullet\mathbf{R}^\beta, j, j \mid -, -]$$
- Dado un paso $\frac{[\mathbf{F}^\beta\to\bullet\bot,k,k\mid -,-]}{[M^\gamma\to\bullet\delta,k,k\mid -,-]} \in \mathcal{D}_{\mathrm{E}}^{\mathrm{FootPred}}$ existe un paso $\frac{}{[M^\gamma\to\bullet\delta,k,k\mid -,-]} \in \mathcal{D}_{\mathrm{buE}}^{\mathrm{Init}}$ y por tanto existe la inferencia
$$[\mathbf{F}^\beta \to \bullet\bot, k, k \mid -, -] \vdash_{\mathrm{buE}} [M^\gamma \to \bullet\delta, k, k \mid -, -]$$
- Dado un paso $\frac{[M^\gamma\to\delta\bullet,k,l\mid p,q],\ [\mathbf{F}^\beta\to\bullet\bot,k,k\mid -,-]}{[\mathbf{F}^\beta\to\bot\bullet,k,l\mid k,l]} \in \mathcal{D}_{\mathrm{E}}^{\mathrm{FootComp}}$ existe un paso $\frac{}{[\mathbf{F}^\beta\to\bot\bullet,k,l\mid k,l]} \in \mathcal{D}_{\mathrm{buE}}^{\mathrm{Foot}}$ y por tanto existe la inferencia
$$[M^\gamma \to \delta\bullet, k, l \mid p, q],\ [\mathbf{F}^\beta \to \bullet\bot, k, k \mid -, -] \vdash_{\mathrm{buE}} [\mathbf{F}^\beta \to \bot\bullet, k, l \mid k, l]$$

□

La complejidad temporal del esquema de análisis sintáctico **E** con respecto a la cadena de entrada es $\mathcal{O}(n^6)$ puesto que la aparente complejidad $\mathcal{O}(n^7)$ del paso deductivo $\mathcal{D}_{\mathrm{E}}^{\mathrm{AdjComp}}$ puede reducirse a $\mathcal{O}(n^6)$ mediante la aplicación parcial o *currificación* de dicho paso, ya que los índices l y m sólo involucran a los dos primeros ítems antecedentes.

Con el fin de definir un esquema de análisis sintáctico que se corresponda con un algoritmo más cercano al espíritu del algoritmo de Earley, debemos fortalecer la fase predictiva de los esquemas de análisis anteriores, puesto que estos no utilizan toda la información que tienen a su disposición. En concreto:

- Los pasos deductivos $\mathcal{D}_{\mathrm{E}}^{\mathrm{FootPred}}$ en los que se realiza la predicción del pie no comprueban que previamente se haya iniciado la adjunción del árbol β en el nodo M^γ.
- Idem para los pasos deductivos $\mathcal{D}_{\mathrm{E}}^{\mathrm{FootComp}}$ que finalizan el reconocimiento del pie.
- Los pasos deductivos $\mathcal{D}_{\mathrm{E}}^{\mathrm{FootComp}}$ no comprueban que el árbol auxiliar predicho para adjunción en el nodo M^γ sea el mismo que el que ha provocado el reconocimiento del subárbol enraizado en dicho nodo mediante la aplicación de los pasos FootPred.

A continuación definimos un nuevo esquema de análisis **Ear** derivado a partir de **E**, sobre el que hemos aplicado las siguientes modificaciones:

- La aplicación de un filtro dinámico a los pasos deductivos FootPred y FootComp consistente en la verificación de la existencia de los ítems que representan el comienzo de la operación de adjunción en el nodo M^γ.
- La aplicación de un refinamiento al paso AdjComp, que se divide en dos pasos AdjComp^1 y AdjComp^2, el primero realizando las comprobaciones pertinentes en el caso de que se haya producido la adjunción de un árbol auxiliar en un nodo de la espina de otro árbol auxiliar. En la figura 3.4 se muestra una representación gráfica de la aplicación del paso deductivo AdjComp^1, el más complejo de los dos ya que el árbol γ en el que se realiza la adjunción es un árbol auxiliar y el nodo de adjunción pertenece a su espina. Las partes de

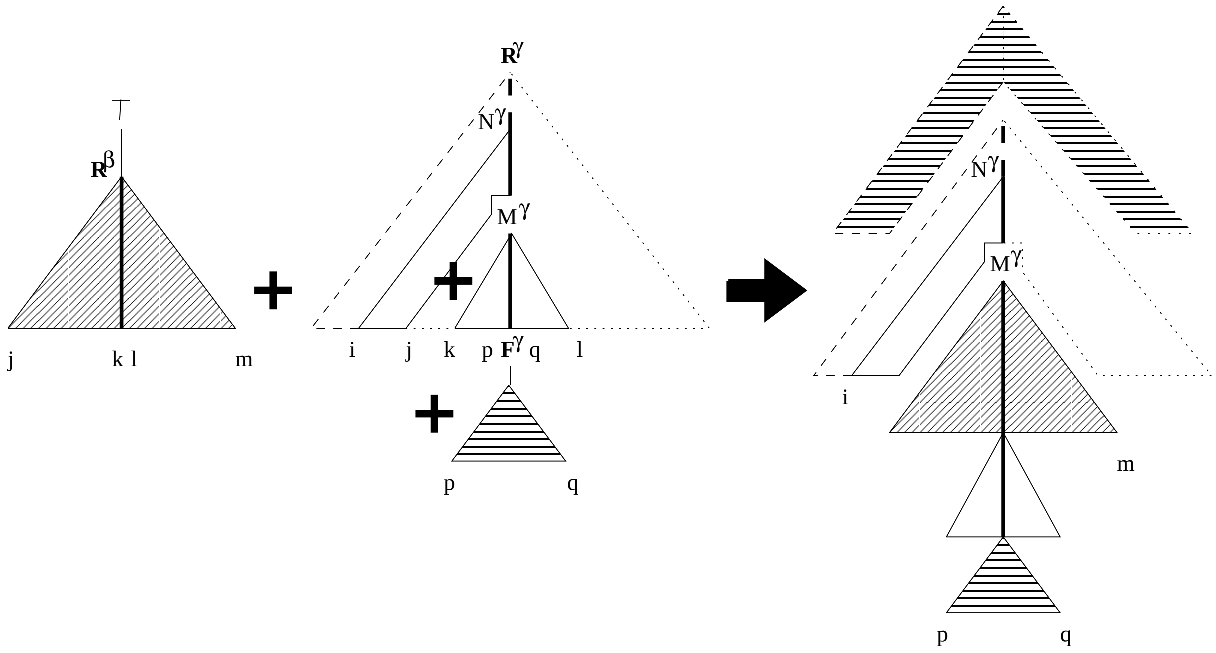

Figura 3.4: Descripción gráfica de un paso $\mathcal{D}_{\text{Ear}}^{\text{AdjComp}^1}$

los árboles involucrados que no se encuentran representados por los ítems que intervienen en el paso deductivo se muestran en línea discontinua si el algoritmo de análisis tiene que haber pasado obligatoriamente por dicha parte del árbol al menos en la fase predictiva y en línea punteada si se trata de partes que serán analizadas posteriormente.

Esquema de análisis sintáctico 3.5 El sistema de análisis $\mathbb{P}_{\text{Ear}}$ que se corresponde con una versión del algoritmo de Earley sin la propiedad del prefijo válido con predicción fuerte, dada una gramática de adjunción de árboles $\mathcal{T}$ y una cadena de entrada $a_1 \dots a_n$ se define como sigue:

$$\mathcal{I}_{\text{Ear}} = \mathcal{I}_{\text{buE}}$$

$$\mathcal{D}_{\text{Ear}}^{\text{Init}} = \mathcal{D}_{\text{E}}^{\text{Init}}$$

$$\mathcal{D}_{\text{Ear}}^{\text{Scan}} = \mathcal{D}_{\text{buE}}^{\text{Scan}}$$

$$\mathcal{D}_{\text{Ear}}^{\text{Pred}} = \mathcal{D}_{\text{E}}^{\text{Pred}}$$

$$\mathcal{D}_{\text{Ear}}^{\text{Comp}} = \mathcal{D}_{\text{buE}}^{\text{Comp}}$$

$$\mathcal{D}_{\text{Ear}}^{\text{AdjPred}} = \mathcal{D}_{\text{E}}^{\text{AdjPred}}$$

$$\mathcal{D}_{\text{Ear}}^{\text{FootPred}} = \frac{[\mathbf{F}^\beta \to \bullet\bot, k, k \mid -, -],\ [N^\gamma \to \delta \bullet M^\gamma \nu, i, j \mid p, q]}{[M^\gamma \to \bullet\delta, k, k \mid -, -]} \quad \beta \in \text{adj}(M^\gamma)$$

$$\mathcal{D}_{\text{Ear}}^{\text{FootComp}} = \frac{\begin{array}{l}[M^\gamma \to \upsilon\bullet, k, l \mid p, q],\\ [\mathbf{F}^\beta \to \bullet\bot, k, k \mid -, -],\\ [N^\gamma \to \delta \bullet M^\gamma \nu, i, j \mid p', q']\end{array}}{[\mathbf{F}^\beta \to \bot\bullet, k, l \mid k, l]} \quad \begin{array}{l}\beta \in \text{adj}(M^\gamma),\\ p \cup p' \textit{ y } q \cup q' \textit{ está definido}\end{array}$$

$$\mathcal{D}_{\text{Ear}}^{\text{AdjComp}^1} = \frac{\begin{array}{l}[\top \to \mathbf{R}^\beta \bullet, j, m \mid k, l],\\ [M^\gamma \to \upsilon \bullet, k, l \mid p, q],\\ [\mathbf{F}^\gamma \to \bot \bullet, p, q \mid p, q],\\ [N^\gamma \to \delta \bullet M^\gamma \nu, i, j \mid -, -]\end{array}}{[N^\gamma \to \delta M^\gamma \bullet \nu, i, m \mid p, q]} \quad \beta \in \text{adj}(M^\gamma)$$

$$\mathcal{D}_{\text{Ear}}^{\text{AdjComp}^2} = \frac{\begin{array}{l}[\top \to \mathbf{R}^\beta \bullet, j, m \mid k, l],\\ [M^\gamma \to \upsilon \bullet, k, l \mid -, -],\\ [N^\gamma \to \delta \bullet M^\gamma \nu, i, j \mid p', q']\end{array}}{[N^\gamma \to \delta M^\gamma \bullet \nu, i, m \mid p', q']} \quad \beta \in \text{adj}(M^\gamma)$$

$$\mathcal{D}_{\text{Ear}} = \mathcal{D}_{\text{Ear}}^{\text{Init}} \cup \mathcal{D}_{\text{Ear}}^{\text{Scan}} \cup \mathcal{D}_{\text{Ear}}^{\text{Pred}} \cup \mathcal{D}_{\text{Ear}}^{\text{Comp}} \cup \mathcal{D}_{\text{Ear}}^{\text{AdjPred}} \cup \mathcal{D}_{\text{Ear}}^{\text{FootPred}} \cup \mathcal{D}_{\text{Ear}}^{\text{FootComp}} \cup \mathcal{D}_{\text{Ear}}^{\text{AdjComp}^1} \cup \mathcal{D}_{\text{Ear}}^{\text{AdjComp}^2}$$

$$\mathcal{F}_{\text{Ear}} = \mathcal{F}_{\text{buE}}$$

§

Proposición 3.4 $\mathbf{E} \stackrel{\text{sr}}{\Longrightarrow} \mathbf{E}' \stackrel{\text{df}}{\Longrightarrow} \mathbf{Ear}$.

<u>Demostración</u>:

Como primer paso definiremos el esquema de análisis $\mathbf{E}'$ que se obtiene a partir de $\mathbf{E}$ simplemente rompiendo el conjunto de pasos deductivos $\mathcal{D}_{\text{E}}^{\text{AdjComp}}$ en dos conjuntos $\mathcal{D}_{\text{E}'}^{\text{AdjComp}^1}$ y $\mathcal{D}_{\text{E}'}^{\text{AdjComp}^2}$. El sistema de análisis $\mathbb{P}_{\text{E}'}$ sería por tanto el siguiente:

$$\mathcal{I}_{\text{E}'} = \mathcal{I}_{\text{buE}}$$

$$\mathcal{D}_{\text{E}'}^{\text{Init}} = \mathcal{D}_{\text{E}}^{\text{Init}}$$

$$\mathcal{D}_{\text{E}'}^{\text{Scan}} = \mathcal{D}_{\text{buE}}^{\text{Scan}}$$

$$\mathcal{D}_{\text{E}'}^{\text{Pred}} = \mathcal{D}_{\text{E}}^{\text{Pred}}$$

$$\mathcal{D}_{\text{E}'}^{\text{Comp}} = \mathcal{D}_{\text{buE}}^{\text{Comp}}$$

$$\mathcal{D}_{\text{E}'}^{\text{AdjPred}} = \mathcal{D}_{\text{E}}^{\text{AdjPred}}$$

$$\mathcal{D}_{\text{E}'}^{\text{FootPred}} = \mathcal{D}_{\text{E}}^{\text{FootPred}}$$

$$\mathcal{D}_{\text{E}'}^{\text{FootComp}} = \mathcal{D}_{\text{E}}^{\text{FootComp}}$$

$$\mathcal{D}_{\text{E}'}^{\text{AdjComp}^1} = \frac{\begin{array}{l}[\top \to \mathbf{R}^\beta \bullet, j, m \mid k, l],\\ [M^\gamma \to \upsilon \bullet, k, l \mid p, q],\\ [N^\gamma \to \delta \bullet M^\gamma \nu, i, j \mid -, -]\end{array}}{[N^\gamma \to \delta M^\gamma \bullet \nu, i, m \mid p, q]} \quad \beta \in \text{adj}(M^\gamma)$$

$$\mathcal{D}_{\text{E}'}^{\text{AdjComp}^2} = \frac{\begin{array}{l}[\top \to \mathbf{R}^\beta \bullet, j, m \mid k, l],\\ [M^\gamma \to \upsilon \bullet, k, l \mid -, -],\\ [N^\gamma \to \delta \bullet M^\gamma \nu, i, j \mid p', q']\end{array}}{[N^\gamma \to \delta M^\gamma \bullet \nu, i, m \mid p', q']} \quad \beta \in \text{adj}(M^\gamma)$$

$$\mathcal{D}_{\text{E}'} = \mathcal{D}_{\text{E}'}^{\text{Init}} \cup \mathcal{D}_{\text{E}'}^{\text{Scan}} \cup \mathcal{D}_{\text{E}'}^{\text{Pred}} \cup \mathcal{D}_{\text{E}'}^{\text{Comp}} \cup \mathcal{D}_{\text{E}'}^{\text{AdjPred}} \cup \mathcal{D}_{\text{E}'}^{\text{FootPred}} \cup \mathcal{D}_{\text{E}'}^{\text{FootComp}} \cup \mathcal{D}_{\text{E}'}^{\text{AdjComp}^1} \cup \mathcal{D}_{\text{E}'}^{\text{AdjComp}^2}$$

$$\mathcal{F}_{\mathrm{E}'} = \mathcal{F}_{\mathrm{buE}}$$

Las condiciones a verificar son que $\mathcal{I}_{\mathrm{E}} \subseteq \mathcal{I}_{\mathrm{E}'}$ y que $\overset{*}{\vdash}_{\mathrm{E}} \subseteq \overset{*}{\vdash}_{\mathrm{E}'}$. La primera condición se verifica por la propia definición de los ítems mientras que la segunda se obtiene directamente considerando que el único cambio que se ha producido en $\mathbb{P}_{\mathrm{E}'}$ es hacer explícita la incompatibilidad del par de índices (p,q) con el par (p',q') de los pasos $\mathcal{D}_{\mathrm{E}}^{\mathrm{AdjComp}}$: si son p' y q' los índices que están definidos, entonces el paso $\mathcal{D}_{\mathrm{E}}^{\mathrm{AdjComp}}$ se convierte en $\mathcal{D}_{\mathrm{E}'}^{\mathrm{AdjComp}^1}$, mientras que si son p y q los índices que están definidos, entonces el paso $\mathcal{D}_{\mathrm{E}}^{\mathrm{AdjComp}}$ se convierte en $\mathcal{D}_{\mathrm{E}'}^{\mathrm{AdjComp}^2}$.

Para demostrar que el esquema de análisis sintáctico **Ear** es el resultado de aplicar un filtrado dinámico al esquema de análisis **E**$'$, debemos demostrar que $\mathcal{I}_{\mathrm{E}'} \supseteq \mathcal{I}_{\mathrm{Ear}}$ y que $\vdash_{\mathrm{E}'} \supseteq \vdash_{\mathrm{Ear}}$ para los sistemas de análisis sintáctico $\mathbb{P}_{\mathrm{E}'}$ y $\mathbb{P}_{\mathrm{Ear}}$. Lo primero es cierto por definición puesto que $\mathcal{I}_{\mathrm{buE}} = \mathcal{I}_{\mathrm{E}'} = \mathcal{I}_{\mathrm{Ear}}$. Respecto a lo segundo es suficiente con mostrar que $\vdash_{\mathrm{E}'} \supseteq \vdash_{\mathrm{Ear}}$.

Los pasos $\mathcal{D}_{\mathrm{Ear}}^{\mathrm{Init}}$, $\mathcal{D}_{\mathrm{Ear}}^{\mathrm{Scan}}$, $\mathcal{D}_{\mathrm{Ear}}^{\mathrm{pred}}$, $\mathcal{D}_{\mathrm{Ear}}^{\mathrm{Comp}}$ y $\mathcal{D}_{\mathrm{Ear}}^{\mathrm{AdjPred}}$ son idénticos a sus homónimos del sistema $\mathbb{P}_{\mathrm{E}'}$. Respecto a los otros pasos:

$\mathcal{D}_{\mathrm{Ear}}^{\mathrm{FootPred}}$: Dado un paso $\frac{[\mathbf{F}^\beta \to \bullet\bot, k, k \mid -,-],\ [N^\gamma \to \delta \bullet M^\gamma \nu, i, j \mid p, q]}{[M^\gamma \to \bullet\delta, k, k \mid -,-]}$ existe un paso $\frac{[\mathbf{F}^\beta \to \bullet\bot, k, k \mid -,-]}{[M^\gamma \to \bullet\delta, k, k \mid -,-]} \in \mathcal{D}_{\mathrm{E}'}^{\mathrm{FootPred}}$ y por tanto existe la inferencia

$$[\mathbf{F}^\beta \to \bullet\bot, k, k \mid -,-],\ [N^\gamma \to \delta \bullet M^\gamma \nu, i, j \mid p, q] \vdash_{\mathrm{E}'} [M^\gamma \to \bullet\delta, k, k \mid -,-]$$

$\mathcal{D}_{\mathrm{Ear}}^{\mathrm{FootComp}}$: Dado un paso $\frac{[M^\gamma \to \upsilon\bullet, k, l \mid p, q],\ [\mathbf{F}^\beta \to \bullet\bot, k, k \mid -,-],[N^\gamma \to \delta \bullet M^\gamma \nu, i, j \mid p', q']}{[\mathbf{F}^\beta \to \bot\bullet, k, l \mid k, l]}$ existe un paso $\frac{[M^\gamma \to \upsilon\bullet, k, l \mid p, q],\ [\mathbf{F}^\beta \to \bullet\bot, k, k \mid -,-]}{[\mathbf{F}^\beta \to \bot\bullet, k, l \mid k, l]} \in \mathcal{D}_{\mathrm{E}'}^{\mathrm{FootComp}}$ y por tanto existe la inferencia

$$\begin{array}{l}[M^\gamma \to \upsilon\bullet, k, l \mid p, q],\ [\mathbf{F}^\beta \to \bullet\bot, k, k \mid -,-],\ [N^\gamma \to \delta \bullet M^\gamma \nu, i, j \mid p', q'] \\ \vdash_{\mathrm{E}'} [\mathbf{F}^\beta \to \bot\bullet, k, l \mid k, l]\end{array}$$

$\mathcal{D}_{\mathrm{Ear}}^{\mathrm{AdjComp}^1}$: Dado un paso $\frac{[\top \to \mathbf{R}^\beta\bullet, j, m \mid k, l],\ [M^\gamma \to \upsilon\bullet, k, l \mid p, q],\ [\mathbf{F}^\gamma \to \bot\bullet, p, q \mid p, q],\ [N^\gamma \to \delta \bullet M^\gamma \nu, i, j \mid -,-]}{[N^\gamma \to \delta M^\gamma \bullet \nu, i, m \mid p, q]}$ existe un paso $\frac{[\top \to \mathbf{R}^\beta\bullet, j, m \mid k, l],\ [M^\gamma \to \upsilon\bullet, k, l \mid p, q],\ [N^\gamma \to \delta \bullet M^\gamma \nu, i, j \mid -,-]}{[N^\gamma \to \delta M^\gamma \bullet \nu, i, m \mid p, q]} \in \mathcal{D}_{\mathrm{E}'}^{\mathrm{AdjComp}}$ y por tanto existe la inferencia

$$\begin{array}{l}[\top \to \mathbf{R}^\beta\bullet, j, m \mid k, l],\ [M^\gamma \to \upsilon\bullet, k, l \mid p, q],\ [\mathbf{F}^\gamma \to \bot\bullet, p, q \mid p, q],\ [N^\gamma \to \delta \bullet M^\gamma \nu, i, j \mid -,-] \\ \vdash_{\mathrm{E}'} [N^\gamma \to \delta M^\gamma \bullet \nu, i, m \mid p, q]\end{array}$$

$\mathcal{D}_{\mathrm{Ear}}^{\mathrm{AdjComp}^2}$: Dado un paso $\frac{[\top \to \mathbf{R}^\beta\bullet, j, m \mid k, l],\ [M^\gamma \to \upsilon\bullet, k, l \mid -,-],\ [N^\gamma \to \delta \bullet M^\gamma \nu, i, j \mid p, q]}{[N^\gamma \to \delta M^\gamma \bullet \nu, i, m \mid p', q']}$ existe un paso $\frac{[\top \to \mathbf{R}^\beta\bullet, j, m \mid k, l],\ [M^\gamma \to \upsilon\bullet, k, l \mid -,-],\ [N^\gamma \to \delta \bullet M^\gamma \nu, i, j \mid p', q']}{[N^\gamma \to \delta M^\gamma \bullet \nu, i, m \mid p', q']} \in \mathcal{D}_{\mathrm{E}'}^{\mathrm{AdjComp}}$ y por tanto existe la inferencia

$$\begin{array}{l}[\top \to \mathbf{R}^\beta\bullet, j, m \mid k, l],\ [M^\gamma \to \upsilon\bullet, k, l \mid -,-],\ [N^\gamma \to \delta \bullet M^\gamma \nu, i, j \mid p', q'] \\ \vdash_{\mathrm{E}'} [N^\gamma \to \delta M^\gamma \bullet \nu, i, m \mid p', q']\end{array}$$

$\square$

La complejidad temporal del esquema de análisis sintáctico **Ear** con respecto a la longitud n de la cadena de entrada es $\mathcal{O}(n^6)$ puesto que la aparente complejidad $\mathcal{O}(n^7)$ de los pasos deductivos $\mathcal{D}_{\mathrm{Ear}}^{\mathrm{AdjComp}^1}$ y $\mathcal{D}_{\mathrm{Ear}}^{\mathrm{AdjComp}^1}$ puede reducirse a $\mathcal{O}(n^6)$ mediante la aplicación parcial o *currificación* de dichos pasos, puesto que el índice l sólo involucra a los dos primeros ítems antecedentes en cada uno de ellos.

3.6 Algoritmos de tipo Earley con la propiedad del prefijo válido

El primer algoritmo de análisis sintáctico para TAG que satisfacía la propiedad del prefijo válido fue el descrito por Schabes y Joshi en [173] y por Schabes en [168]. La principal particularidad de dicho algoritmo es que su complejidad temporal con respecto a la cadena de entrada es $\mathcal{O}(n^7)$, tal como muestran Díaz Madrigal et al. en [65], mientras que los algoritmos sin la propiedad del prefijo válido presentan una complejidad $\mathcal{O}(n^6)$. Durante mucho tiempo cobró fuerza la opinión de que aquellos algoritmos que cumpliesen la propiedad del prefijo válido deberían tener una complejidad más alta que aquellos que no la cumpliesen. Sin embargo, Nederhof presentó en [125] un algoritmo para el análisis de TAG que cumple la propiedad del prefijo válido[6] y que presenta una complejidad temporal $\mathcal{O}(n^6)$. Veremos que dicho algoritmo es fácilmente derivable a partir del esquema de análisis **Ear** presentado en la sección anterior, correspondiente al algoritmo de tipo Earley sin la propiedad del prefijo válido.

El esquema de análisis sintáctico **Ear** describe un algoritmo que no cumple la propiedad del prefijo válido porque los pasos deductivos que se encargan de reconocer el nodo correspondiente al pie de un árbol auxiliar no pueden verificar la contigüidad de las fronteras del árbol al que pertenece el nodo de adjunción y del árbol auxiliar. Analicemos detalladamente dichos pasos:

- El paso deductivo $\mathcal{D}_{\text{Ear}}^{\text{FootPred}}$ puede verificar, mediante el ítem $[N^\gamma \to \delta \bullet M^\gamma \nu, i, j \mid p, q]$, que existe un nodo M^γ en el que el árbol auxiliar β puede ser adjuntado para reconocer la cadena de entrada a partir de la posición j. También puede verificar, mediante el ítem $[\mathbf{F}^\beta \to \bullet\bot, k, k \mid -, -]$, que se ha alcanzado el nodo pie del árbol auxiliar β. Lo que no puede verificar este paso deductivo es que el árbol al que pertenece dicho nodo pie se corresponda con la instancia de β que ha sido utilizada en la operación de adjunción que nos ocupa, pues para ello tendría que verificar que el extremo izquierdo de la frontera de la instancia de β es j, información que no es posible obtener a partir de los ítems definidos para el esquema de análisis **Ear**.

- El paso deductivo $\mathcal{D}_{\text{Ear}}^{\text{FootComp}}$ puede verificar, mediante el ítem $[N^\gamma \to \delta \bullet M^\gamma \nu, i, j \mid p, q]$, que existe un nodo M^γ en el que el árbol auxiliar β puede ser adjuntado para reconocer la cadena de entrada a partir de la posición j. También puede verificar, mediante el ítem $[\mathbf{F}^\beta \to \bullet\bot, k, k \mid -, -]$, que se ha alcanzado el nodo pie del árbol auxiliar β en la posición k de la cadena de entrada. Por último, el ítem $[M^\gamma \to \delta\bullet, k, l \mid p, q]$ permite verificar que la frontera del subárbol enraizado en M^γ comienza en la posición k de la cadena de entrada. Pero al igual que en el caso anterior y por las mismas razones, no puede verificar que el árbol al que pertenece el nodo pie se corresponde con la instancia de β que ha sido utilizada en la operación de adjunción que nos ocupa.

En consecuencia, para obtener un esquema de análisis que se corresponda con un algoritmo del tipo Earley para TAG que posea la propiedad del prefijo válido es necesario modificar la forma de los ítems para incluir un nuevo elemento, un índice que indique la posición del extremo izquierdo de la frontera del árbol al que se refieren los nodos de cada ítem que se genere. Esta operación se corresponde con la aplicación de un refinamiento de los ítems utilizados hasta el momento. Los nuevos ítems son de la forma

$$\left\{ \begin{array}{ll} [h, N^\gamma \to \delta \bullet \nu, i, j \mid p, q] \mid & \exists \alpha \in \boldsymbol{I},\ \mathbf{R}^\alpha \stackrel{*}{\Rightarrow} a_1 \ldots a_h \mathbf{R}^\gamma \mu,\ \mathbf{R}^\gamma \stackrel{*}{\Rightarrow} a_{h+1} \ldots a_i \delta \nu \upsilon \text{ y además:} \\ & \delta \stackrel{*}{\Rightarrow} a_i \ldots a_p\, \mathbf{F}^\gamma\, a_{q+1} \ldots a_j \stackrel{*}{\Rightarrow} a_i \ldots a_j \text{ sii } (p,q) \neq (-,-) \\ & \delta \stackrel{*}{\Rightarrow} a_i \ldots a_j \text{ sii } (p,q) = (-,-) \end{array} \right\}$$

con lo cual un ítem $[N^\gamma \to \delta \bullet \nu, i, j \mid p, q]$ de **Ear** se corresponde ahora con el conjunto de ítems $[h, N^\gamma \to \delta \bullet \nu, i, j \mid p, q]\ \forall h \in [0, n]$.

[6] Para una comparación experimental entre los algoritmos de Schabes y Nederhof, consultar [63].

Una vez definidos los nuevos ítems podemos pasar a describir el esquema de análisis **Earley** que corresponde a la primera versión de un algoritmo de tipo Earley para TAG que cumple la propiedad del prefijo válido. El correspondiente sistema de análisis sintáctico se define a continuación.

Esquema de análisis sintáctico 3.6 El sistema de análisis $\mathbb{P}_{\text{Earley}}$ que se corresponde con la el algoritmo de análisis sintáctico de tipo Earley para TAG que cumple la propiedad del prefijo válido, dada una gramática de adjunción de árboles $\mathcal{T}$ y una cadena de entrada $a_1 \dots a_n$ se define como sigue:

$$\mathcal{I}_{\text{Earley}} = \left\{ \begin{array}{ll} [h, N^\gamma \to \delta \bullet \nu, i, j \mid p, q] \mid & N^\gamma \to \delta\nu \in \mathcal{P}(\gamma),\ \gamma \in \boldsymbol{I} \cup \boldsymbol{A}, \\ & 0 \leq h \leq i \leq j,\ (p, q) \leq (i, j) \end{array} \right\}$$

$$\mathcal{D}^{\text{Init}}_{\text{Earley}} = \frac{}{\vdash [0, \top \to \bullet \mathbf{R}^\alpha, 0, 0 \mid -, -]} \quad \alpha \in \boldsymbol{I}$$

$$\mathcal{D}^{\text{Scan}}_{\text{Earley}} = \frac{[h, N^\gamma \to \delta \bullet a\nu, i, j \mid p, q],\ [a, j, j+1]}{[h, N^\gamma \to \delta a \bullet \nu, i, j+1 \mid p, q]}$$

$$\mathcal{D}^{\text{Pred}}_{\text{Earley}} = \frac{[h, N^\gamma \to \delta \bullet M^\gamma \nu, i, j \mid p, q]}{[h, M^\gamma \to \bullet \upsilon, j, j \mid -, -]} \quad \mathbf{nil} \in \text{adj}(M^\gamma)$$

$$\mathcal{D}^{\text{Comp}}_{\text{Earley}} = \frac{[h, N^\gamma \to \delta \bullet M^\gamma \nu, i, k \mid p, q],\ [h, M^\gamma \to \upsilon \bullet, k, j \mid p', q']}{[h, N^\gamma \to \delta M^\gamma \bullet \nu, i, j \mid p \cup p', q \cup q']} \quad \mathbf{nil} \in \text{adj}(M^\gamma)$$

$$\mathcal{D}^{\text{AdjPred}}_{\text{Earley}} = \frac{[h, N^\gamma \to \delta \bullet M^\gamma \nu, i, j \mid p, q]}{[j, \top \to \bullet \mathbf{R}^\beta, j, j \mid -, -]} \quad \beta \in \text{adj}(M^\gamma)$$

$$\mathcal{D}^{\text{FootPred}}_{\text{Earley}} = \frac{[j, \mathbf{F}^\beta \to \bullet \bot, k, k \mid -, -],\ [h, N^\gamma \to \delta \bullet M^\gamma \nu, i, j \mid p, q]}{[h, M^\gamma \to \bullet \delta, k, k \mid -, -]} \quad \beta \in \text{adj}(M^\gamma)$$

$$\mathcal{D}^{\text{FootComp}}_{\text{Earley}} = \frac{\begin{array}{l} [h, M^\gamma \to \delta \bullet, k, l \mid p, q], \\ [j, \mathbf{F}^\beta \to \bullet \bot, k, k \mid -, -], \\ [h, N^\gamma \to \delta \bullet M^\gamma \nu, i, j \mid p', q'] \end{array}}{[j, \mathbf{F}^\beta \to \bot \bullet, k, l \mid k, l]} \quad \begin{array}{l} \beta \in \text{adj}(M^\gamma), \\ p \cup p' \text{ y } q \cup q' \text{ está definido} \end{array}$$

$$\mathcal{D}^{\text{AdjComp}^1}_{\text{Earley}} = \frac{\begin{array}{l} [j, \top \to \mathbf{R}^\beta \bullet, j, m \mid k, l], \\ [h, M^\gamma \to \upsilon \bullet, k, l \mid p, q], \\ [h, \mathbf{F}^\gamma \to \bot \bullet, p, q \mid p, q], \\ [h, N^\gamma \to \delta \bullet M^\gamma \nu, i, j \mid -, -] \end{array}}{[h, N^\gamma \to \delta M^\gamma \bullet \nu, i, m \mid p, q]} \quad \beta \in \text{adj}(M^\gamma)$$

$$\mathcal{D}^{\text{AdjComp}^2}_{\text{Earley}} = \frac{\begin{array}{l} [j, \top \to \mathbf{R}^\beta \bullet, j, m \mid k, l], \\ [h, M^\gamma \to \upsilon \bullet, k, l \mid -, -], \\ [h, N^\gamma \to \delta \bullet M^\gamma \nu, i, j \mid p', q'] \end{array}}{[h, N^\gamma \to \delta M^\gamma \bullet \nu, i, m \mid p', q']} \quad \beta \in \text{adj}(M^\gamma)$$

$$\mathcal{D}_{\text{Earley}} = \mathcal{D}^{\text{Init}}_{\text{Earley}} \cup \mathcal{D}^{\text{Scan}}_{\text{Earley}} \cup \mathcal{D}^{\text{Pred}}_{\text{Earley}} \cup \mathcal{D}^{\text{Comp}}_{\text{Earley}} \cup \mathcal{D}^{\text{AdjPred}}_{\text{Earley}}$$
$$\cup\, \mathcal{D}^{\text{FootPred}}_{\text{Earley}} \cup \mathcal{D}^{\text{FootComp}}_{\text{Earley}} \cup \mathcal{D}^{\text{AdjComp}^1}_{\text{Earley}} \cup \mathcal{D}^{\text{AdjComp}^2}_{\text{Earley}}$$

$$\mathcal{F}_{\text{Earley}} = \left\{ \ [0, \top \to \mathbf{R}^\alpha \bullet, 0, n \mid -, -] \mid \alpha \in \boldsymbol{I} \ \right\}$$

§

En la figura 3.5 se muestra una representación gráfica de la aplicación del paso deductivo $\mathcal{D}^{\text{AdjComp}^1}_{\text{Earley}}$. Es interesante comparar este nueva figura con la 3.4 correspondiente al mismo paso deductivo del algoritmo de tipo Earley sin la propiedad del prefijo válido con el fin de observar cómo se restringen los árboles candidatos en la aplicación del paso deductivo. Se puede observar que la única diferencia entre ambas radica en que el extremo izquierdo del árbol γ está explícitamente indicado por el índice h en la figura 3.5, mientras que en la figura 3.4 se consideraba universalmente cuantificado.

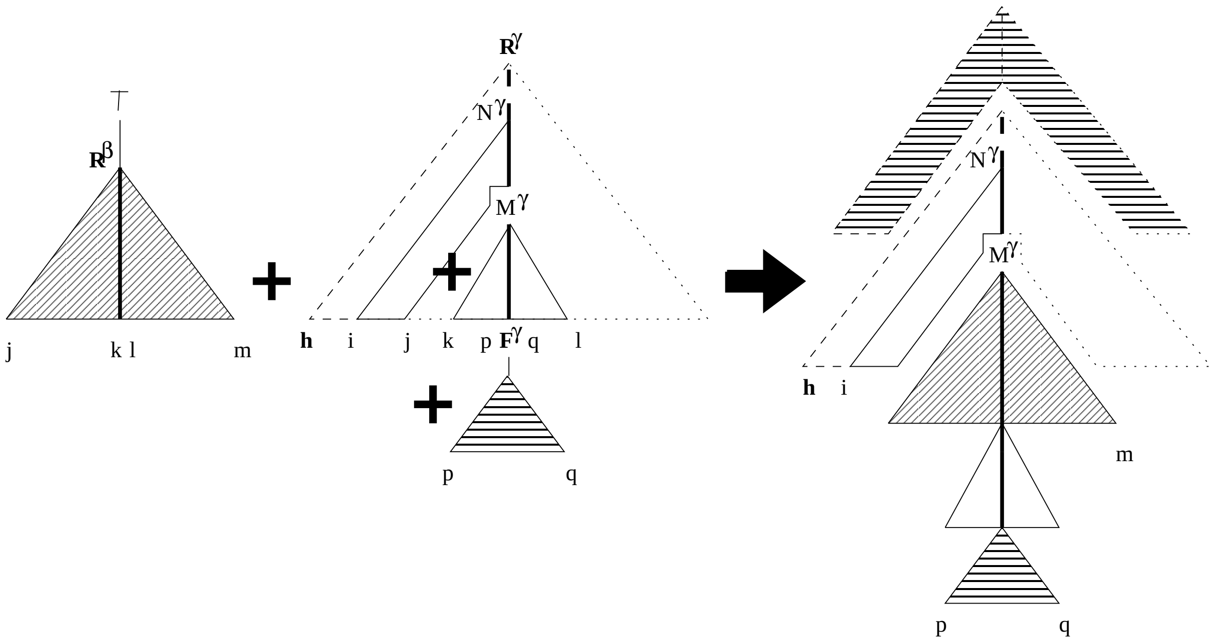

Figura 3.5: Descripción gráfica de un paso $\mathcal{D}^{\text{AdjComp}^1}_{\text{Earley}}$

Proposición 3.5 **Ear** $\stackrel{\mathbf{ir}}{\Longrightarrow}$ **Earley**.

<u>Demostración</u>:

Para demostrar que el esquema de análisis **Earley** es derivable del esquema de análisis **Ear** mediante refinamiento de los ítems definiremos la siguiente función:

$$f([h, N^\gamma \to \delta \bullet \nu, i, j \mid p, q \mid adj]) = [N^\gamma \to \delta \bullet \nu, i, j \mid p, q \mid adj]$$

de la cual se obtiene directamente que $\mathcal{I}_{\text{Ear}} = f(\mathcal{I}_{\text{Earley}})$ y que $\Delta_{\text{Ear}} = f(\Delta_{\text{Earley}})$ por inducción en la longitud de las secuencias de derivación. En consecuencia, $\mathbb{P}_{\text{Ear}} \xrightarrow{\mathbf{ir}} \mathbb{P}_{\text{Earley}}$, con lo que hemos probado lo que pretendíamos. ☐

Un aspecto interesante a tener en cuenta es que el ítem $[h, \mathbf{F}^\gamma \to \bot\bullet, p, q \mid p, q]$ es redundante en el paso $\mathcal{D}_{\text{Earley}}^{\text{AdjComp}^1}$, puesto que su existencia viene implícitamente determinada por la existencia del ítem $[h, M^\gamma \to \delta\bullet, k, l \mid p, q]$, ya que en otro caso este último sería inconsistente y por consiguiente algoritmo sería incorrecto. La finalidad de su presencia en dicho conjunto de pasos deductivos, así como en $\mathcal{D}_{\text{Ear}}^{\text{AdjComp}^1}$, es facilitar la transición hacia el esquema de análisis **Nederhof**. Si prescindimos de dicho ítem los pasos deductivos $\mathcal{D}_{\text{Earley}}^{\text{AdjComp}^1}$ y $\mathcal{D}_{\text{Earley}}^{\text{AdjComp}^2}$ se podrían fundir en uno solo:

$$\mathcal{D}_{\text{Earley}}^{\text{AdjComp}} = \frac{\begin{array}{l}[j, \top \to \mathbf{R}^\beta\bullet, j, m \mid k, l],\\ [h, M^\gamma \to \upsilon\bullet, k, l \mid p, q],\\ [h, N^\gamma \to \delta \bullet M^\gamma\nu, i, j \mid p', q']\end{array}}{[h, N^\gamma \to \delta M^\gamma \bullet \nu, i, m \mid p \cup p', q \cup q']} \quad \beta \in \text{adj}(M^\gamma)$$

Por idénticas razones, los pasos $\mathcal{D}_{\text{Ear}}^{\text{AdjComp}^1}$ y $\mathcal{D}_{\text{Ear}}^{\text{AdjComp}^1}$ del esquema **Ear** podrían fundirse en un nuevo paso $\mathcal{D}_{\text{Ear}}^{\text{AdjComp}}$:

$$\mathcal{D}_{\text{Ear}}^{\text{AdjComp}} = \frac{\begin{array}{l}[\top \to \mathbf{R}^\beta\bullet, j, m \mid k, l],\\ [M^\gamma \to \upsilon\bullet, k, l \mid p, q],\\ [N^\gamma \to \delta \bullet M^\gamma\nu, i, j \mid p', q']\end{array}}{[N^\gamma \to \delta M^\gamma \bullet \nu, i, m \mid p \cup p', q \cup q']} \quad \beta \in \text{adj}(M^\gamma)$$

En consecuencia, en lugar de la evolución $\mathbf{E} \overset{\mathbf{sr}}{\Longrightarrow}\overset{\mathbf{df}}{\Longrightarrow} \mathbf{Ear} \overset{\mathbf{ir}}{\Longrightarrow} \mathbf{Earley}$ podríamos haber definido una línea evolutiva $\mathbf{E} \overset{\mathbf{df}}{\Longrightarrow} \mathbf{Ear}' \overset{\mathbf{ir}}{\Longrightarrow} \mathbf{Earley}' \overset{\mathbf{sr}}{\Longrightarrow}\overset{\mathbf{df}}{\Longrightarrow} \mathbf{Earley}$, donde $\mathbf{Ear}'$ y $\mathbf{Earley}'$ son como **Ear** y **Earley**, respectivamente, excepto por la sustitución de los pasos AdjComp^1 y AdjComp^2 por AdjComp. Este resultado viene a mostrar una vez más que existen varios caminos para transformar un esquema de análisis sintáctico en otro, tal como establece Sikkel en [189] para el caso de los algoritmos de análisis de gramáticas independientes del contexto y que nosotros mostramos aquí para el caso de las gramáticas de adjunción de árboles.

El algoritmo descrito por el esquema **Earley** presenta una complejidad temporal de $\mathcal{O}(n^7)$. Aunque aparentemente la utilización de 8 índices con respecto a la cadena de entrada en los pasos deductivos $\mathcal{D}_{\text{Earley}}^{\text{AdjComp}^1}$ y $\mathcal{D}_{\text{Earley}}^{\text{AdjComp}^2}$ hace pensar en una complejidad $\mathcal{O}(n^8)$, la utilización de aplicación parcial o *currificación* en dichos pasos reduce la complejidad hasta $\mathcal{O}(n^7)$. En el caso de $\mathcal{D}_{\text{Earley}}^{\text{AdjComp}^1}$ la aplicación parcial sobre los dos primeros ítems involucra combinar 7 índices, de donde resulta la complejidad $\mathcal{O}(n^7)$ con respecto a la cadena de entrada, pero únicamente los 5 índices j, m, h, p y q forman parte del resultado intermedio puesto que son los únicos que se necesitarán en posteriores aplicaciones parciales. La siguiente aplicación parcial combina el ítem intermedio con el tercer ítem del paso deductivo, operación que involucra únicamente a los 5 ítems mencionados anteriormente, que son conservados en el resultado intermedio producido. Por último, la aplicación parcial con el cuarto ítem involucra la combinación de los 6 índices h, i, j, m, p, q. El caso de $\mathcal{D}_{\text{Earley}}^{\text{AdjComp}^2}$ es análogo al de $\mathcal{D}_{\text{Earley}}^{\text{AdjComp}^1}$.

El aumento de la complejidad de $\mathcal{O}(n^6)$ a $\mathcal{O}(n^7)$ se debe al índice adicional incorporado en los ítems y que indica la posición del extremo izquierdo de la frontera del árbol que se está considerando. La inclusión en los ítems de este índice había surgido por la necesidad de controlar que se están utilizando los árboles correctos para reconocer el pie de un árbol auxiliar. En consecuencia, dicho índice sólo es de real utilidad en los pasos FootPred y FootComp. El resto de los pasos deductivos únicamente propagan el valor de dicho índice. Por consiguiente, en caso de que sea necesario estos últimos pasos deductivos pueden ser refinados, dividiéndolos en varios pasos con el fin de generar ítems intermedios carentes de dicho índice. Esta técnica, si se aplica convenientemente, puede llegar a reducir la complejidad de los algoritmos de análisis. Evidentemente, hay que verificar que los ítems intermedios portan la información necesaria para que el resultado de la composición de los pasos deductivos obtenidos mediante el refinamiento de uno dado sea equivalente al resultado obtenido por aplicar directamente el paso deductivo sin refinar.

En el caso completo del esquema de análisis **Earley**, para reducir la complejidad a $\mathcal{O}(n^6)$ es suficiente con hacer uso de la *propiedad de independencia del contexto de TAG* [214]. Básicamente, lo que dicha propiedad establece es que cada operación de adjunción es independiente de la previa o posterior aplicación de cualquier otra operación de adjunción. Una consecuencia de esta propiedad es que si en un nodo M^γ de un árbol γ está permitida la adjunción de un árbol auxiliar β y se cumplen las tres condiciones siguientes:

1. la parte izquierda de la frontera de $\gamma - M^\gamma$ se extiende desde la posición h hasta la posición j de la cadena de entrada, donde $\gamma - M^\gamma$ denota el resultado de escindir el subárbol enraizado por M^γ de γ;
2. la frontera del árbol β se expande desde la posición j hasta la posición m de la cadena de entrada con una discontinuidad en el pie desde la posición k hasta la l;
3. la frontera del subárbol enraizado por M^γ abarca precisamente desde la posición k hasta la posición l;

como resultado de la adjunción de β en M^γ la frontera del subárbol enraizado por este último nodo se expande desde las posición j hasta la m sin discontinuidad y dicho subárbol puede ser insertado en todo árbol $\beta - M^\gamma$ cuya frontera izquierda finalice en la posición j independientemente de la posición h en la que se sitúe el extremo izquierdo de dicha frontera. Precisamente, los algoritmos de tipo Earley para TAG que no cumplen la propiedad del prefijo válido hacen uso de esta propiedad para verificar la corrección de la adjunción realizada en los pasos AdjComp. Como se recordará de la sección precedente, los ítems de los esquemas de análisis de dichos algoritmos no incorporan el índice h del extremo izquierdo.

En consecuencia, los ítems que utilizaremos en el esquema de análisis sintáctico **Nederhof** correspondiente al algoritmo de tipo Earley para TAG presentado por Nederhof en [125] que preserva la propiedad del prefijo válido con una complejidad $\mathcal{O}(n^6)$, son de dos tipos: los definidos para el esquema **Earley** y los pseudo-ítem que definimos a continuación

$$\left\{ \begin{array}{lll} [[N^\gamma \rightarrow \delta \bullet \nu, i, j \mid p, q]] \mid & \delta \stackrel{*}{\Rightarrow} a_i \ldots a_p\, \mathbf{F}^\gamma\, a_{q+1} \ldots a_j \stackrel{*}{\Rightarrow} a_i \ldots a_j & \text{sii } (p,q) \neq (-,-) \\ & \delta \stackrel{*}{\Rightarrow} a_i \ldots a_j & \text{sii } (p,q) = (-,-) \end{array} \right\}$$

Ahora podemos definir el esquema de análisis **Nederhof** cuyo sistema de análisis mostramos a continuación.

Esquema de análisis sintáctico 3.7 El sistema de análisis $\mathbb{P}_{\text{Nederhof}}$ que se corresponde con la el algoritmo de análisis sintáctico de tipo Earley para TAG que cumple la propiedad del prefijo válido y posee una complejidad $\mathcal{O}(n^6)$, dada una gramática de adjunción de árboles $\mathcal{T}$ y una cadena de entrada $a_1 \ldots a_n$ se define como sigue:

$$\mathcal{I}^{(1)}_{\text{Nederhof}} = \mathcal{I}_{\text{Earley}} = \left\{ \begin{array}{ll} [h, N^\gamma \rightarrow \delta \bullet \nu, i, j \mid p, q] \mid & N^\gamma \rightarrow \delta\nu \in \mathcal{P}(\gamma),\ \gamma \in \boldsymbol{I} \cup \boldsymbol{A}, \\ & 0 \leq h \leq i \leq j,\ (p,q) \leq (i,j) \end{array} \right\}$$

$$\mathcal{I}^{(2)}_{\text{Nederhof}} = \left\{ \begin{array}{ll} [[N^\gamma \rightarrow \delta \bullet \nu, i, j \mid p, q]] \mid & N^\gamma \rightarrow \delta\nu \in \mathcal{P}(\gamma),\ \gamma \in \boldsymbol{I} \cup \boldsymbol{A}, \\ & 0 \leq h \leq i \leq j,\ (p,q) \leq (i,j) \end{array} \right\}$$

$$\mathcal{I}_{\text{Nederhof}} = \mathcal{I}^{(1)}_{\text{Nederhof}} \cup \mathcal{I}^{(2)}_{\text{Nederhof}}$$

$$\mathcal{D}^{\text{Init}}_{\text{Nederhof}} = \mathcal{D}^{\text{Init}}_{\text{Earley}} = \frac{}{\vdash [0, \top \to \bullet \mathbf{R}^{\alpha}, 0, 0 \mid -, -]} \quad \alpha \in \boldsymbol{I}$$

$$\mathcal{D}^{\text{Scan}}_{\text{Nederhof}} = \mathcal{D}^{\text{Scan}}_{\text{Earley}} = \frac{[h, N^{\gamma} \to \delta \bullet a\nu, i, j \mid p, q],\ [a, j, j+1]}{[h, N^{\gamma} \to \delta a \bullet \nu, i, j+1 \mid p, q]}$$

$$\mathcal{D}^{\text{Pred}}_{\text{Nederhof}} = \mathcal{D}^{\text{Pred}}_{\text{Earley}} = \frac{[h, N^{\gamma} \to \delta \bullet M^{\gamma}\nu, i, j \mid p, q]}{[h, M^{\gamma} \to \bullet \upsilon, j, j \mid -, -]} \quad \mathbf{nil} \in \text{adj}(M^{\gamma})$$

$$\mathcal{D}^{\text{Comp}}_{\text{Nederhof}} = \mathcal{D}^{\text{Comp}}_{\text{Earley}} = \frac{[h, N^{\gamma} \to \delta \bullet M^{\gamma}\nu, i, k \mid p, q],\ [h, M^{\gamma} \to \upsilon \bullet, k, j \mid p', q']}{[h, N^{\gamma} \to \delta M^{\gamma} \bullet \nu, i, j \mid p \cup p', q \cup q']} \quad \mathbf{nil} \in \text{adj}(M^{\gamma})$$

$$\mathcal{D}^{\text{AdjPred}}_{\text{Nederhof}} = \mathcal{D}^{\text{AdjPred}}_{\text{Earley}} = \frac{[h, N^{\gamma} \to \delta \bullet M^{\gamma}\nu, i, j \mid p, q]}{[j, \top \to \bullet \mathbf{R}^{\beta}, j, j \mid -, -]} \quad \beta \in \text{adj}(M^{\gamma})$$

$$\mathcal{D}^{\text{FootPred}}_{\text{Nederhof}} = \mathcal{D}^{\text{FootPred}}_{\text{Earley}} = \frac{[j, \mathbf{F}^{\beta} \to \bullet \bot, k, k \mid -, -],\ [h, N^{\gamma} \to \delta \bullet M^{\gamma}\nu, i, j \mid p, q]}{[h, M^{\gamma} \to \bullet \delta, k, k \mid -, -]} \quad \beta \in \text{adj}(M^{\gamma})$$

$$\mathcal{D}^{\text{FootComp}}_{\text{Nederhof}} = \mathcal{D}^{\text{FootComp}}_{\text{Earley}} = \frac{\begin{array}{l}[h, M^{\gamma} \to \upsilon \bullet, k, l \mid p, q],\\ [j, \mathbf{F}^{\beta} \to \bullet \bot, k, k \mid -, -],\\ [h, N^{\gamma} \to \delta \bullet M^{\gamma}\nu, i, j \mid p', q']\end{array}}{[j, \mathbf{F}^{\beta} \to \bot \bullet, k, l \mid k, l]} \quad \begin{array}{l}\beta \in \text{adj}(M^{\gamma}),\\ p \cup p' \ y \ q \cup q' \ \textit{está definido}\end{array}$$

$$\mathcal{D}^{\text{AdjComp}^0}_{\text{Nederhof}} = \frac{\begin{array}{l}[j, \top \to \mathbf{R}^{\beta} \bullet, j, m \mid k, l],\\ [h, M^{\gamma} \to \upsilon \bullet, k, l \mid p, q],\end{array}}{[[M^{\gamma} \to \upsilon \bullet, j, m \mid p, q]]} \quad \beta \in \text{adj}(M^{\gamma})$$

$$\mathcal{D}^{\text{AdjComp}^1}_{\text{Nederhof}} = \frac{\begin{array}{l}[[M^{\gamma} \to \upsilon \bullet, j, m \mid p, q]],\\ [h, \mathbf{F}^{\gamma} \to \bot \bullet, p, q \mid p, q],\\ [h, N^{\gamma} \to \delta \bullet M^{\gamma}\nu, i, j \mid -, -]\end{array}}{[h, N^{\gamma} \to \delta M^{\gamma} \bullet \nu, i, m \mid p, q]}$$

$$\mathcal{D}^{\text{AdjComp}^2}_{\text{Nederhof}} = \frac{\begin{array}{l}[[M^{\gamma} \to \upsilon \bullet, j, m \mid -, -]],\\ [h, N^{\gamma} \to \delta \bullet M^{\gamma}\nu, i, j \mid p', q']\end{array}}{[h, N^{\gamma} \to \delta M^{\gamma} \bullet \nu, i, m \mid p', q']}$$

$$\begin{aligned}\mathcal{D}_{\text{Nederhof}} = \ & \mathcal{D}^{\text{Init}}_{\text{Nederhof}} \cup \mathcal{D}^{\text{Scan}}_{\text{Nederhof}} \cup \mathcal{D}^{\text{Pred}}_{\text{Nederhof}} \cup \mathcal{D}^{\text{Comp}}_{\text{Nederhof}} \cup \mathcal{D}^{\text{AdjPred}}_{\text{Nederhof}} \cup \mathcal{D}^{\text{FootPred}}_{\text{Nederhof}} \\ & \cup \mathcal{D}^{\text{FootComp}}_{\text{Nederhof}} \cup \mathcal{D}^{\text{AdjComp}^0}_{\text{Nederhof}} \cup \mathcal{D}^{\text{AdjComp}^1}_{\text{Nederhof}} \cup \mathcal{D}^{\text{AdjComp}^2}_{\text{Nederhof}}\end{aligned}$$

$$\mathcal{F}_{\text{Nederhof}} = \mathcal{F}_{\text{Earley}} = \left\{\ [0, \top \to \mathbf{R}^{\alpha} \bullet, 0, n \mid -, -] \mid \alpha \in \boldsymbol{I}\ \right\}$$

§

Obsérvese que se ha aplicado un refinamiento al paso deductivo $\mathcal{D}_{\text{Earley}}^{\text{AdjComp}^1}$ para obtener los pasos $\mathcal{D}_{\text{Nederhof}}^{\text{AdjComp}^0}$ y $\mathcal{D}_{\text{Nederhof}}^{\text{AdjComp}^1}$. Análogamente, el paso deductivo $\mathcal{D}_{\text{Earley}}^{\text{AdjComp}^2}$ ha sido refinado en los pasos $\mathcal{D}_{\text{Nederhof}}^{\text{AdjComp}^0}$ y $\mathcal{D}_{\text{Nederhof}}^{\text{AdjComp}^2}$. Para garantizar la corrección se verifica que:

- La aplicación consecutiva de $\mathcal{D}_{\text{Nederhof}}^{\text{AdjComp}^0}$ y $\mathcal{D}_{\text{Earley}}^{\text{AdjComp}^1}$ (resp.$\mathcal{D}_{\text{Nederhof}}^{\text{AdjComp}^0}$ y $\mathcal{D}_{\text{Nederhof}}^{\text{AdjComp}^2}$) es equivalente a la aplicación de $\mathcal{D}_{\text{Earley}}^{\text{AdjComp}^1}$ (resp. $\mathcal{D}_{\text{Earley}}^{\text{AdjComp}^2}$). Es fácil comprobar que ambos utilizan la misma información: todos los antecedentes de $\mathcal{D}_{\text{Nederhof}}^{\text{AdjComp}^1}$ (resp. $\mathcal{D}_{\text{Nederhof}}^{\text{AdjComp}^2}$) son utilizados por pasos del esquema **Nederhof** y toda la información presente en los antecedentes de los pasos del esquema **Nederhof** es utilizada por el paso $\mathcal{D}_{\text{Nederhof}}^{\text{AdjComp}^1}$ (resp. $\mathcal{D}_{\text{Nederhof}}^{\text{AdjComp}^2}$), puesto que el ítem intermedio no crea nueva información, sino que simplemente es un "resumen" de información contenida en los otros antecedentes. Es también fácil comprobar que en ambos casos se genera la misma información, puesto que los ítems generados (excluyendo pseudo-ítems) son idénticos en ambos casos.

- El paso deductivo $\mathcal{D}_{\text{Nederhof}}^{\text{AdjComp}^1}$ (resp. $\mathcal{D}_{\text{Nederhof}}^{\text{AdjComp}^2}$) solo puede ser aplicado si previamente se ha aplicado el paso $\mathcal{D}_{\text{Nederhof}}^{\text{AdjComp}^0}$. Se puede verificar fácilmente puesto que el paso $\mathcal{D}_{\text{Nederhof}}^{\text{AdjComp}^0}$ genera un ítem intermedio y los únicos pasos que toman un ítem intermedio como antecedente son $\mathcal{D}_{\text{Nederhof}}^{\text{AdjComp}^1}$ y $\mathcal{D}_{\text{Nederhof}}^{\text{AdjComp}^2}$.

En la figura 3.5 se muestra una representación gráfica de la aplicación de los pasos deductivos $\mathcal{D}_{\text{Nederhof}}^{\text{AdjComp}^0}$ y $\mathcal{D}_{\text{Nederhof}}^{\text{AdjComp}^1}$.

La complejidad del algoritmo descrito por el esquema de análisis **Nederhof** es $\mathcal{O}(n^6)$, puesto que en la combinación de los ítems de cualquier paso intervienen activamente a los sumo 6 índices con respecto a la cadena de entrada.

En [64] se describe una versión de este algoritmo en la que se utiliza una representación plana de los árboles elementales en lugar de la representación multicapa que se ha utilizado aquí.

Proposición 3.6 Earley $\stackrel{\text{sr}}{\Longrightarrow}$ Nederhof.

<u>Demostración</u>:

Para demostrar que el esquema de análisis **Nederhof** puede ser obtenido mediante un refinamiento de los pasos deductivos del esquema **Earley** debemos probar que para todo sistema de análisis $\mathbb{P}_{\text{Earley}}$ y $\mathbb{P}_{\text{Nederhof}}$ se cumple $\mathbb{P}_{\text{Earley}} \stackrel{\text{sr}}{\Longrightarrow} \mathbb{P}_{\text{Nederhof}}$. Ello conlleva demostrar que $\mathcal{I}_{\text{Earley}} \subseteq \mathcal{I}_{\text{Nederhof}}$ y que $\stackrel{*}{\vdash}_{\text{Earley}} \subseteq \stackrel{*}{\vdash}_{\text{Nederhof}}$. Lo primero se obtiene directamente puesto que $\mathcal{I}_{\text{Earley}} \subseteq \mathcal{I}_{\text{Nederhof}}$ por definición de los sistemas de análisis. Lo segundo se obtiene demostrando que $\mathcal{D}_{\text{Earley}} \subseteq \stackrel{*}{\vdash}_{\text{Nederhof}}$. Los únicos pasos deductivos de $\mathbb{P}_{\text{Earley}}$ que no se han incorporado directamente en $\mathbb{P}_{\text{Nederhof}}$ son $\mathcal{D}_{\text{Earley}}^{\text{AdjComp}^1}$ y $\mathcal{D}_{\text{Earley}}^{\text{AdjComp}^2}$ y para ellos se cumple que:

- Un paso $\mathcal{D}_{\text{Earley}}^{\text{AdjComp}^1}$ es equivalente a la aplicación de un paso $\mathcal{D}_{\text{Nederhof}}^{\text{Comp}^0}$ seguido de la aplicación de un paso $\mathcal{D}_{\text{Nederhof}}^{\text{Comp}^1}$:

$$\frac{[j, \top \to \mathbf{R}^{\beta}\bullet, j, m \mid k, l],\ [h, M^{\gamma} \to \upsilon\bullet, k, l \mid p, q],}{[[M^{\gamma} \to \upsilon\bullet, j, m \mid p, q]]}$$

$$\frac{[[M^{\gamma} \to \upsilon\bullet, j, m \mid p, q]],\ [h, \mathbf{F}^{\gamma} \to \bot\bullet, p, q \mid p, q],\ [h, N^{\gamma} \to \delta \bullet M^{\gamma}\nu, i, j \mid -, -]}{[h, N^{\gamma} \to \delta M^{\gamma} \bullet \nu, i, m \mid p, q]}$$

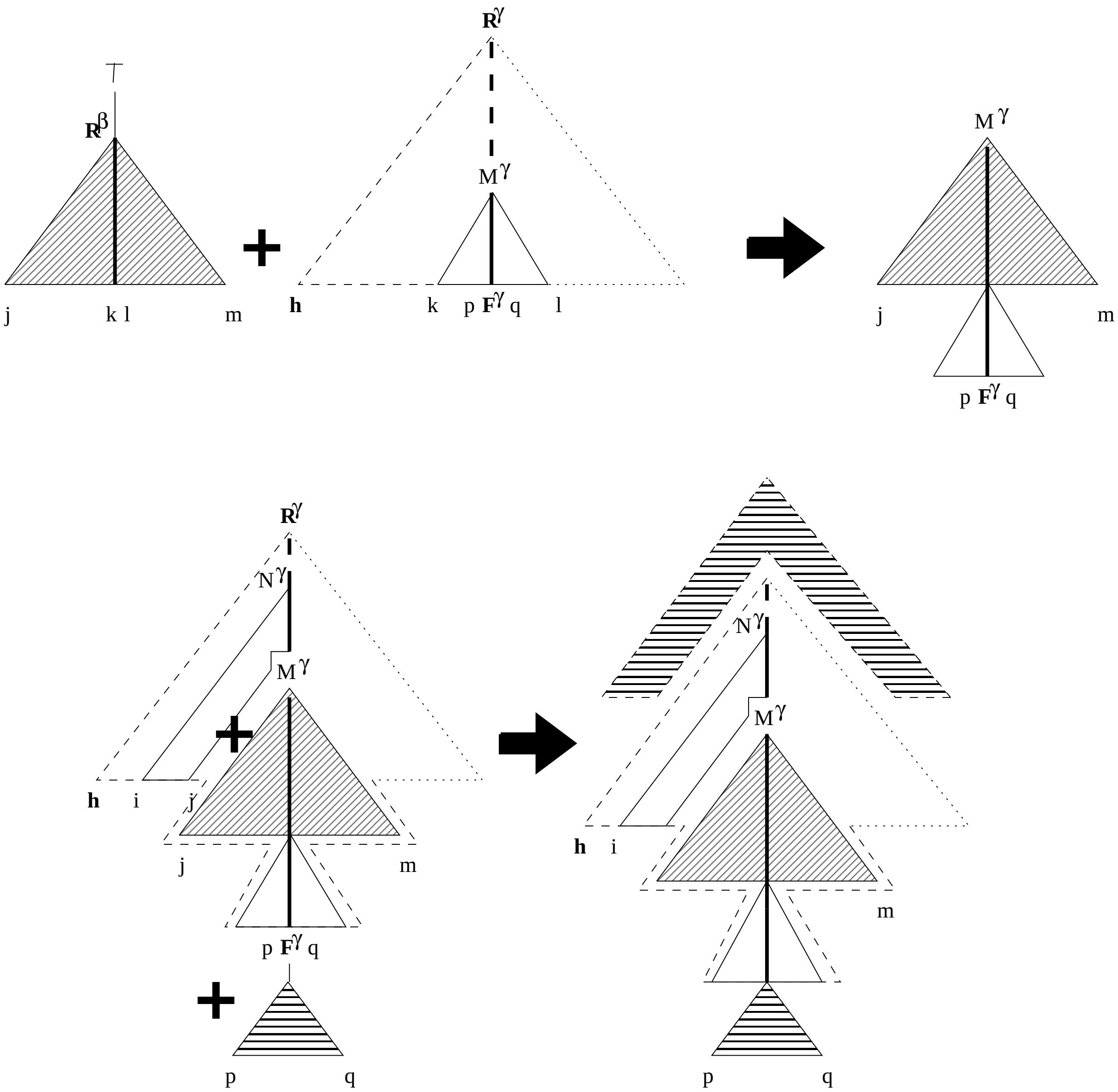

Figura 3.6: Descripción gráfica de la aplicación consecutiva de los pasos $\mathcal{D}^{\text{AdjComp}^0}_{\text{Nederhof}}$ y $\mathcal{D}^{\text{AdjComp1}}_{\text{Nederhof}}$

- Un paso $\mathcal{D}^{\text{AdjComp}^2}_{\text{Earley}}$ es equivalente a la aplicación de un paso $\mathcal{D}^{\text{Comp}^0}_{\text{Nederhof}}$ seguido de la aplicación de un paso $\mathcal{D}^{\text{Comp}^2}_{\text{Nederhof}}$:

$$\frac{[j, \top \to \mathbf{R}^\beta \bullet, j, m \mid k, l],\ [h, M^\gamma \to \upsilon \bullet, k, l \mid p, q],}{[[M^\gamma \to \upsilon \bullet, j, m \mid p, q]]}$$

$$\frac{[[M^\gamma \to \upsilon \bullet, j, m \mid p, q]],\ [h, N^\gamma \to \delta \bullet M^\gamma \nu, i, j \mid p', q']}{[h, N^\gamma \to \delta M^\gamma \bullet \nu, i, m \mid p', q']}$$

□

3.7 Análisis sintáctico de TAG lexicalizadas

Las gramáticas lexicalizadas poseen una propiedad muy interesante desde el punto de vista del análisis sintáctico: son finitamente ambiguas. Puesto que cada componente de la gramática (en el caso de TAG, cada árbol elemental) está asociado con un componente léxico, solamente un conjunto finito de tales estructuras pueden ser utilizadas para el análisis de una cadena de entrada dada y además solamente existe un número finito de combinaciones de dichas estructuras. En resumen, las gramáticas lexicalizadas impiden la aparición de análisis cíclicos.

El análisis de gramáticas lexicalizadas puede realizarse en dos fases, una primera en la cual se seleccionan todas las estructuras relevantes para la cadena de entrada que se pretende analizar y una segunda en la cual se aplica un algoritmo de análisis sintáctico que combine dichas estructuras. Este tipo de procesamiento se corresponde con un análisis fuera de línea. Las gramáticas lexicalizadas también pueden ser analizadas en línea, de tal modo que según se va avanzado en la lectura de la cadena de entrada se proporcionen las estructuras elementales correspondientes. Algunos autores [174, 168] sugieren que en análisis fuera de línea está mejor adaptado a este tipo de gramáticas puesto que las estructuras seleccionadas en la primera fase posibilitan al analizador sintáctico la utilización de información ascendente no local[7], restringiendo de este modo las posibilidades de combinación de las estructuras e incluso el número de estructuras a considerar. En efecto, al actuar de este modo, el analizador sintáctico solamente considerará aquellas estructuras relevantes para la cadena a analizar, por lo que se podría decir que trabaja sobre una subgramática relevante para la cadena de entrada.

Los beneficios que se obtiene de la lexicalización dependen del algoritmo de análisis que se vaya a aplicar. Los algoritmos puramente ascendentes, del tipo CYK, únicamente se benefician de la reducción del número de estructuras a considerar durante el proceso de análisis. Un algoritmo puramente descendente, basado en una exploración en profundidad con retroceso, conseguiría mayores beneficios, puesto que al ser las gramáticas lexicalizadas finitamente ambiguas el espacio de búsqueda es finito y por lo tanto el análisis terminará en todos los casos[8]. Loa algoritmos mixtos que utilizan información ascendente y descendente, como por ejemplo los algoritmos de tipo Earley, se ven también beneficiados por la lexicalización. Una primera ventaja surge del hecho de que ningún árbol elemental tiene la cadena vacía por frontera, lo cual significa que una adjunción no puede ser predicha y completada sin avanzar en el reconocimiento de la cadena de entrada. Por tanto, al terminación está asegurada. Adicionalmente, la

[7] Esta información puede incluso no estar acotada con respecto a la distancia [168], de tal modo que no se puede imitar su efecto mediante la utilización de un número limitado de símbolos de preanálisis. Esta características se puede aplicar por ejemplo al reconocimiento de frases hechas con expresiones arbitrarias intercaladas.

[8] Un analizador sintáctico puramente descendente puede no terminar para una gramática no lexicalizada puesto que puede intentar repetir indefinidamente el análisis del mismo conjunto de estructuras sin avanzar en el reconocimiento de la cadena de entrada.

utilización de una estrategia de dos fases permite que la selección de estructuras de acuerdo a los componentes léxicos presentes en la cadena de entrada ayude al analizador sintáctico en la tarea de filtrar las predicciones y/o compleciones para la adjunción y la sustitución. Resultados experimentales realizados Schabes y Joshi [174, 168] muestran que la estrategia de dos fases aumenta considerablemente la eficiencia de un algoritmo de análisis sintáctico de tipo Earley para TAG.

Todos los algoritmos mostrados en este capítulo pueden ser fácilmente adaptados a gramáticas de adjunción de árboles lexicalizadas. Para ello sólo es preciso incluir un paso deductivo para tratar la sustitución de un árbol en un nodo de sustitución. Puesto que dicho operación es independiente del contexto, no afecta a la complejidad espacial ni temporal de los algoritmos.

3.8 El bosque de análisis

Los algoritmos mostrados hasta el momento, tal y como han sido descritos, son realmente reconocedores y no analizadores sintácticos, puesto que no construyen árboles de derivación. Sin embargo, cada uno de los pasos deductivos contiene la información necesaria para generar la parte correspondiente de un árbol de derivación y, puesto que todos los algoritmos recorren todas las posibles derivaciones, se pueden reconstruir todos los posibles árboles de derivación.

Puesto que estamos tratando con analizadores no deterministas se trata de construir una estructura, denominada *bosque de análisis* [30, 215] que permita representar todas las derivaciones de un forma compacta, compartiendo subderivaciones comunes, y que permita extraer cada una de las derivaciones en tiempo lineal con respecto al tamaño del bosque de análisis. El problema de la construcción del bosque de análisis para TAG ha sido estudiado con anterioridad por Vijay-Shanker y Weir en [215], que han propuesto dos posibles soluciones: la utilización de gramáticas independientes del contexto y la utilización de gramáticas lineales de índices. Cualquiera de las soluciones es aplicable a los algoritmos de análisis sintáctico mostrados en este capítulo.

3.8.1 Gramáticas independientes del contexto como bosque de análisis

Es posible representar el bosque compartido mediante una gramática independiente del contexto que capture la independencia al contexto de la operación de adjunción. Los no-terminales de la gramática serán de la forma $\langle tb, N^\gamma, i, j, p, q\rangle$ donde $tb \in \{\top, \bot\}$ se utiliza para indicar si el no-terminal representa al nodo N^γ antes ($\bot$) o después ($\top$) de una adjunción. Es interesante observar que los no-terminales son casi idénticos a los ítems utilizados en el esquema de análisis **CYK**. Mediante una pequeña modificación en los pasos deductivos[9] sería posible hacer $adj =$ true siempre que $tb = \top$ y que $adj =$ false siempre que $tb = \bot$. Puesto que los ítems de los restantes esquemas son un refinamiento de los ítems de **CYK** la información necesaria para los no-terminales se puede obtener directamente a partir de los ítems.

Respecto a la forma de las producciones, a modo de ejemplo, mostramos la producción correspondiente a la adjunción del árbol auxiliar β en el nodo N^γ:

$$\langle \top, N^\gamma, i, j, r, s\rangle \to \langle \top, \mathbf{R}^\beta, i, j, p, q\rangle \ \langle \bot, N^\gamma, p, q, r, s\rangle$$

la cual se corresponde con el paso deductivo de adjunción del esquema de análisis **CYK**. Al igual que ocurría con los no-terminales, las producciones del bosque de análisis se pueden obtener directamente a partir de los pasos deductivos en los diferentes esquemas de análisis.

[9] Esencialmente la adición de un paso deductivo $\mathcal{D}^{\text{NoAdj}}_{\text{CYK}} = \frac{[N^\gamma, i, j \mid p, q \mid \text{false}]}{[N^\gamma, i, j \mid p, q \mid \text{true}]}$ es aplicable siempre que la realización de una adjunción sobre N^γ sea opcional.

El número de producciones es $\mathcal{O}(n^6)$ y la construcción de la gramática tienen una complejidad temporal $\mathcal{O}(n^6)$, por lo que la complejidad temporal de los algoritmos permanece inalterable, aunque la complejidad espacial aumenta de $\mathcal{O}(n^4)$ ó $\mathcal{O}(n^5)$ a $\mathcal{O}(n^6)$.

Un aspecto interesante a destacar es que aunque el bosque de análisis construido de esta forma codifica las derivaciones para una cadena de entrada dada, el lenguaje derivado por la gramática independiente del contexto no es importante. Lo que importa es que el lenguaje generado es no vacío si la cadena pertenece a la TAG original y en tal caso las derivaciones para la TAG original puede ser obtenidas en tiempo lineal a partir de las derivaciones de la gramática independiente del contexto que codifica el bosque compartido, siempre que esta haya sido podada para eliminar los símbolos inútiles.

3.8.2 Gramáticas lineales de índices como bosque de análisis

Se puede representar el bosque compartido mediante una gramática lineal de índices utilizando la transformación de TAG en LIG definida en [214]. A modo de ejemplo, las siguiente producciones representan la adjunción del árbol auxiliar β en el nodo N^γ:

$$\langle \top, i, j\rangle[\circ\circ N^\gamma] \to \langle \top, i, j\rangle[\circ\circ N^\gamma \mathbf{R}^\beta]$$

$$\langle \bot, p, q\rangle[\circ\circ N^\gamma \mathbf{F}^\beta] \to \langle \bot, p, q\rangle[\circ\circ N^\gamma]$$

La primera producción representa el final de la adjunción mientras que la segunda representa el reconocimiento del nodo pie del árbol auxiliar.

La información contenida en los no-terminales y producciones de la gramática lineal de índices puede obtenerse directamente a partir de los ítems y pasos deductivos de los esquemas de análisis sintáctico.

El tamaño de la gramática es $\mathcal{O}(n^3)$ y el tiempo necesario para construirla es $\mathcal{O}(n^6)$, por lo que las complejidades temporal y espacial de los algoritmos no se ven afectadas.

El inconveniente de esta representación estriba en que no se pueden extraer directamente los árboles de derivación individuales, sino que es preciso construir una estructura auxiliar que toma la forma de un autómata finito que reconoce las pilas de índices asociadas a cada terminal. Una vez construido dicho autómata finito, cada uno de los árboles de derivación puede ser extraído con una complejidad temporal que tiene por cota inferior $\mathcal{O}(n^4)$ y que en el caso de las gramáticas de adjunción de árboles lexicalizadas tiene como cota superior $\mathcal{O}(n^5)$. Puesto que el tamaño del bosque compartido es $\mathcal{O}(n^3)$, no se asegura que en todos los casos la recuperación de los árboles de derivación individuales se pueda realizar en tiempo lineal con respecto al tamaño del bosque compartido.

3.9 Otros algoritmos de análisis sintáctico para TAG

Presentamos a continuación un conjunto de algoritmos de análisis sintáctico de TAG que si bien no están en el camino principal de la evolución de los algoritmos de análisis mostrado anteriormente, presentan interés bien por constituir caminos laterales del camino principal de evolución, bien por haber constituido hitos importantes aunque posteriormente hayan quedado relegados, o bien por constituir ejemplos singulares de aplicación a TAG de ciertas áreas del análisis sintáctico, como la incrementalidad o el paralelismo.

3.9.1 El algoritmo de Lang

Lang describe en [106] un algoritmo tabular de análisis de TAG, con el objetivo principal de mostrar que técnicas de análisis muy generales pueden ser utilizadas para desarrollar un analizador sintáctico de tipo Earley para TAG. Concretamente, Lang utiliza las siguientes técnicas:

gramáticas de cláusulas definidas (DCG) [143], autómatas lógicos a pila (LPDA) [56] y evaluación en programación dinámica de autómatas a pila. Efectivamente, su algoritmo se desglosa en tres fases principales:

1. Traducción de la gramática TAG a una gramática DCG.
2. Construcción de un LPDA ascendente predictivo.
3. Modificación de la técnica general de evaluación en programación dinámica de este LPDA con el fin de obtener un analizador que actúe de izquierda a derecha a pesar de la presencia de discontinuidades, modificación que se basa en propiedades generales de los LPDA.

Trataremos de describir un esquema de análisis **Lang** para el algoritmo de Lang, tarea que no es ciertamente fácil dada su complejidad: como muestra de ello, basta pensar en que su comportamiento fue descrito en [106] mediante 28 tipos de transiciones LPDA. En [23] se muestra una implementación del algoritmo.

El primer paso para la descripción de **Lang** consiste en describir el conjunto de ítems que serán utilizados. En este caso, dicho conjunto puede subdividirse en 6 subconjuntos diferentes, que denominaremos 1a, 1b, 1c, 2a, 2b y 2c. Los tres primeros estarán dedicados al reconocimiento de nodos que no forman parte de la espina de un árbol auxiliar, mientras que los tres restantes tratarán precisamente con dichos elementos. Definimos a continuación los ítems que forman cada uno de dichos conjuntos:

1a Los ítems del primer conjunto son de la forma

$$\left\{ \begin{array}{lll} [X,i,j] \mid & i=j & \text{si } X=\overline{N^\gamma},\ N^\gamma \notin \text{espina}(\gamma) \\ & N^\gamma \stackrel{*}{\Rightarrow} a_i \ldots a_{j-1} & \text{sii } X=\overline{\overline{N^\gamma}},\ N^\gamma \notin \text{espina}(\gamma) \end{array} \right\}$$

donde un $\overline{N^\gamma}$ denota que el nodo N^γ ha sido visitado en la fase descendente del algoritmo, mientras que $\overline{\overline{N^\gamma}}$ denota que el nodo N^γ ha sido visitado en la fase ascendente del algoritmo, esto es, que el subárbol que cuelga de dicho nodo ha sido completamente analizado. Este tipo de ítems se utiliza para representar la parte del análisis que es análoga al caso de gramáticas independientes del contexto, es decir, aquella que consiste simplemente en recorrer un árbol elemental en forma prefija sin realizar adjunciones.

1b El segundo conjunto de ítems se utiliza para determinar la parte que ha sido reconocida de cada producción que forma parte de un árbol elemental, en el caso de que no intervenga directamente en un operación de adjunción. Excluimos entonces de este tipo de ítems aquellos que se refieran a producciones que definen la espina de un árbol auxiliar, puesto que estas siempre están relacionadas con la resolución de una operación de adjunción ya que son las encargadas de propagar la información relativa al pie. Los ítems en este conjunto son de la forma

$$\left\{ [N^\gamma \to \delta \bullet \nu, i, j] \mid \quad N^\gamma \notin \text{espina}(\gamma),\ \delta \stackrel{*}{\Rightarrow} a_i \ldots a_{j-1} \right\}$$

1c El tercer conjunto de ítems es especial en el sentido de que se utiliza únicamente para representar un resultado intermedio cuando se ha terminado de recorrer un árbol auxiliar pero no se conoce todavía si el pie ha sido correctamente predicho. La forma de estos ítems es

$$\left\{ \begin{array}{lll} [N^\gamma \to \delta \bullet \nu, i, j \mid\mid p, q] \mid & N^\gamma \notin \text{espina}(\gamma) \text{ y además:} & \\ & \delta \stackrel{*}{\Rightarrow} a_i \ldots a_{p-1} \mathbf{F}^\beta a_q \ldots a_{j-1} & \text{sii } (p,q) \neq (-,-),\ \beta \in \text{adj}(N^\gamma) \\ & \delta \stackrel{*}{\Rightarrow} a_i \ldots a_{j-1} & \text{sii } (p,q) = (-,-) \end{array} \right\}$$

2a El cuarto conjunto de ítems se utiliza para propagar la información del pie a través de los nodos que forman la espina de un árbol auxiliar β. Los ítems son en este caso de la forma

$$\left\{ \begin{array}{lll} [X,i,j \mid p,q] \mid & i=j,\ p=q=- & \text{sii } X=\overline{N^\beta},\ N^\beta \in \text{espina}(\beta) \\ & N^\beta \stackrel{*}{\Rightarrow} a_i \ldots a_{p-1}\mathbf{F}^\beta a_q \ldots\ldots a_{j-1} & \text{sii } X=\overline{\overline{N^\beta}},\ N^\beta \in \text{espina}(\beta) \end{array} \right\}$$

donde $\overline{N^\beta}$ indica que el nodo N^β de la espina del árbol auxiliar β ha sido visitado en la fase descendente del algoritmo, mientras que $\overline{\overline{N^\beta}}$ indica que dicho nodo ha sido visitado en la fase ascendente del algoritmo y por tanto el subárbol que cuelga de él ha sido analizado, teniendo en cuenta que el pie puede haber una discontinuidad con respecto a la cadena de entrada.

2b Los ítems del quinto conjunto se utilizan para determinar que una parte de una producción en la espina de un árbol auxiliar ha sido reconocida y para propagar, en el caso de que el nodo pie ya haya sido predicho, la información correspondiente a la discontinuidad producida por dicho pie. Concretamente, los ítems son de la forma

$$\left\{ \begin{array}{lll} [N^\beta \to \delta \bullet \nu, i, j \mid p,q] \mid & \beta \in \boldsymbol{A},\ N^\beta \in \text{espina}(\beta) \text{ y además:} & \\ & \delta \stackrel{*}{\Rightarrow} a_i \ldots a_{p-1}\mathbf{F}^\beta a_q \ldots a_{j-1} & \text{sii } (p,q) \neq (-,-) \\ & \delta \stackrel{*}{\Rightarrow} a_i \ldots a_{j-1} & \text{sii } (p,q) = (-,-) \end{array} \right\}$$

2c El sexto conjunto de ítems es especial en el sentido de que se utiliza únicamente para representar un resultado intermedio cuando se ha terminado de recorrer un árbol auxiliar que ha sido adjuntado en la espina de otro árbol auxiliar, y cuando no se conoce todavía si el pie ha sido reconocido correctamente. Los ítems de este conjunto presentan la forma

$$\left\{ \begin{array}{lll} [N^{\beta_1} \to \delta \bullet \nu, i, j \mid -,- \mid p,q] \mid & N^{\beta_1} \in \text{espina}(\beta_1) \text{ y además:} & \\ & \delta \stackrel{*}{\Rightarrow} a_i \ldots a_{p-1}\mathbf{F}^{\beta_2} a_q \ldots a_{j-1} & \text{sii } (p,q) \neq (-,-), \\ & & \beta_2 \in \text{adj}(N^{\beta_1}) \\ & \delta \stackrel{*}{\Rightarrow} a_i \ldots a_{j-1} & \text{sii } (p,q) = (-,-) \end{array} \right\}$$

Si bien la utilización de tipos de ítems especializados puede ser beneficiosa en aras de una optimización del algoritmo, puesto que se evita el almacenamiento de elementos no relevantes en cada punto del proceso de análisis, tiene como inconveniente que hace necesario utilizar un elevado número de pasos deductivos, algunos de los cuales pueden considerarse redundantes, pues realizan la misma operación pero sobre diferentes tipos de ítems. Hemos tratado de reducir en lo posible el número de pasos deductivos, pero respetando lo más fielmente posible la filosofía original del algoritmo. Finalmente, el esquema de análisis **Lang** incluye 19 pasos deductivos, frente a las 28 transiciones LPDA del algoritmo original. La tabla 3.2 muestra la relación entre los pasos deductivos y las acciones del algoritmo tal y como son descritas por Lang [106]. A continuación mostramos el sistema de análisis correspondiente al esquema que estamos tratando.

Esquema de análisis sintáctico 3.8 El sistema de análisis $\mathbb{P}_{\text{Lang}}$ que se corresponde con el algoritmo de análisis sintáctico para TAG descrito por Lang, dada una gramática de adjunción de árboles $\mathcal{T}$ y una cadena de entrada $a_1 \ldots a_n$ se define como sigue:

$$\mathcal{I}^{(1a)}_{\text{Lang}} = \left\{ [X,i,j] \mid X \in \{\overline{N^\gamma}, \overline{\overline{N^\gamma}}\},\ \exists N^\gamma \to \delta \in \mathcal{P}(\gamma),\ \gamma \in \boldsymbol{I} \cup \boldsymbol{A},\ N^\gamma \notin \text{espina}(\gamma),\ 0 \leq i \leq j \right\}$$

$$\mathcal{I}^{(1b)}_{\text{Lang}} = \left\{ [N^\gamma \to \delta \bullet \nu, i, j] \mid N^\gamma \to \delta\nu \in \mathcal{P}(\gamma),\ \gamma \in \boldsymbol{I} \cup \boldsymbol{A},\ N^\gamma \notin \text{espina}(\gamma),\ 0 \leq i \leq j \right\}$$

Pasos deductivos	**Tarea realizada**
$\mathcal{D}^{\text{Init}}_{\text{Lang}}$	*inicialización*
$\mathcal{D}^{\text{Call}}_{\text{Lang}}$ $\mathcal{D}^{\text{Sel}}_{\text{Lang}}$ $\mathcal{D}^{\text{Tab}}_{\text{Lang}}$ $\mathcal{D}^{\text{Ret}}_{\text{Lang}}$	*expansión normal* de un nodo no hoja
$\mathcal{D}^{\text{Scan}}_{\text{Lang}}$	*reconocimiento de terminal*
$\mathcal{D}^{\text{AdjCall}}_{\text{Lang}}$ $\mathcal{D}^{\text{Adjret}}_{\text{Lang}}$ $\mathcal{D}^{\text{FootCall}}_{\text{Lang}}$ $\mathcal{D}^{\text{FootRet}}_{\text{Lang}}$	*decisión de adjunción normal* y *adjunción*
$\mathcal{D}^{\text{SpineCallNormal}}_{\text{Lang}}$ $\mathcal{D}^{\text{SpineRetNormal}}_{\text{Lang}}$ $\mathcal{D}^{\text{SpineCall}}_{\text{Lang}}$ $\mathcal{D}^{\text{SpineSel}}_{\text{Lang}}$ $\mathcal{D}^{\text{SpineTab}}_{\text{Lang}}$ $\mathcal{D}^{\text{SpineRet}}_{\text{Lang}}$	*expansión de la espina*
$\mathcal{D}^{\text{SpineAdjCall}}_{\text{Lang}}$ $\mathcal{D}^{\text{SpineAdjRet}}_{\text{Lang}}$	*decisión de adjunción en la espina* y *adjunción*
$\mathcal{D}^{\text{Foot}}_{\text{Lang}}$	*salto del pie*

Tabla 3.2: Actividades realizadas por los pasos deductivos del esquema **Lang**

$$\mathcal{I}^{(1c)}_{\text{Lang}} = \left\{ \begin{array}{ll} [N^\gamma \to \delta \bullet \nu, i, j \mid\mid p, q] \mid & N^\gamma \to \delta\nu \in \mathcal{P}(\gamma),\ \gamma \in \boldsymbol{I} \cup \boldsymbol{A},\ N^\beta \notin \text{espina}(\beta), \\ & 0 \le i \le p \le q \le j,\ (p,q) \le (i,j) \end{array} \right\}$$

$$\mathcal{I}^{(2a)}_{\text{Lang}} = \left\{ \begin{array}{ll} [X, i, j \mid p, q] \mid & X \in \{\overline{N^\beta}, \overline{\overline{N^\beta}}\},\ \exists N^\beta \to \delta \in \mathcal{P}(\beta),\ N^\beta \in \text{espina}(\beta), \\ & \beta \in \boldsymbol{A},\ 0 \le i \le p \le q \le j,\ (p,q) \le (i,j) \end{array} \right\}$$

$$\mathcal{I}^{(2b)}_{\text{Lang}} = \left\{ \begin{array}{ll} [N^\beta \to \delta \bullet \nu, i, j \mid p, q] \mid & N^\beta \to \delta\nu \in \mathcal{P}(\beta),\ \beta \in \boldsymbol{A},\ N^\beta \in \text{espina}(\beta), \\ & 0 \le i \le p \le q \le j,\ (p,q) \le (i,j) \end{array} \right\}$$

$$\mathcal{I}^{(2c)}_{\text{Lang}} = \left\{ \begin{array}{ll} [N^\beta \to \delta \bullet \nu, i, j \mid -, - \mid p, q] \mid & N^\beta \to \delta\nu \in \mathcal{P}(\beta),\ \beta \in \boldsymbol{A},\ N^\beta \in \text{espina}(\beta), \\ & 0 \le i \le p \le q \le j,\ (p,q) \le (i,j) \end{array} \right\}$$

$$\mathcal{I}_{\text{Lang}} = \mathcal{I}_{\text{Lang}}^{(1a)} \cup \mathcal{I}_{\text{Lang}}^{(1b)} \cup \mathcal{I}_{\text{Lang}}^{(1c)} \cup \mathcal{I}_{\text{Lang}}^{(2a)} \cup \mathcal{I}_{\text{Lang}}^{(2b)} \cup \mathcal{I}_{\text{Lang}}^{(2c)}$$

$$\mathcal{D}_{\text{Lang}}^{\text{Init}} = \frac{}{[\overline{\mathbf{R}^\alpha}, 0, 0]} \quad \alpha \in \boldsymbol{I}$$

$$\mathcal{D}_{\text{Lang}}^{\text{Call}} = \frac{[N^\gamma \to \delta \bullet M^\gamma \nu, i, j]}{[\overline{M^\gamma}, j, j]} \quad \mathbf{nil} \in \text{adj}(M^\gamma)$$

$$\mathcal{D}_{\text{Lang}}^{\text{Sel}} = \frac{[\overline{M^\gamma}, j, j]}{[M^\gamma \to \bullet \delta, j, j]}$$

$$\mathcal{D}_{\text{Lang}}^{\text{Tab}} = \frac{[M^\gamma \to \bullet \delta, j, k],}{[\overline{\overline{M^\gamma}}, j, k]}$$

$$\mathcal{D}_{\text{Lang}}^{\text{Ret}} = \frac{[N^\gamma \to \delta \bullet M^\gamma \nu, i, j],\ [\overline{\overline{M^\gamma}}, j, k]}{[N^\gamma \to \delta M^\gamma \bullet \nu, i, k]} \quad \mathbf{nil} \in \text{adj}(M^\gamma)$$

$$\mathcal{D}_{\text{Lang}}^{\text{Scan}} = \frac{[N^\gamma \to \delta \bullet a\nu, i, j-1],\ [a, j-1, j]}{[N^\gamma \to \delta a \bullet \nu, i, j]}$$

$$\mathcal{D}_{\text{Lang}}^{\text{AdjCall}} = \frac{[N^\gamma \to \delta \bullet M^\gamma \nu, i, j]}{[\overline{\mathbf{R}^\beta}, j, j \mid -, -]} \quad \beta \in \text{adj}(M^\gamma)$$

$$\mathcal{D}_{\text{Lang}}^{\text{AdjRet}} = \frac{[N^\gamma \to \delta \bullet M^\gamma \nu, i, j],\ [\overline{\overline{\mathbf{R}^\beta}}, j, k \mid p, q]}{[N^\gamma \to \delta \bullet M^\gamma \nu, i, k \mid\mid p, q]} \quad \beta \in \text{adj}(M^\gamma),\ M^\gamma \notin \text{espina}(\gamma)$$

$$\mathcal{D}_{\text{Lang}}^{\text{FootCall}} = \frac{[N^\gamma \to \delta \bullet M^\gamma \nu, i, k \mid\mid p, q],}{[\overline{M^\gamma}, p, p]} \quad M^\gamma \notin \text{espina}(\gamma)$$

$$\mathcal{D}_{\text{Lang}}^{\text{FootRet}} = \frac{[N^\gamma \to \delta \bullet M^\gamma \nu, i, k \mid\mid p, q],\ [\overline{\overline{M^\gamma}}, p, q]}{[N^\gamma \to \delta M^\gamma \bullet \nu, i, k]} \quad M^\gamma \notin \text{espina}(\gamma)$$

$$\mathcal{D}_{\text{Lang}}^{\text{SpineCallNormal}} = \frac{[N^\beta \to \delta \bullet M^\beta \nu, i, j \mid p, q]}{[\overline{M^\beta}, j, j]} \quad \begin{array}{l} \beta \in \boldsymbol{A},\ \mathbf{nil} \in \text{adj}(M^\beta), \\ N^\beta \in \text{espina}(\beta),\ M^\beta \notin \text{espina}(\beta), \end{array}$$

$$\mathcal{D}_{\text{Lang}}^{\text{SpineRetNormal}} = \frac{[N^\beta \to \delta \bullet M^\beta \nu, i, j \mid p, q],\ [\overline{\overline{M^\beta}}, j, k]}{[N^\beta \to \delta M^\beta \bullet \nu, i, k \mid p, q]} \quad \begin{array}{l} \beta \in \boldsymbol{A},\ \mathbf{nil} \in \text{adj}(M^\beta), \\ N^\beta \in \text{espina}(\beta),\ M^\beta \notin \text{espina}(\beta), \end{array}$$

$$\mathcal{D}_{\text{Lang}}^{\text{SpineCall}} = \frac{[N^\beta \to \delta \bullet M^\beta \nu, i, j \mid -, -]}{[\overline{M^\beta}, j, j \mid -, -]} \quad \beta \in \boldsymbol{A},\ \mathbf{nil} \in \text{adj}(M^\beta),\ M^\beta \in \text{espina}(\beta)$$

$$\mathcal{D}_{\text{Lang}}^{\text{SpineSel}} = \frac{[\overline{M^\beta}, j, j \mid -, -]}{[M^\beta \to \bullet\delta, j, j \mid -, -]} \quad \beta \in \boldsymbol{A},\ M^\beta \in \text{espina}(\beta)$$

$$\mathcal{D}_{\text{Lang}}^{\text{SpineTab}} = \frac{[M^\beta \to \delta\bullet, j, k \mid p, q]}{[\overline{M^\beta}, j, k \mid p, q]} \quad \beta \in \boldsymbol{A},\ M^\beta \in \text{espina}(\beta)$$

$$\mathcal{D}_{\text{Lang}}^{\text{SpineRet}} = \frac{[N^\beta \to \delta \bullet M^\beta \nu, i, j \mid -, -],\ [\overline{M^\beta}, j, k \mid p, q]}{[N^\beta \to \delta M^\beta \bullet \nu, i, k \mid p, q]} \quad \beta \in \boldsymbol{A},\ \textbf{nil} \in \text{adj}(M^\beta),\ M^\beta \in \text{espina}(\beta)$$

$$\mathcal{D}_{\text{Lang}}^{\text{SpineAdjCall}} = \frac{[N^{\beta_1} \to \delta \bullet M^{\beta_2} \nu, i, j \mid -, -]}{[\overline{\mathbf{R}^{\beta_2}}, j, j \mid -, -]} \quad \beta_2 \in \text{adj}(M^{\beta_1}),\ M^{\beta_1} \in \text{espina}(\beta_1)$$

$$\mathcal{D}_{\text{Lang}}^{\text{SpineAdjRet}} = \frac{[N^{\beta_1} \to \delta \bullet M^{\beta_1} \nu, i, j \mid -, -],\ [\overline{\overline{\mathbf{R}^{\beta_2}}}, j, k \mid p, q]}{[N^{\beta_1} \to \delta \bullet M^{\beta_1} \nu, i, k \mid -, - \mid p, q]} \quad \beta_2 \in \text{adj}(M^{\beta_1}),\ M^{\beta_1} \in \text{espina}(\beta_1)$$

$$\mathcal{D}_{\text{Lang}}^{\text{SpineFootCall}} = \frac{[N^\beta \to \delta \bullet M^\beta \nu, i, k \mid -, - \mid p, q],}{[\overline{M^\beta}, p, p \mid -, -]} \quad M^\beta \in \text{espina}(\beta)$$

$$\mathcal{D}_{\text{Lang}}^{\text{SpineFootRet}} = \frac{[N^\beta \to \delta \bullet M^\beta \nu, i, k \mid -, - \mid p, q],\ [\overline{\overline{M^\beta}}, p, q \mid p', q']}{[N^\beta \to \delta M^\beta \bullet \nu, i, k \mid p', q']} \quad M^\beta \in \text{espina}(\beta)$$

$$\mathcal{D}_{\text{Lang}}^{\text{Foot}} = \frac{[\overline{\mathbf{F}^\beta}, j, j \mid -, -]}{[\overline{\overline{\mathbf{F}^\beta}}, j, k \mid j, k]}$$

$$\begin{aligned}
\mathcal{D}_{\text{Lang}} = \quad & \mathcal{D}_{\text{Lang}}^{\text{Init}} \cup \mathcal{D}_{\text{Lang}}^{\text{Call}} \cup \mathcal{D}_{\text{Lang}}^{\text{Sel}} \cup \mathcal{D}_{\text{Lang}}^{\text{Tab}} \cup \mathcal{D}_{\text{Lang}}^{\text{Ret}} \cup \mathcal{D}_{\text{Lang}}^{\text{Scan}} \cup \\
& \mathcal{D}_{\text{Lang}}^{\text{AdjCall}} \cup \mathcal{D}_{\text{Lang}}^{\text{AdjRet}} \cup \mathcal{D}_{\text{Lang}}^{\text{FootCall}} \cup \mathcal{D}_{\text{Lang}}^{\text{FootRet}} \cup \\
& \mathcal{D}_{\text{Lang}}^{\text{SpineCallNormal}} \cup \mathcal{D}_{\text{Lang}}^{\text{SpineRetNormal}} \cup \mathcal{D}_{\text{Lang}}^{\text{SpineCall}} \cup \mathcal{D}_{\text{Lang}}^{\text{SpineSel}} \cup \mathcal{D}_{\text{Lang}}^{\text{SpineTab}} \cup \mathcal{D}_{\text{Lang}}^{\text{SpineRet}} \cup \\
& \mathcal{D}_{\text{Lang}}^{\text{SpineAdjCall}} \cup \mathcal{D}_{\text{Lang}}^{\text{SpineAdjRet}} \cup \mathcal{D}_{\text{Lang}}^{\text{SpineFootCall}} \cup \mathcal{D}_{\text{Lang}}^{\text{SpineFootRet}} \cup \mathcal{D}_{\text{Lang}}^{\text{Foot}}
\end{aligned}$$

$$\mathcal{F}_{\text{Lang}} = \left\{ \ [\overline{\overline{\mathbf{R}^\alpha}}, 0, n] \mid \alpha \in \boldsymbol{I} \ \right\}$$

§

En interesante observar que ningún paso deductivo tiene más de dos antecedentes, lo cual no resulta sorprendente si tenemos en cuenta que el algoritmo fue originalmente descrito para LPDA, un formalismo cuyas transiciones tienen a lo sumo dos antecedentes. La necesidad de mantener el límite de dos antecedentes probablemente haya sido una de las causas que motivaron la necesidad de definir un número tan elevado de ítems para este esquema de análisis.

Con respecto a la forma de proceder del algoritmo descrito por el esquema de análisis **Lang**, diremos que este comienza el procesamiento de una cadena de entrada prediciendo el nodo raíz de un árbol inicial mediante el paso deductivo $\mathcal{D}_{\text{Lang}}^{\text{Init}}$. Posteriormente, siempre que puede trata de proceder como si las producciones que conforman los árboles fuesen totalmente independientes del contexto: los pasos $\mathcal{D}_{\text{Lang}}^{\text{Call}}$ predicen un nodo, los pasos $\mathcal{D}_{\text{Lang}}^{\text{Sel}}$ seleccionan la producción que tienen a dicho nodo como padre (no terminal del lado izquierdo), los pasos $\mathcal{D}_{\text{Lang}}^{\text{Ret}}$ permiten ir avanzando en el reconocimiento de la parte derecha de dicha producción y los pasos $\mathcal{D}_{\text{Lang}}^{\text{Tab}}$ indican que la producción ha sido totalmente analizada y por consiguiente el nodo padre. Los pasos $\mathcal{D}_{\text{Lang}}^{\text{Scan}}$ se utilizan para reconocer los nodos etiquetados por símbolos terminales.

Las adjunciones en los nodos que no forman parte de una espina se predicen mediante los pasos $\mathcal{D}_{\text{Lang}}^{\text{AdjCall}}$. Una vez terminado el reconocimiento del árbol auxiliar, se genera un ítem intermedio mediante un paso $\mathcal{D}_{\text{Lang}}^{\text{AdjRet}}$ a partir del cual se trata de reconocer, mediante un paso $\mathcal{D}_{\text{Lang}}^{\text{FootPred}}$, si el subárbol que cuelga del nodo en el que se realizó la adjunción reconoce una parte de la cadena de entrada que coincide con la discontinuidad indicada por el pie del árbol auxiliar. Si es así, la adjunción se terminará aplicando un paso $\mathcal{D}_{\text{Lang}}^{\text{FootRet}}$.

Las producciones que forman parte de la espina de un árbol auxiliar reciben un tratamiento especial, pues existe un conjunto de pasos deductivos que realizan un tratamiento análogo al descrito en los párrafos anteriores, pero sólo para este tipo de producciones:

- El paso deductivo $\mathcal{D}_{\text{Lang}}^{\text{SpineCallNormal}}$ se encarga de predecir aquellos no terminales que se encuentran a derecha o izquierda del nodo de la producción que forma parte de la espina, mientras que el paso $\mathcal{D}_{\text{Lang}}^{\text{SpineCallNormal}}$ avanza en el reconocimiento de la producción cuando ya se ha terminado el análisis de uno de dichos nodos.

- Los pasos deductivos $\mathcal{D}_{\text{Lang}}^{\text{SpineCall}}$, $\mathcal{D}_{\text{Lang}}^{\text{SpineSel}}$, $\mathcal{D}_{\text{Lang}}^{\text{SpineTab}}$ y $\mathcal{D}_{\text{Lang}}^{\text{SpineRet}}$ se encargan de expandir el nodo que están en la espina, propagando la información relativa a la discontinuidad producida por el pie.

- Los pasos $\mathcal{D}_{\text{Lang}}^{\text{SpineAdjCall}}$ y $\mathcal{D}_{\text{Lang}}^{\text{SpineAdjRet}}$ se encargan de realizar la operación de adjunción, generando un ítem intermedio a partir del cual se puede aplicar un paso deductivo $\mathcal{D}_{\text{Lang}}^{\text{SpineFootCall}}$ para comenzar el reconocimiento del subárbol correspondiente al pie, proceso que finalizará mediante la aplicación de un paso $\mathcal{D}_{\text{Lang}}^{\text{SpineFootRet}}$.

Por la forma de reconocer el pie en el paso deductivo $\mathcal{D}_{\text{Lang}}^{\text{Foot}}$ este algoritmo es semejante a las versiones del algoritmo de Earley ascendente para TAG, puesto que se predicen todas las posibles posiciones del pie y posteriormente, en la finalización de la adjunción, mediante los pasos deductivos $\mathcal{D}_{\text{Lang}}^{\text{AdjRet}}$, $\mathcal{D}_{\text{Lang}}^{\text{FootCall}}$, $\mathcal{D}_{\text{Lang}}^{\text{FootRet}}$, $\mathcal{D}_{\text{Lang}}^{\text{SpineAdjRet}}$, $\mathcal{D}_{\text{Lang}}^{\text{SpineFootCall}}$ y $\mathcal{D}_{\text{Lang}}^{\text{SpineFootRet}}$, se comprueba qué discontinuidad del pie es compatible con la cadena de entrada analizada.

Sin embargo, el algoritmo también comparte ciertas características de los algoritmos de tipo Earley que no preservan la propiedad del prefijo válido, puesto que realiza un recorrido en orden prefijo de los árboles elementales. Sin embargo dicho orden es alterado por las operaciones de adjunción.

Se podría decir entonces que el algoritmo de Lang está a medio camino entre un Earley ascendente y un Earley sin propiedad del prefijo válido. Ciertamente, fortaleciendo la predicción del pie y utilizando un sólo tipo de ítem obtendríamos un algoritmo similar al descrito por el esquema de análisis sintáctico **E**.

El esquema de análisis sintáctico **Lang** es susceptible de ser simplificado si observamos que ciertos pasos se aplican en cadena. Por ejemplo, los pasos $\mathcal{D}_{\text{Lang}}^{\text{Call}}$ generan ítems que sólo pueden ser utilizados como antecedentes de los pasos $\mathcal{D}^{\mathrm{Sel}}_{\mathrm{Lang}}$ y por tanto dichos pasos pueden ser contraídos para formar un nuevo conjunto de pasos $\mathcal{D}^{\mathrm{Pred}}_{\mathrm{Lang}}$ análogo al conjunto de pasos predictivos del algoritmo de Earley. Análogamente, los pasos $\mathcal{D}^{\mathrm{Tab}}_{\mathrm{Lang}}$ y $\mathcal{D}^{\mathrm{Ret}}_{\mathrm{Lang}}$ pueden contraerse para formar un conjunto de pasos $\mathcal{D}^{\mathrm{Comp}}_{\mathrm{Lang}}$ similar al conjunto de pasos de compleción del algoritmo de Earley. El mismo procedimiento puede ser aplicado a los pasos deductivos que se encargan de realizar la expansión de la espina.

De forma similar, los pasos $\mathcal{D}^{\mathrm{AdjComp}}_{\mathrm{Lang}}$ generan ítems que sólo pueden ser utilizados en los pasos $\mathcal{D}^{\mathrm{FootPred}}_{\mathrm{Lang}}$ y $\mathcal{D}^{\mathrm{FootComp}}_{\mathrm{Lang}}$. Podemos suprimir $\mathcal{D}^{\mathrm{AdjComp}}_{\mathrm{Lang}}$ si en los pasos que tratan el pie añadimos como antecedentes los antecedentes de dicho paso. El mismo procedimiento puede ser aplicado a los pasos deductivos que se encargan de realizar el reconocimiento del pie cuando el nodo de adjunción forma parte de la espina.

También se puede simplificar el conjunto de ítems si tenemos en cuenta que $\mathcal{A}^\gamma$ es equivalente a $N^\gamma \to \bullet\delta$ y que $\overline{\overline{N^\gamma}}$ es equivalente a $N^\gamma \to \delta\bullet$. Como consecuencia, el conjunto 1a de ítems se puede considerar incluido en el conjunto 1b, mientras que el 2a se encuentra incluido en el 2b. Por tanto, aplicando un filtrado estático al esquema de análisis **Lang** podemos eliminar dichos conjuntos de ítems redundantes, modificando adecuadamente los ítems de los pasos deductivos que se vean afectados.

A continuación definimos el sistema de análisis correspondiente al esquema de análisis **Lang′** resultado de aplicar la contracción de pasos deductivos y el filtrado estático mencionados anteriormente.

Esquema de análisis sintáctico 3.9 El sistema de análisis $\mathbb{P}_{\mathrm{Lang}'}$, dada una gramática de adjunción de árboles $\mathcal{T}$ y una cadena de entrada $a_1 \ldots a_n$ se define como sigue:

$$\mathcal{I}_{\mathrm{Lang}'} = \mathcal{I}^{(1b)}_{\mathrm{Lang}} \cup \mathcal{I}^{(2b)}_{\mathrm{Lang}}$$

$$\mathcal{D}^{\mathrm{Init}}_{\mathrm{Lang}'} = \frac{}{[\top \to \bullet\mathbf{R}^\alpha, 0, 0]} \quad \alpha \in \boldsymbol{I}$$

$$\mathcal{D}^{\mathrm{Pred}}_{\mathrm{Lang}'} = \frac{[N^\gamma \to \delta \bullet M^\gamma \nu, i, j]}{[M^\gamma \to \bullet\delta, j, j]} \quad \mathbf{nil} \in \mathrm{adj}(M^\gamma)$$

$$\mathcal{D}^{\mathrm{Comp}}_{\mathrm{Lang}'} = \frac{[N^\gamma \to \delta \bullet M^\gamma \nu, i, j],\ [M^\gamma \to \bullet\delta, j, k]}{[N^\gamma \to \delta M^\gamma \bullet \nu, i, k]} \quad \mathbf{nil} \in \mathrm{adj}(M^\gamma)$$

$$\mathcal{D}^{\mathrm{Scan}}_{\mathrm{Lang}'} = \mathcal{D}^{\mathrm{Scan}}_{\mathrm{Lang}}$$

$$\mathcal{D}^{\mathrm{AdjPred}}_{\mathrm{Lang}'} = \frac{[N^\gamma \to \delta \bullet M^\gamma \nu, i, j]}{[\top \to \bullet\mathbf{R}^\beta, j, j \mid -, -]} \quad \beta \in \mathrm{adj}(M^\gamma)$$

$$\mathcal{D}^{\mathrm{FootPred}}_{\mathrm{Lang}'} = \frac{\begin{array}{c}[N^\gamma \to \delta \bullet M^\gamma \nu, i, j],\\ [\top \to \mathbf{R}^\beta\bullet, j, k \mid p, q],\end{array}}{[M^\gamma \to \bullet\upsilon, p, p]} \quad \beta \in \mathrm{adj}(M^\gamma),\ M^\gamma \notin \mathrm{espina}(\gamma)$$

$$\mathcal{D}^{\text{FootComp}}_{\text{Lang}'} = \frac{\begin{array}{l}[N^\gamma \to \delta \bullet M^\gamma \nu, i, j],\\ [\top \to \mathbf{R}^\beta \bullet, j, k \mid p, q],\\ [M^\gamma \to \upsilon \bullet, p, q]\end{array}}{[N^\gamma \to \delta M^\gamma \bullet \nu, i, k]} \quad \beta \in \text{adj}(M^\gamma),\ M^\gamma \notin \text{espina}(\gamma)$$

$$\mathcal{D}^{\text{SpinePredNormal}}_{\text{Lang}'} = \frac{[N^\beta \to \delta \bullet M^\beta \nu, i, j \mid p, q]}{[M^\beta \to \bullet \upsilon, j, j]} \quad \begin{array}{l}\beta \in \boldsymbol{A},\ \mathbf{nil} \in \text{adj}(M^\beta),\\ N^\beta \in \text{espina}(\beta),\ M^\beta \notin \text{espina}(\beta)\end{array}$$

$$\mathcal{D}^{\text{SpineCompNormal}}_{\text{Lang}'} = \frac{\begin{array}{l}[N^\beta \to \delta \bullet M^\beta \nu, i, j \mid p, q],\\ [M^\beta \to \upsilon \bullet, j, k]\end{array}}{[N^\beta \to \delta M^\beta \bullet \nu, i, k \mid p, q]} \quad \begin{array}{l}\beta \in \boldsymbol{A},\ \mathbf{nil} \in \text{adj}(M^\beta),\\ N^\beta \in \text{espina}(\beta),\ M^\beta \notin \text{espina}(\beta)\end{array}$$

$$\mathcal{D}^{\text{SpinePred}}_{\text{Lang}'} = \frac{[N^\beta \to \delta \bullet M^\beta \nu, i, j \mid -, -]}{[M^\beta \to \bullet \delta, j, j \mid -, -]} \quad \beta \in \boldsymbol{A},\ \mathbf{nil} \in \text{adj}(M^\beta),\ M^\beta \in \text{espina}(\beta)$$

$$\mathcal{D}^{\text{SpineComp}}_{\text{Lang}'} = \frac{\begin{array}{l}[N^\beta \to \delta \bullet M^\beta \nu, i, j \mid -, -],\\ [M^\beta \to \delta \bullet, j, k \mid p, q]\end{array}}{[N^\beta \to \delta M^\beta \bullet \nu, i, k \mid p, q]} \quad \beta \in \boldsymbol{A},\ \mathbf{nil} \in \text{adj}(M^\beta),\ M^\beta \in \text{espina}(\beta)$$

$$\mathcal{D}^{\text{SpineAdjPred}}_{\text{Lang}'} = \frac{[N^{\beta_1} \to \delta \bullet M^{\beta_2} \nu, i, j \mid -, -]}{[\top \to \bullet \mathbf{R}^{\beta_2}, j, j \mid -, -]} \quad \beta_2 \in \text{adj}(M^{\beta_1}),\ M^{\beta_1} \in \text{espina}(\beta_1)$$

$$\mathcal{D}^{\text{SpineFootPred}}_{\text{Lang}'} = \frac{\begin{array}{l}[N^{\beta_1} \to \delta \bullet M^{\beta_1} \nu, i, j \mid -, -],\\ [\top \to \mathbf{R}^{\beta_2} \bullet, j, k \mid p, q]\end{array}}{[M^{\beta_1} \to \bullet \upsilon, p, p \mid -, -]} \quad \beta_2 \in \text{adj}(M^{\beta_1}),\ M^{\beta_1} \in \text{espina}(\beta_1)$$

$$\mathcal{D}^{\text{SpineFootComp}}_{\text{Lang}'} = \frac{\begin{array}{l}[N^{\beta_1} \to \delta \bullet M^{\beta_1} \nu, i, j \mid -, -],\\ [\top \to \mathbf{R}^{\beta_2} \bullet, j, k \mid p, q],\\ [M^{\beta_1} \to \upsilon \bullet, p, q \mid p', q']\end{array}}{[N^{\beta_1} \to \delta M^{\beta_1} \bullet \nu, i, k \mid p', q']} \quad \beta_2 \in \text{adj}(M^{\beta_1}),\ M^{\beta_1} \in \text{espina}(\beta_1)$$

$$\mathcal{D}^{\text{Foot}}_{\text{Lang}'} = \frac{[\mathbf{F}^\beta \to \bullet \bot, j, j \mid -, -]}{[\mathbf{F}^\beta \to \bot \bullet, j, k \mid j, k]}$$

$$\begin{aligned}\mathcal{D}_{\text{Lang}'} = \ & \mathcal{D}^{\text{Init}}_{\text{Lang}'} \cup \mathcal{D}^{\text{Pred}}_{\text{Lang}'} \cup \mathcal{D}^{\text{Comp}}_{\text{Lang}'} \cup \mathcal{D}^{\text{Scan}}_{\text{Lang}'} \cup \\ & \mathcal{D}^{\text{AdjPred}}_{\text{Lang}'} \cup \mathcal{D}^{\text{FootPred}}_{\text{Lang}'} \cup \mathcal{D}^{\text{FootComp}}_{\text{Lang}'} \cup \\ & \mathcal{D}^{\text{SpinePredNormal}}_{\text{Lang}'} \cup \mathcal{D}^{\text{SpineCompNormal}}_{\text{Lang}'} \cup \mathcal{D}^{\text{SpinePred}}_{\text{Lang}'} \cup \mathcal{D}^{\text{SpineComp}}_{\text{Lang}'} \cup \\ & \mathcal{D}^{\text{SpineAdjPred}}_{\text{Lang}'} \cup \mathcal{D}^{\text{SpineFootPred}}_{\text{Lang}'} \cup \mathcal{D}^{\text{SpineFootComp}}_{\text{Lang}'} \cup \mathcal{D}^{\text{Foot}}_{\text{Lang}'}\end{aligned}$$

$$\mathcal{F}_{\text{Lang}'} = \left\{ \ [\top \to \mathbf{R}^\alpha \bullet, 0, n] \mid \alpha \in \boldsymbol{I} \ \right\}$$

§

Proposición 3.7 **Lang** $\stackrel{\text{sc}}{\Longrightarrow}\stackrel{\text{sf}}{\Longrightarrow}$ **Lang**$'$.

El esquema de análisis **Lang**$'$ es muy similar al esquema de análisis **E**, radicando las diferencias en los dos puntos siguientes:

- La utilización de diferentes tipos de ítems por parte de **Lang**$'$. Efectivamente, podemos considerar que los ítems del tipo 1b son casos particulares del tipo 2b en los que los últimos índices toman el valor $-$, reduciendo así los diferentes tipos de ítems a 2b y 2c. Con ello se lograría también contraer numerosos pasos de análisis, pues no sería necesario distinguir entre operaciones realizadas sobre producciones que forman la espina y operaciones realizadas sobre otro tipo de producciones.

- El reconocimiento del subárbol que cuelga del pie en un árbol de derivación se realiza de forma muy particular en **Lang**$'$, prediciendo todas las posibles posiciones del nodo pie y comprobando al finalizar la adjunción si alguna de dichas predicciones es compatible con la cadena de entrada. En la figura 3.7 se muestra una representación gráfica de la predicción del pie en el caso de que el nodo de adjunción forme parte de la espina, por ser este el caso más complejo. En la figura 3.8 se muestra una representación gráfica de la finalización del reconocimiento del pie.

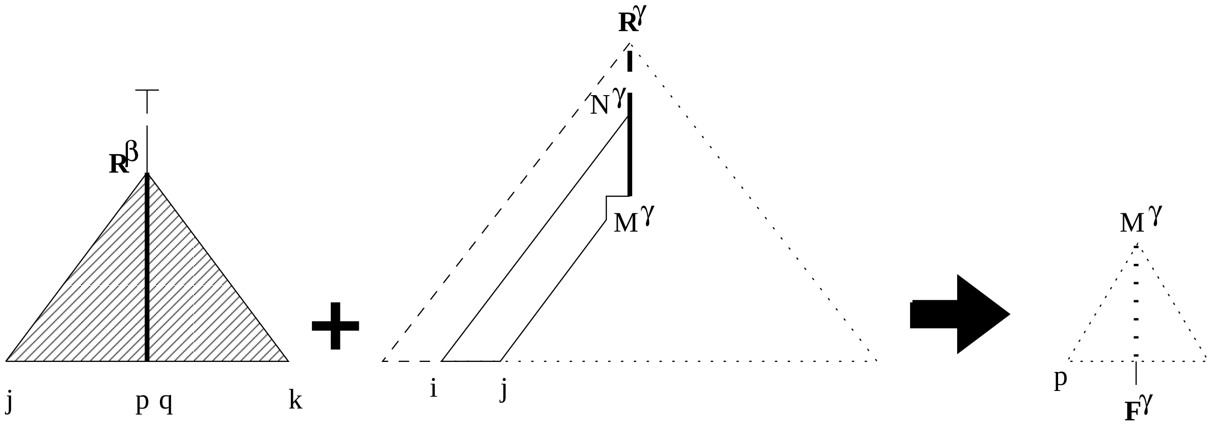

Figura 3.7: Descripción gráfica de la aplicación de un paso $\mathcal{D}_{\text{Lang}}^{\text{SpineFootPred}}$

3.9.2 Algoritmos bidireccionales

Lavelli y Satta presentan en [109] un algoritmo de tipo *head-corner* para el análisis de gramáticas de adjunción lexicalizadas que toma como *núcleo* o *head* el ancla léxica de cada árbol elemental. Dicho algoritmo puede considerarse como una extensión bidireccional de un algoritmo de tipo Earley para TAG en el cual el paso inicial reconoce las anclas léxicas y posteriormente trata de expandir el árbol elemental tanto a izquierda como a derecha[10]. Durante estas expansiones el algoritmo realiza predicciones descendentes con el fin de reconocer los nodos que no se encuentran en el camino de la raíz al ancla del árbol que se está analizando. La motivación lingüística de este algoritmo radica en que el ancla de cada árbol es el nodo del árbol que aporta mayor información acerca de la estructura sintáctica que dicho aŕbol representa y por tanto parece conveniente comenzar el análisis por dichos nodos.

[10]Una consecuencia de la bidireccionalidad del análisis es que los ítems deberán contener producciones con dos puntos en lugar de producciones con punto simples.

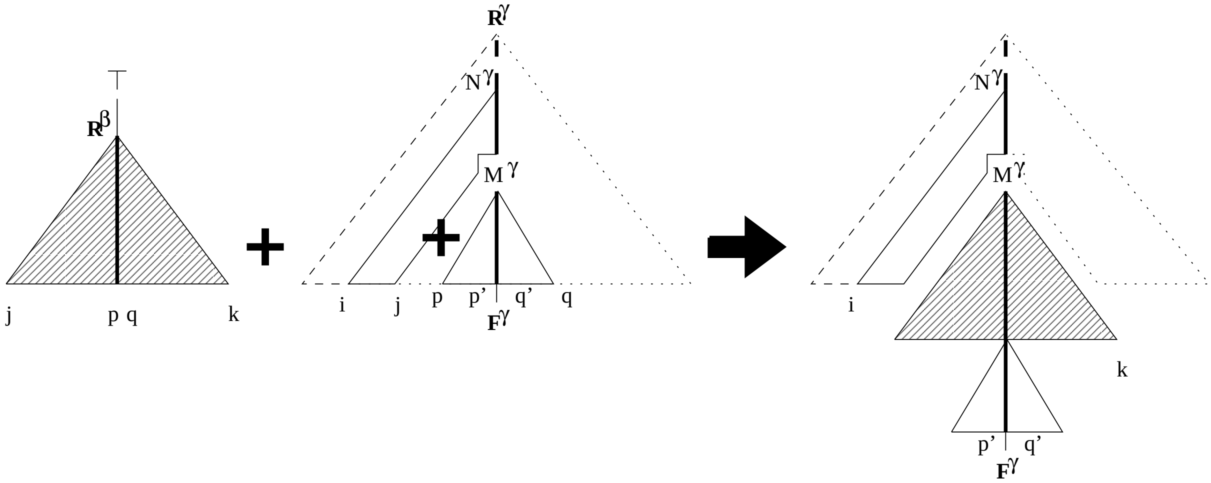

Figura 3.8: Descripción gráfica de la aplicación de un paso $\mathcal{D}_{\text{Lang}}^{\text{SpineFootComp}}$

Van Noord presenta en [205] otro algoritmo *head-corner* para gramáticas de adjunción de árboles lexicalizadas[11]. La singularidad con respecto al de Lavelli y Satta es que este algoritmo considera los nodos pie como núcleo de los árboles auxiliares[12]. Esta elección de los núcleos viene determinada por el funcionamiento del algoritmo, que se define como dirigido por el núcleo cuando se trata de analizar árboles iniciales y dirigido por el pie cuando se trata de analizar árboles auxiliares. A diferencia del algoritmo de Lavelli y Satta, el algoritmo de van Noord procede de modo totalmente ascendente, sin realizar predicciones. Otra diferencia importante es que el reconocimiento de los árboles auxiliares no se inicia por el reconocimiento de su ancla, sino bajo la demanda de una operación de adjunción, una consecuencia de la consideración de los nodos pie como núcleo de los árboles auxiliares. Sarkar presenta en [165] los resultados obtenidos en diversos experimentos realizados con el analizador de Van Noord.

Díaz Madrigal et al. presentan en [59, 62] un algoritmo bidireccional ascendente para el análisis de gramáticas de adjunción de árboles lexicalizadas inspirado en el algoritmo de De Vreugh y Honig para gramáticas independientes del contexto [57, 189]. El algoritmo resultante es similar al propuesto por Van Noord, con la importante salvedad de que todo nodo de un árbol elemental etiquetado por un terminal, así como el nodo pie de los árboles auxiliares, es considerado núcleo. Los mismos autores presentan en [61, 58] diversas optimizaciones, relativas al tratamiento de los nodos de sustitución y de los nodos etiquetados por ϵ, que mejoran el comportamiento práctico del algoritmo. El algoritmo resultante es comparado en [60] con otros algoritmos tabulares para el análisis de TAG.

Evans y Weir proponen en [73] un algoritmo bidireccional para el análisis sintáctico de gramáticas de adjunción lexicalizadas en el cual el recorrido de un árbol elemental comienza por el ancla y asciende nivel a nivel examinando primero los nodos a la izquierda del camino desde el ancla a la raíz y después los nodos a la derecha de dicho camino. La principal aportación de este algoritmo consiste en la codificación en forma de autómata finito del recorrido de los árboles elementales. Posteriormente dicho autómata finito es utilizado para construir un conjunto de ítems mediante un algoritmo que puede considerarse una versión bidireccional del algoritmo descrito por el esquema de análisis sintáctico $\mathbf{buE_1}$.

Lopez presenta en [114, 112] un algoritmo bidireccional ascendente para gramáticas de ad-

[11]Van Noord describe en [204] una versión anterior del mismo algoritmo que involucra una transformación de TAG a HPSG.

[12]El núcleo de los árboles iniciales sigue siendo el ancla.

junción de árboles lexicalizadas. La principal innovación del algoritmo consiste en la definición de los *caminos conectados*. Dado un árbol elemental, cada uno de los caminos conectados representa una parte de dicho árbol que puede ser recorrida propagando una posición determinada de la cadena de entrada. Por tanto, un camino conectado comienza en un nodo raíz, un nodo de sustitución, un nodo ancla o un nodo pie y termina en el siguiente nodo raíz, nodo de sustitución, nodo ancla o nodo pie. El algoritmo maneja ítems de la forma $[i, j, \Gamma_L, \Gamma_R, p, q, star]$, donde i y j representan las posiciones de un segmento de la cadena de entrada, Γ_L y Γ_R son caminos conectados, p y q son las posiciones de la cadena de entrada correspondientes al pie y $star$ es la dirección del nodo de adjunción predicho más cercano. Los ítems se combinan mediante pasos deductivos similares a los utilizados en el esquema Earley ascendente $\mathbf{buE_1}$ con la salvedad de que los caminos conectados reducen el número de ítems que es necesario generar para recorrer un árbol. La complejidad temporal permanece en $\mathcal{O}(n^6)$, donde n es la longitud de la cadena de entrada, pero se reduce el factor de la constante asociada al tamaño de la gramática. El algoritmo ha sido adaptado al tratamiento de frases orales espontáneas mediante la incorporación de pasos deductivos que tratan diversos fenómenos que aparecen frecuentemente en ese contexto, como son la duda, la elipsis y la autoreparación [113, 115, 112].

Halber presenta en [78] un algoritmo bidireccional basado en *chart* para el análisis de gramáticas de adjunción de árboles lexicalizadas. Este algoritmo puede verse como una extensión bidireccional del algoritmo descrito por el esquema de análisis **E** con el añadido de los *enlaces gramaticales* que son almacenados en cada ítem y que son utilizados para el cálculo de diferentes tipos de probabilidades sobre los árboles generados durante el proceso de análisis.

3.9.3 Algoritmos de varias fases

Existe un conjunto de algoritmos de análisis sintáctico de gramáticas de adjunción que proceden en varias fases, la primera de las cuales suele ser un análisis del esqueleto independiente del contexto y las siguientes dedicadas a la extracción de los árboles de derivación de TAG de entre la salida de la primera fase.

Harbusch presenta en [80] un algoritmo para el análisis sintáctico de TAG que pretende trabajar con una complejidad en el peor caso de $\mathcal{O}(n^4 \log n)$, resultado cuestionado posteriormente [172, 148]. El algoritmo de Harbusch trabaja con gramáticas de adjunción en las que cada nodo no terminal tiene a lo sumo dos descendientes y en las que no se permite que ningún nodo esté etiquetado por ϵ, a excepción de un único nodo de un árbol inicial especialmente concebido para la derivación de la cadena vacía. Además, cada árbol elemental debe producir al menos un terminal, excepto el árbol inicial especial para ϵ. El proceso de análisis se divide en las siete fases siguientes:

1. Paso especial para el tratamiento de la cadena vacía.

2. Definición de números de nodo que identifiquen unívocamente cada uno de los nodos de los árboles elementales de la gramática.

3. Obtención del esqueleto independiente de contexto de la gramática de adjunción que se pretendiese analizar.

4. Aplicación el algoritmo CYK para gramáticas independientes del contexto sobre el esqueleto independiente del contexto. Además de las acciones habituales realizadas en un algoritmo CYK, también se realizan ciertos cálculos relativos a la posición del pie en los árboles auxiliares.

5. Obtención de *números de nodo extendidos* para todos los ítems resultantes de la aplicación del algoritmo CYK, con los que se pretende representar la información concerniente a los árboles auxiliares necesaria en la resolución de las operaciones de adjunción, típicamente información relativa a los nodos pie.

6. Poda de aquellos árboles que no sean válidos por no respetar las restricciones propias de la formación de árboles en las gramáticas de adjunción de árboles.

7. Obtención del resultado, determinando si la cadena de entrada pertenece o no al lenguaje definido por la gramática de adjunción utilizada para el análisis.

Poller describe en [148] un algoritmo incremental de izquierda a derecha, utilizado para analizar una variante de TAG con la misma capacidad generativa denominada TAG(LD/TLP) en el que los árboles elementales se descomponen en árboles elementales de descripción de dominancia local (LD) y relaciones de precedencia lineal (TLP). El proceso de análisis se reparte entre las dos fases siguientes:

1. Análisis del esqueleto independiente del contexto de la gramática de adjunción mediante una variación del algoritmo de Earley [69] que genera ítems con punteros adicionales con el fin de permitir navegar entre las derivaciones independientes del contexto para, en una segunda fase, obtener los árboles derivados de acuerdo con la gramática de adjunción.

2. Recuperación de los árboles TAG que no tienen operaciones de adjunción pendientes. Para ello se van reconociendo de modo ascendente los árboles elementales que forman parte de un árbol de derivación.

Poller y Becker presentan en [149] una versión simplificada del algoritmo anterior para TAG limpias[13]. El primer paso consiste en la aplicación del algoritmo de Earley estándar sobre el esqueleto independiente del contexto de la gramática de adjunción a analizar. El segundo paso consiste en un proceso iterativo de eliminación de árboles adjuntados a partir de los ítems generados en el primer paso, lo que se corresponde con una estrategia ascendente para la obtención del árbol derivado. Como los autores señalan, la eficiencia práctica del algoritmo varía según el método elegido para la obtención del esqueleto independiente de contexto de la TAG a analizar. Se proponen los dos métodos siguientes.:

1. Considerar únicamente las etiquetas de los nodos para generar las producciones independientes del contexto.

2. Considerar la dirección de los nodos para construir dichas producciones. En este caso, durante la aplicación del algoritmo de Earley se pueden chequear algunas restricciones de adjunción durante la predicción de no-terminales en el algoritmo de Earley.

Con el primer método se obtiene una mayor compartición en la aplicación de la primera fase del algoritmo de análisis mientras que con el segundo se produce un mejor filtrado de derivaciones incorrectas con respecto a la gramática de adjunción.

[13]Una TAG es limpia si los nodos raíz de los árboles elementales y los nodos pie de los árboles auxiliares tienen restricciones de adjunción nulas. Cualquier TAG puede ser transformada fácilmente en una TAG limpia *casi* fuertemente equivalente [149] añadiendo un nodo adicional a cada árbol elemental que tenga como hijo el nodo raíz de dicho árbol, con la misma etiqueta que el nodo raíz y restricción de adjunción nula, más un nodo adicional por cada árbol auxiliar que tenga como padre al nodo pie de dicho árbol, con la misma etiqueta que el nodo pie y restricción de adjunción nula.

3.9.4 Algoritmos basados en LIG

Algunos autores consideran que aunque las gramáticas de adjunción son un formalismo adecuado para la descripción de fenómenos lingüísticos, su análisis sintáctico puede realizarse de modo más eficiente aplicando una transformación previa a otro formalismo más adecuado para el tratamiento computacional. Generalmente, el formalismo elegido suele ser el de las gramáticas lineales de índices (LIG) [75].

En esta línea, Vijay-Shanker y Weir describen en [213] y [214] dos métodos muy similares de transformación de gramáticas de adjunción de árboles en gramáticas lineales de índices, a las que posteriormente aplican un algoritmo tabular de tipo CYK, lo que conlleva una limitación en la forma de las gramáticas de adjunción que pueden ser tratadas, puesto que el número de hijos de cada nodo no puede ser mayor que dos. Este inconveniente puede evitarse aplicando el método general de transformación de TAG a LIG descrito en la sección 2.4.3, el cual no establece ninguna restricción en el número de hijos de cada nodo.

Para permitir sustitución simultánea, Schabes y Shieber utilizan en [175] un algoritmo de tipo Earley para LIG que no es general, sólo apto para la transformación de TAG a LIG propuesta.

Schabes muestra en [170] cómo analizar TAG lexicalizadas estocásticas transformándolas a LIG estocásticas y aplicando un algoritmo de tipo CYK para analizar estas últimas.

Díaz Madrigal y Toro Bonilla presentan en [66] un algoritmo de análisis sintáctico de TAG basado en una representación plana de los árboles elementales [64] implementable en Prolog que realiza implícitamente una conversión a LIG, puesto que en cada momento del proceso de análisis se tienen en cuenta la pila de adjunciones asociada al nodo que se está tratando y justamente, cuando se transforma una TAG a LIG se pretende que la pila de índices asociada a un nodo represente las adjunciones pendientes en la espina de la que forma parte dicho nodo.

De Kercadio sigue un enfoque similar en [51], utilizando una representación plana de los árboles elementales muy similar a la presentada por Díaz Madrigal et al. en [64] e introduciendo una pila en los ítems que permite correlacionar adecuadamente las predicciones de adjunción con el reconocimiento de los nodos pie. El algoritmo resultante satisface la propiedad del prefijo válido.

3.9.5 Algoritmos LR

Los algoritmos de análisis de la familia LR [6] pueden considerarse entre los algoritmos de análisis más potentes para gramáticas independientes del contexto. Su potencia se basa en la precompilación de información descendente de la gramática en un conjunto de tablas de análisis sintáctico que posteriormente serán utilizadas para guiar un analizador sintáctico ascendente durante el análisis de cada cadena de entrada.

Algunos autores han intentado extender los algoritmos LR a la clase de las gramáticas de adjunción. Sin embargo la tarea es dificultosa puesto que la información compilada en las tablas de análisis LR para gramáticas independientes del contexto se corresponde básicamente con la información proporcionada por los pasos predictivos del algoritmo de Earley [11, 12]. Sin embargo, en el caso de gramáticas de adjunción de árboles los algoritmos de tipo Earley realizan dos tipos adicionales de predicción acerca de la operación de adjunción y del reconocimiento del pie.

A continuación presentamos los distintos intentos de extender la técnica LR a las gramáticas de adjunción de árboles que han sido publicados hasta el momento. Normalmente, estos algoritmos no están basados en técnicas tabulares, sino que trabajan directamente sobre cierto tipo de extensión de los autómatas a pila. Sin embargo, los autores de los distintos algoritmos argumentan que es posible extender al caso de TAG las técnicas de tabulación diseñadas para los algoritmos LR que trabajan con gramáticas independientes del contexto [199, 12] con el fin de poder tratar gramáticas ambiguas.

El algoritmo LR de Schabes y Vijay-Shanker

Schabes y Vijay-Shanker [176, 168] han propuesto un algoritmo de análisis sintáctico para gramáticas de adjunción de árboles deterministas[14] basado en un extensión de la técnica de análisis LR para lenguajes independientes del contexto. En este caso, en lugar de utilizar un autómata a pila como mecanismo operacional, se utiliza la versión ascendente del autómata a pila embebido, denominada BEPDA[15].

Un analizador sintáctico LR para TAG consiste de una cadena de entrada, una salida, una secuencia de pilas, un programa conductor y una tabla de análisis sintáctico con tres partes (ACCIÓN, IR_A$_{derecha}$, IR_A$_{foot}$). Al igual que en el caso de los analizadores LR para lenguajes independientes del contexto, el programa de análisis sintáctico es el mismo para todos los analizadores LR, de modo que para analizar una gramática u otra sólo es preciso cambiar las tablas de análisis. El programa de análisis lee la cadena de entrada de izquierda a derecha, un carácter cada vez, y utiliza la secuencia de pilas para almacenar los estados.

En los analizadores LR para lenguajes independientes del contexto, cada estado del autómata finito utilizado para guiar el proceso de análisis se construye mediante la cerradura de un conjunto de producciones con punto, cerradura que se realiza aplicando sucesivamente el paso predictivo del algoritmo de Earley a cada una de las producciones del conjunto. En el caso de los analizadores LR para TAG también existe una fuerte relación entre estos y los algoritmos de análisis de tipo Earley para TAG sin la propiedad del prefijo válido, concretamente con la versión definida mediante el esquema de análisis $\mathbf{E}$, de tal modo que cada estado del autómata finito asociado con un analizador LR para TAG se define como la cerradura bajo los pasos deductivos $\mathcal{D}_{\mathrm{E}}^{\mathrm{AdjPred}}$, $\mathcal{D}_{\mathrm{E}}^{\mathrm{FootPred}}$, $\mathcal{D}_{\mathrm{E}}^{\mathrm{Pred}}$ y $\mathcal{D}_{\mathrm{E}}^{\mathrm{Comp}}$ de un conjunto de producciones con punto, prescindiendo de los índices sobre la cadena de entrada. Una consecuencia de aplicar la operación cerradura mediante este conjunto de operaciones, será que los algoritmos LR para TAG no posean la propiedad del prefijo válido, pues mediante $\mathcal{D}_{\mathrm{E}}^{\mathrm{FootPred}}$ se puede predecir el reconocimiento de un subárbol que no se corresponde con ninguna operación de adjunción.

Con respecto a las transiciones entre estados, existen tres tipos diferentes de transiciones que pueden partir de un estado S_i:

1. *Desplazamiento*: si $N^\gamma \to \delta \bullet a\nu \in S_i$ y $a \in V_T$ entonces hay una transición etiquetada por a hacia el estado S_j, que contiene la producción $N^\gamma \to \delta a \bullet \nu$. Dicha transición se denota por $S_j \in \text{trans}(S_i, a)$.

2. *Adjunción*: si $N^\gamma \to \delta \bullet M^\gamma \nu \in S_i$ y $\beta \in \text{adj}(M^\gamma)$, entonces $S_j \in \text{trans}(S_i, \beta_{derecha})$, tal que el estado S_j contiene la producción $N^\gamma \to \delta M^\gamma \bullet \nu$.

3. *Pie*: si $\mathbf{F}^\beta \to \bullet\bot \in S_i$, entonces hay una transición $S_j \in \text{trans}(S_i, \mathbf{F}^\beta)$, tal que el estado S_j contiene la producción $\mathbf{F}^\beta \to \bot\bullet$.

Las partes IR_A$_{derecha}$ y IR_A$_{foot}$ de la tabla de análisis sintáctico almacenan las transiciones etiquetadas por $\beta_{derecha}$ y por $\mathbf{F}^\beta$, respectivamente.

[14]Además de ser deterministas, las gramáticas deben satisfacer la condición de que cada nodo tiene o bien una restricción de adjunción nula o bien una restricción de adjunción obligatoria, aunque este último condicionante no limita la capacidad de reconocimiento puesto que todo lenguaje de adjunción puede describirse mediante una gramática escrita en esta forma.

[15]Los BEPDA se estudian en detalle en el capítulo 7.

Con respecto a la tabla de análisis sintáctico, esta se construye a partir de la información proporcionada por el autómata finito. Definimos $\text{trans}(i, x)$ como el conjunto de estados alcanzables desde el esto i mediante la transición etiquetada por x. Los cinco tipos de acciones almacenadas en $\text{ACCIÓN}(S_i, a)$ son:

1. *Desplazar al estado S_j*: se aplica si y sólo si $j \in \text{trans}(S_i, a)$.
2. *Restaurar derecha de $M^\gamma \to \upsilon$*: se aplica si y sólo si $N^\gamma \to \delta \bullet M^\gamma \nu \in S_i$ y M^γ tiene una restricción de adjunción obligatoria.
3. *Reducir raíz de β*: se aplica si y sólo si en el estado S_i hay una producción $\top \to \mathbf{R}^\beta \bullet$ y $\beta \in \boldsymbol{A}$.
4. *Aceptar*: ocurre si y sólo si el siguiente elemento en la cadena de entrada es el marcador de fin de cadena y en el estado S_i hay una producción $\top \to \mathbf{R}^\alpha \bullet$ y $\alpha \in \boldsymbol{I}$.
5. *Error*: se aplica si y sólo si ninguna de las otras acciones es aplicable.

El programa conductor procede leyendo el componente léxico a de la cadena de entrada que se está tratando y el estado S_i que está en la cima de la secuencia de pilas y consultando la entrada $\text{ACCIÓN}(S_i, a)$ de la tabla de acciones. Los posibles movimientos vienen dados por:

1. $\text{ACCIÓN}(S_i, a)$=*Desplazar al estado S_j*. En este caso se crea una nueva pila conteniendo S_j. Esta acción se corresponde con el paso deductivo $\mathcal{D}_{\text{E}}^{\text{Scan}}$ del esquema de análisis **E**.
2. $\text{ACCIÓN}(S_i, a)$=*Restaurar derecha de $M^\gamma \to \upsilon$*. Esta acción se corresponde con el paso deductivo $\mathcal{D}_{\text{E}}^{\text{FootComp}}$ del esquema de análisis **E**. Se trata por tanto de continuar el análisis a partir del árbol auxiliar β que ha sido adjuntado al nodo M^γ cuyo subárbol acabamos de reconocer. Se pueden dar dos casos:
   (a) Si $\gamma \in \boldsymbol{I}$ o bien $\gamma \in \boldsymbol{A}$ pero el subárbol enraizado por M^γ no incluye al nodo pie de γ, se toman las k pilas unitarias superiores de la secuencia de pilas, donde k es el número de terminales en la frontera del subárbol enraizado por M^γ, que se unen para formar una sola pila, y se sitúa una nueva pila en la cima conteniendo S_m =$\text{IR\_A}_{foot}(S_k, \beta)$, donde S_k es el estado contenido en la pila situada inmediatamente debajo de las k pilas unidas y $\beta \in \text{adj}(M^\gamma)$.
   (b) Si $\gamma \in \boldsymbol{A}$ y el subárbol enraizado por M^γ incluye al nodo pie de γ, la operación es más compleja: si k_1 y k_2 son respectivamente el número de terminales a la derecha y a la izquierda del pie de γ, debemos unir los elementos de la pila en posición $k_1 + 1$ (que corresponden al análisis del pie) con los elementos de las k_2 pilas unitarias bajo esa pila (que contienen los terminales de la frontera a la izquierda del nodo pie) y los elementos de las k_1 pilas unitarias superiores (que contienen los terminales de la frontera a la derecha del pie), situando una nueva pila unitaria conteniendo $m \in \text{IR\_A}_{foot}(S_{k'}, \beta)$ en la cima de la secuencia de pilas, donde $S_{k'}$ es el elemento de la pila unitaria situada inmediatamente debajo de todas las pilas unidas.
3. $\text{ACCIÓN}(S_i, a)$=*Reducir raíz de β*. Esta acción se corresponde con el paso deductivo $\mathcal{D}_{\text{E}}^{\text{AdjComp}}$ del esquema de análisis **E**. Se realiza, por tanto, cuando el analizador sintáctico ha terminado el análisis del árbol auxiliar β y debe eliminarse la información acerca de β que hay en la pila para continuar el análisis en el nodo en el cual se realizó la adjunción. Denotaremos mediante k_1 y k_2 al número de terminales a la derecha y a la izquierda del

pie de β y mediante k_3 al número de terminales a la izquierda del nodo pie de β que son subsumidos por el nodo de la espina más cercano a la raíz que tiene una restricción de adjunción obligatoria. Los $k_1 + k_3 + 1$ símbolos de la pila en la cima de la secuencia de pilas son extraídos y las $k_2 - k_3$ pilas unitarias que se encuentran bajo la nueva pila de la cima son eliminadas. Sea S_j el elemento de la cima de la nueva secuencia de pilas. Dicho elemento es extraído, mientras una nueva pila unitaria es situada en la cima, cuyo contenido es $m \in$IR_A$_{derecha}(S_j, \beta)$.

Lamentablemente, el algoritmo es incorrecto ya que el último paso descrito no realiza todas las comprobaciones pertinentes [99, 123]. Puesto que el paso deductivo $\mathcal{D}_{\mathrm{E}}^{\mathrm{FootPred}}$ se utiliza para realizar la cerradura de los estados, para cada pie se predicen todos los posibles nodos de *cualquier* árbol auxiliar en el que se pueda realizar la adjunción de β. Al aplicar la operación *Restaura Derecha de* $M^\gamma \rightarrow \upsilon$ se supone que M^γ es el nodo sobre el cual se ha realizado la adjunción de β, pero esto no tiene porqué ser cierto. La primera consecuencia de este hecho es la pérdida de la propiedad del prefijo válido pero una consecuencia más importante es que, al no realizarse ninguna comprobación adicional durante la operación *Reduce Raíz de* β puede que demos por terminada la adjunción de β a un árbol α de modo incorrecto, puesto que estaremos asociado al pie un subárbol incorrecto. Comprobar el número de terminales no es suficiente para garantizar que el subárbol que se ha utilizado para reconocer el pie del árbol auxiliar β coincide con el árbol que se está utilizando para terminar la adjunción de β.

El algoritmo LR de Kinyon

Kinyon ha propuesto un nuevo algoritmo de análisis LR para gramáticas de adjunción de árboles lexicalizadas (LTAG) que toma como punto de partida el algoritmo de Schabes y Vijay-Shanker pero que presenta las siguientes modificaciones respecto a este:

- Admite múltiples acciones en cada entrada de la tabla, permitiendo realizar análisis no deterministas.
- Introduce un nuevo tipo de reducciones *Reducir inicial* α sobre los árboles iniciales con el fin de integrar la operación de sustitución utilizada en LTAG.
- Aplica un filtro en la tabla de análisis para sacar partido de la lexicalización de la gramática, puesto que sólo se considerará aquella parte de la tabla que sea consistente con los árboles de análisis anclados en los componentes léxicos de la cadena de entrada.
- La operación de cerradura de los estados del autómata finito utilizado para construir las tablas de análisis sintáctico asocia a cada producción la pila de los nodos de adjunción antecesores que han sido predichos al llegar al pie de un árbol auxiliar. Por tanto, dicha operación cerradura trabaja sobre *ítems* de la forma $[N^\gamma \rightarrow \delta \bullet \nu, stars]$, donde *stars* representa la pila de nodos de adjunción. Mediante está pila se corregirá el comportamiento defectuoso que tenía el algoritmo de Schabes y Vijay-Shanker.

La inclusión de una pila en los ítems que conforman la cerradura de los estados LR constituye el principal inconveniente de este algoritmo, puesto que al no estar limitado el tamaño de dicha pila ni por el tamaño de la gramática ni por el de la cadena de entrada, la complejidad tanto espacial como temporal del proceso de construcción del autómata LR aumenta considerablemente. De hecho, la operación cerradura puede no terminar para algunas gramáticas.

A continuación pasamos a describir las particularidades del algoritmo LR de Kinyon con respecto al de Schabes y Vijay-Shanker. La operación cerradura utilizada para crear los estados se realiza aplicando los siguientes pasos a partir de un estado inicial que contiene el ítem $[\top \rightarrow \bullet \mathbf{R}^\alpha, \{\ \}]$, donde α es un árbol inicial:

1. AdjPred: Si $[N^\gamma \to \delta \bullet M^\gamma \nu, \mathit{stars}] \in S_i$ y $\beta \in \mathrm{adj}(M^\gamma)$ entonces
   - si $M^\gamma \in \mathrm{espina}(\gamma)$ entonces $[\top \to \bullet \mathbf{R}^\beta, \mathit{stars}] \in S_i$
   - si $M^\gamma \notin \mathrm{espina}(\gamma)$ entonces $[\top \to \bullet \mathbf{R}^\beta, \{\ \}] \in S_i$
2. FootPred: Si $[\mathbf{F}^\beta \to \bullet \bot, \mathit{stars}] \in S_i$ y $\beta \in \mathrm{adj}(M^\gamma)$ entonces $[M^\gamma \to \bullet \delta,\ \mathrm{apilar}(M^\gamma, \mathit{stars})] \in S_i$.
3. Pred: Si $[N^\gamma \to \delta \bullet M^\gamma \nu, \mathit{stars}] \in S_i$ y $\mathbf{nil} \in \mathrm{adj}(M^\gamma)$ entonces
   - si $M^\gamma \in \mathrm{espina}(\gamma)$ entonces $[M^\gamma \to \bullet \upsilon, \mathit{stars}] \in S_i$.
   - si $M^\gamma \notin \mathrm{espina}(\gamma)$ entonces $[M^\gamma \to \bullet \upsilon, \{\ \}] \in S_i$.
4. Comp: Si $[N^\gamma \to \delta \bullet M^\gamma \nu, \mathit{stars}'] \in S_i$ y $[M^\gamma \to \upsilon \bullet, \mathit{stars}] \in S_i$ y $M^\gamma \neq \mathbf{F}^\gamma$ entonces
   - si $M^\gamma \in \mathrm{espina}(\gamma)$ entonces $[N^\gamma \to \delta M^\gamma \bullet \nu, \mathit{stars}] \in S_i$
   - si $M^\gamma \notin \mathrm{espina}(\gamma)$ entonces $[N^\gamma \to \delta M^\gamma \bullet \nu, \mathit{stars}'] \in S_i$
5. SubsPred: Si $[N^\gamma \to \delta \bullet M^\gamma \nu, \mathit{stars}] \in S_i$ y el nodo M^γ está marcado para sustitución y etiqueta(M^γ) = etiqueta$(\mathbf{R}^\alpha)$, entonces $[\top \to \mathbf{R}^\alpha, \{\ \}] \in S_i$.

Para cada estado S_i se definen los tres tipos siguientes de transiciones:

1. *Desplazamiento*: si $[N^\gamma \to \delta \bullet M^\gamma \nu, \mathit{stars}] \in S_i$ y etiqueta$(M^\gamma) \in V_T$ o bien M^γ es un nodo de sustitución, entonces hay una transición etiquetada por etiqueta(M^γ) hacia el estado S_j que contiene el ítem $[N^\gamma \to \delta M^\gamma \bullet \nu, \mathit{stars}]$. Dicha transición se denota $S_j \in \mathrm{trans}(S_i, \mathrm{etiqueta}(M^\gamma))$.
2. *Adjunción*: si $[M^\gamma \to \delta \bullet, \mathit{stars}] \in S_i$ y $M^\gamma = \mathrm{cima}(\mathit{stars})$ y $\beta \in \mathrm{adj}(M^\gamma)$ entonces $S_j \in \mathrm{trans}(S_i, \beta_{derecha})$, tal que el estado S_j contiene el ítem $[N^\gamma \to \delta M^\gamma \bullet \nu, \mathit{stars}'']$ donde stars'' es el subconjunto de stars formado por los nodos que dominan N^γ .
3. *Pie*: si $[\mathbf{F}^\beta \to \bullet \bot, \mathit{stars}] \in S_i$ entonces $S_j \in \mathrm{trans}(S_i, \mathbf{F}^\beta)$, tal que el estado S_j contiene el ítem $[\mathbf{F}^\beta \to \bot \bullet, \mathit{stars}]$.

Con respecto a la tabla de análisis sintáctico, los cinco tipos de acciones almacenadas en ACCIÓN(S_i, a) son:

1. *Desplazar al estado S_j*: se aplica si y sólo si $j \in \mathrm{trans}(S_i, a)$.
2. *Restaurar derecha de $M^\gamma \to \upsilon$ con β*: se aplica si y sólo si $[N^\gamma \to \delta \bullet M^\gamma \nu, \mathit{stars}] \in S_i$, $M^\gamma = \mathrm{cima}(\mathit{stars})$ y $\beta \in \mathrm{adj}(M^\gamma)$.
3. *Reducir raíz de β con* star: se aplica si y sólo si $[\top \to \mathbf{R}^\beta \bullet, \{\mathit{star}\}] \in S_i$ y $\beta \in \boldsymbol{A}$
4. *Reducir Árbol Inicial α*: se aplica si y sólo si $[\top \to \mathbf{R}^\beta \bullet, \{\ \}] \in S_i$.
5. *Aceptar*: ocurre si y sólo si el siguiente elemento en la cadena de entrada es el marcador de fin de cadena y $[\top \to \mathbf{R}^\alpha \bullet, \{\ \}] \in S_i$ y $\alpha \in \boldsymbol{I}$.
6. *Error*: se aplica si y sólo si ninguna de las otras acciones es aplicable.

El programa conductor procede leyendo terminal a de la cadena de entrada que se está tratando y el estado S_i que está en la cima de la secuencia de pilas, a partir de los cuales determina al movimiento a realizar mediante la consulta de la entrada ACCIÓN(S_i, a) de la tabla de acciones:

1. ACCIÓN(S_i, a)=*Desplazar al estado* S_j. En este caso apila S_j.
2. ACCIÓN(S_i, a)=*Restaurar derecha de* $M^\gamma \rightarrow v$ *con* β. Se pueden dar dos casos:
   (a) Si M^γ no domina el nodo pie de γ, sea k el número de terminales más el número de nodos sustituibles dominados por M^γ. Se fusionan las k pilas de la cima, situando una nueva pila unitaria conteniendo $m \in \mathrm{IR\_A}_{foot}(S_{k'}, \beta)$ en la cima de la secuencia de pilas, donde $S_{k'}$ es el elemento de la pila unitaria situada inmediatamente debajo de todas las pilas fusionadas.
   (b) Si M^γ domina el nodo pie de γ, sean k_1 y k_2 el número de terminales más el número de nodos sustituibles a la derecha y a la izquierda del pie de γ, respectivamente. Debemos unir los elementos de la pila en posición $k_1 + 1$ con los elementos de las k_2 pilas unitarias bajo la pila k_1+2, situándola en la posición k_1+2. Crearemos una nueva pila unitaria en la cima de la secuencia de pilas conteniendo $m \in \mathrm{IR\_A}_{foot}(S_{k_1+1}, \beta)$.
3. ACCIÓN(S_i, a)=*Reducir Raíz de* β *con star*. Sean k_1 y k_2 el número de terminales más el número de nodos sustituibles a la derecha y a la izquierda del pie de β. Sea $A = \mathrm{etiqueta}(star)$ y p el número de símbolos terminales más el número de nodos de sustitución a la izquierda del nodo pie dominado por *star*. En caso de que $\{star\} = \emptyset$ tomaremos $p = 0$. Se extraerán de la pila los $p + k_2 + 1$ elementos superiores y a continuación se extraerán $k_1 - p$ pilas de tamaño 1 . Sea S_i el nuevo elemento en la cima de la pila. Dicho elemento de extraerá y en su lugar se insertará $S_j \in \mathrm{IR\_A}_{derecha}\beta$.
4. ACCIÓN(S_i, a)=*Reducir Árbol Inicial* α. Sea k el número de símbolos terminales más el número de nodos sustituibles dominados por la raíz de α. Entonces k elementos serán eliminados de la pila y se apilará $m \in \mathrm{trans}(S_{k+1}, \mathrm{etiqueta}(\mathbf{R}^\alpha))$.

El algoritmo procede a aplicar todas las acciones posibles hasta que se alcanza el reconocimiento de la cadena de entrada o se obtiene un error. Cuando en un momento dado son posibles varias acciones alternativas y el filtrado no es suficiente para eliminar el no determinismo, se deberán explorar todas las posibles alternativas utilizando técnicas basadas en retroceso o técnicas pseudo-paralelas basadas en programación dinámica [107, 199].

El algoritmo LR de Nederhof

Nederhof presenta en [123] un algoritmo de tipo LR para TAG que trata de solventar los inconvenientes del algoritmo de Kinyon. Entre las diferencias más destacadas, reseñamos las siguientes:

- Utiliza una autómata a pila en vez de un BEPDA como modelo operacional.
- No utiliza una tabla de acciones para almacenar las acciones a realizar durante el proceso. En su lugar, las acciones se determinan dinámicamente consultando el contenido de la pila.
- Es preciso realizar modificaciones para que admita árboles con nodos hoja etiquetados por ϵ.

La operación cerradura utilizada para la construcción del autómata finito que guía el proceso de análisis trabaja con ítems que contienen, además de una producción independiente del contexto con punto, el nodo correspondiente a la raíz del subárbol sobre el que se está trabajando[16]. Este elemento muestra su utilidad en las operaciones relacionadas con el reconocimiento

[16]Para representar que se está trabajando con el árbol γ al completo, se utilizará el nodo artificial $\top^\gamma$.

del pie de un árbol auxiliar involucrado en una operación de adjunción. Los ítems son por tanto de la forma $[M^\gamma, M^\gamma \rightarrow \delta \bullet \nu]$. Para realizar la operación cerradura es necesario aplicar los siguientes pasos hasta que no puedan crear más estados, partiendo de un estado con el ítem $[\top^\alpha, \top \rightarrow \bullet \mathbf{R}^\alpha]$:

1. AdjPred: Si $[N^\gamma, M^\gamma \rightarrow \delta \bullet P^\gamma \nu] \in S_i$ y $\beta \in \text{adj}(P^\gamma)$ entonces $[\top^\beta, \top \rightarrow \bullet \mathbf{R}^\beta] \in S_i$.

2. FootPred: Si $[N^\beta, \mathbf{F}^\beta \rightarrow \bullet \perp] \in S_i$ y $\beta \in \text{adj}(P^\gamma)$ entonces $[P^\gamma, P^\gamma \rightarrow \bullet \delta] \in S_i$.

3. Pred: Si $N^\gamma, M^\gamma \rightarrow \delta \bullet P^\gamma \nu] \in S_i$ y $\mathbf{nil} \in \text{adj}(P^\gamma)$ entonces $[N^\gamma, P^\gamma \rightarrow \bullet \upsilon] \in S_i$.

4. Comp: Si $N^\gamma, P^\gamma \rightarrow \upsilon \bullet] \in S_i$ entonces $[N^\gamma, M^\gamma \rightarrow \delta P^\gamma \bullet \nu] \in S_i$.

Con respecto a las transiciones entre estados, existen los tres tipos siguientes:

1. *Desplazamiento*: Si $[N^\gamma, M^\gamma \rightarrow \delta \bullet a\nu] \in S_i$ donde $a \in V_T$ entonces existe una transición etiquetada por a hasta el estado S_j que contiene el ítem $[N^\gamma, M^\gamma \rightarrow \delta a \bullet \nu]$. Dicha transición se denota por $\text{IR\_A}(S_i, a)$.

2. *Adjunción*: Si $[N^\gamma, M^\gamma \rightarrow \delta \bullet P^\gamma \nu] \in S_i$ tal que $\beta \in \text{adj}(P^\gamma)$ para algún β, entonces existe una transición etiquetada por P^γ hasta el estado S_j que contiene el ítem $[N^\gamma, M^\gamma \rightarrow \delta P^\gamma \bullet \nu]$. Dicha transición se denota, al igual que en el caso precedente, por $S_j \in \text{IR\_A}(S_i, P^\gamma)$.

3. *Pie*: Si $[N^\beta, \mathbf{F}^\beta \rightarrow \bullet \perp] \in S_i$ y $\beta \in \text{adj}(P^\gamma)$ entonces existe una transición etiquetada por P^γ hasta el estado S_j que contiene el ítem $[N^\beta, \mathbf{F}^\beta \rightarrow \perp \bullet]$. Dicha transición se denota $S_j \in \text{IR\_A}_\perp(S_i, P^\gamma)$.

Para determinar las acciones a realizar durante el proceso de análisis se calculan las dos tablas siguientes a partir de los estados del autómata finito:

1. *Reducciones*: La tabla de reducciones contiene una entrada para cada estado de modo que $reducciones(S_i) = \{\beta \in \boldsymbol{A} \mid [\top^\beta, \top \rightarrow \mathbf{R}^\beta \bullet] \in S_i\} \cup \{N^\gamma \mid [N^\gamma, N^\gamma \rightarrow \delta \bullet]\}$.

2. *Secciones cruzadas*: La tabla de secciones cruzadas tiene una entrada por cada nodo de cada árbol elemental, de tal modo que $CS(N^\gamma)$ contiene todas las posibles fronteras del subárbol enraizado por N^γ, suponiendo que se eliminan sucesivamente los subárboles que lo constituyen. Denotamos mediante $CS(N^\gamma)^+$ a $CS(N^\gamma) - \{N^\gamma\}$. Estas tablas realizan una labor esencial en el algoritmo de análisis, ya que permiten decidir si se puede realizar la reducción de un nodo N^γ. Para ello es necesario que los elementos de la cima de la pila se correspondan con una de las secuencias almacenadas en la entrada de $CS(N^\gamma)^+$. Una forma eficiente de implementar las tablas de secciones cruzadas consiste en utilizar un autómata finito para almacenar el contenido de cada entrada, en lugar de un conjunto de secuencias de nodos.

Como hemos comentado anteriormente, este algoritmo trabaja sobre un autómata a pila. La estructura de la pila es compleja, pues almacena tres tipos diferentes de elementos:

1. Estados S_i del autómata finito.

2. Nodos N^γ de árboles elementales.

3. Nodos de árboles elementales junto a una pila de nodos $M^\gamma[N \mid L]$, donde N representa el nodo en la cima de la pila y L el resto de la pila[17]. Cada una de estas pilas contiene una parte de la espina del árbol derivado en la que se van acumulando los nodos en los que se ha realizado una adjunción que aún no ha sido completada. Como el algoritmo procede de modo ascendente, las adjunciones son reconocidas en el pie y completadas en la raíz de un árbol auxiliar. Esta pilas se corresponden con las pilas asociadas a los símbolos gramaticales de una LIG que representa a una TAG. También constituyen una diferencia fundamental con el algoritmo de Kinyon, puesto que saca dichas pilas de los ítems del autómata finito para introducirlas en la pila que se manipula en tiempo de ejecución.

Dada una determinada configuración de la pila, la acción a realizar es una de las cinco siguientes:

1. *Desplazar al estado S_j*: se aplica cuando en la cima de la pila se encuentra el estado S_i, el siguiente elemento de la cadena de entrada es un terminal a y $S_j \in \text{IR\_A}(S_i, a)$.

2. *Reducir subárbol*: se aplica cuando en la cima de pila se encuentra la secuencia $S_0 X_1 S_1 X_2 S_2 \ldots X_m S_m$ tal que $X_1 \ldots X_m \in CS(N^\gamma)^+$ y $S' \in \text{IR\_A}_\perp(S_0, N^\gamma)$. Entonces dicha secuencia es reemplazada por $S_0 \perp[N^\gamma \mid L]\ S'$. Si N^γ domina un nodo pie entonces L se obtiene del elemento $X_i = M^\gamma[L]$, en caso contrario L es la pila vacía. Lo que se ha hecho es reconocer la sección cruzada de un subárbol de un árbol elemental γ enraizado por un nodo N^γ que admite adjunciones. Como resultado se ha sustituido dicha sección cruzada y los estados asociados por el pie de un posible árbol auxiliar adjuntado con el fin continuar el reconocimiento ascendente de este último. Esta acción se corresponde con la acción *Restaurar Derecha* del algoritmo de Kinyon.

3. *Reducir árbol auxiliar*: se aplica cuando en la cima de la pila se encuentra la secuencia $S_0 X_1 S_1 X_2 \ldots X_m q_m$ y $\beta \in reducciones(S_m)$, $X_1 \ldots X_m \in CS(\mathbf{R}^\beta)^+$ y $S' \in \text{IR\_A}(S_0, N^\gamma)$, donde N^γ se obtiene a partir del único $X_i = M[N \mid L]$. Entonces dicha secuencia es reemplazada por $S_0 X S'$, donde $X = N$ si L es la pila vacía y $X = N[L]$ en otro caso. Lo que se ha hecho es reconocer la sección cruzada de la raíz de un árbol auxiliar β, sustituyendo dicha sección y sus estados asociados por el nodo en el cual se realizó la adjunción de β. Esta acción se corresponde con la acción *Reducir Raíz de β* del algoritmo de Kinyon.

4. *Aceptar*: ocurre si y sólo si en la cima de pila se encuentra el estado final y el siguiente elemento en la cadena de entrada es el marcador de fin de cadena.

5. *Error*: se aplica si y sólo si ninguna de las otras acciones es aplicable.

Los nodos hoja etiquetados por ϵ no dejan ninguna traza en la pila, lo cual presenta problemas a la hora de comprobar si en la cima de pila se encuentra la sección cruzada de algún nodo. Para solventarlo es necesario refinar los ítems utilizados para construir el autómata finito introduciendo un tercer componente que indica los nodos con etiqueta ϵ que han sido visitados por la operación cerradura. Igualmente, es preciso extender el concepto de sección cruzada de modo que al calcularla se tenga en cuenta qué nodos tienen etiqueta ϵ.

Prolo presenta en [150] una variante del algoritmo LR de Nederhof que genera tablas de análisis de menor tamaño.

[17] Las secciones cruzadas para este tipo de elementos se calculan ignorando la pila de nodos.

Generación incremental de las tablas de análisis

La principal ventaja del análisis LR, tanto en el caso de gramáticas independientes del contexto como en el caso de gramáticas de adjunción de árboles es que la información predictiva presente en la gramática se compila en un conjunto de tablas que son utilizadas durante la fase de análisis de las cadenas de entrada. Esta ventaja se convierte en un inconveniente cuando se trabaja en entornos en los que la gramática se ve sometida a cambios frecuentes, puesto que la fase de generación de las tablas consume muchos recursos, tanto en tiempo como en espacio de memoria. Sarkar propone solventar este problema extendiendo las técnicas propuestas para el caso de analizadores LR de gramáticas independientes del contexto [157, 82, 83] a los analizadores LR para TAG. Su propuesta [163] consiste en combinar la generación perezosa de las tablas de análisis[18] con generación incremental de las tablas de análisis. Cuando se produce algún cambio en la gramática, se seleccionan aquellos estados del autómata finito que se ven afectados por dicha modificación, manteniendo únicamente los ítems núcleo[19]. El autómata será reconstruido de acuerdo con la nueva gramática mediante la aplicación de la generación perezosa durante el análisis de nuevas cadenas de entrada.

Aunque diseñada originalmente para el algoritmo de Schabes y Vijay-Shanker [176], esta técnica de generación incremental de analizadores LR para TAG se puede adaptar fácilmente al algoritmo propuesto por Kinyon.

3.9.6 Algoritmos paralelos

Los algoritmos mostrados hasta el momento son algoritmos secuenciales, pensados para ser programados y ejecutados en ordenadores secuenciales. Su límite de complejidad temporal es $\mathcal{O}(n^p)$. Mediante la paralelización de los algoritmos tabulares se pueden conseguir límites inferiores a costa de aumentar las unidades de procesamiento paralelo necesarias para implementar el algoritmo. El principio en que se basan los algoritmos paralelos es el de explotar la redundancia presente en los algoritmos tabulares para el análisis de TAG.

Se ha propuesto varios algoritmos paralelos para el análisis sintáctico de TAG. Palis, Shende y Wei proponen en [139] una versión paralelizada del algoritmo tabular de tipo CYK [209] para el análisis de gramáticas de adjunción que presenta una complejidad lineal y que precisa una malla 5-dimensional de n^5 procesadores para ejecutarse (donde n es la longitud de la cadena de entrada), con la ventaja adicional de que cada procesador realiza una función que no es dependiente de la longitud de la cadena de entrada sino únicamente de la gramática que se está analizando. Este algoritmo, al estar derivado del algoritmo CYK, sólo trabaja con gramáticas de adjunción de árboles en forma binaria, en la que los nodos de los árboles elementales no tienen más de dos hijos. Posteriormente Palis y Wei han propuesto un nuevo algoritmo para gramáticas de adjunción de árboles en forma general [141]. El comportamiento de dicho algoritmo depende fuertemente del tamaño de la gramática, aunque siempre obtiene complejidades inferiores a las versiones secuenciales. Las versiones SIMD de sus algoritmos [140, 141], implementadas en una *Connection Machine CM-2* ofrecen una aceleración con respecto a las versiones secuenciales que muestra un comportamiento asintótico con respecto al tamaño de la gramática.

Nurkkala y Kumar [136] proponen una variación del algoritmo de Palis y Wei [141] para máquinas MIMD conectadas en hipercubo, implementada en una máquina nCUBE/2. Su algoritmo proporciona mayor granularidad al asignar un conjunto dado de árboles elementales

[18]Las tablas de análisis se van generando a medida que son requeridas durante el análisis de una cadena de entrada. Esta técnica es útil cuando la gramática es muy grande y sólo una pequeña parte de la misma será utilizada para analizar una cadena de entrada. Su inconveniente es el elevado consumo de memoria, puesto que el autómata finito utilizado para generar las tablas debe estar también presente durante la fase de análisis.

[19]Aquellos que provocan la aplicación de las operaciones de cerradura de un estado.

a cada procesador. Ciertos resultados parciales del análisis, concretamente el reconocimiento de la raíz de un árbol elemental, se distribuyen asíncronamente a través del multicomputador. Concurrentemente con la ejecución del proceso de análisis, se aplica un algoritmo de detección de terminación distribuido asíncrono. Los resultados experimentales muestran una mejora en la aceleración con respecto al algoritmo de Palis y Wei. Los mismo autores presentan en [137] un algoritmo de tipo Earley para TAG, paralelizado para un multicomputador *KSR1* de memoria compartida con tiempo de acceso no uniforme.

Rajasekaran propone en [151] un algoritmo de tipo Earley ascendente para gramáticas de adjunción de árboles en forma binaria que presenta una complejidad temporal $\mathcal{O}(n \log n)$ cuando se implementa en una máquina paralela *EREW PRAM* con $\frac{n^2 M(n)}{\log n}$ procesadores, donde $M(k)$ es el tiempo necesario para multiplicar dos matrices booleanas de tamaño $k \times k$.

Capítulo 4

Algoritmos de análisis sintáctico para LIG

Describiremos varios algoritmos para el análisis de gramáticas lineales de índices. Además del interés que presenta por sí mismo el análisis sintáctico de este tipo de gramáticas, su estudio presenta como atractivo adicional el que muchos autores prefieran realizar el análisis sintáctico de las gramáticas de adjunción de árboles indirectamente a través de LIG, esto es, traduciendo la gramática original a una gramática lineal de índices y aplicando un algoritmo de análisis sintáctico a esta última.

La mayor parte de los algoritmos de análisis sintáctico de LIG están basados en algoritmos de análisis independientes del contexto extendidos para permitir el tratamiento de las pilas de índices asociadas a los no-terminales de la gramática. Asimismo, es de destacar que prácticamente todos estos algoritmos utilizan técnicas de programación dinámica, obteniendo una complejidad temporal polinómica de orden $\mathcal{O}(n^6)$, donde n es la longitud de la cadena de entrada.

La aportación de este capítulo es múltiple, destacando la definición de nuevos algoritmos para el análisis de LIG, como es el caso del algoritmo de tipo Earley ascendente, del algoritmo de tipo Earley sin la propiedad del prefijo válido y de los algoritmos de tipo Earley que preservan dicha propiedad. Mediante estos nuevos algoritmos podemos aplicar al caso de LIG las mismas estrategias de análisis descritas para el caso de TAG en el capítulo 3. Ello permite mantener la estrategia de análisis cuando se trata de analizar una TAG previamente compilada a LIG. Además de presentar estos nuevos algoritmos, se proporciona un camino evolutivo que los relaciona y permite derivar uno de otro y se presenta una nueva definición de bosque compartido de análisis sintáctico de LIG. Por último, se realiza una descripción conjunta de la mayor parte de los algoritmos de análisis existentes para LIG. El contenido de este capítulo está basado en [15, 19].

4.1 Algoritmo de tipo CYK

A continuación mostramos una extensión para LIG del algoritmo CYK de análisis sintáctico basado en el algoritmo descrito por Vijay-Shanker y Weir en [213], al que incorpora algunas correcciones. Asumiremos que cada producción posee dos elementos en el lado derecho o bien un único elemento que debe ser un terminal. Este condicionante puede verse como una trasposición de la forma normal de Chomsky [85] al caso de las gramáticas lineales de índices.

El algoritmo trabaja reconociendo de forma ascendente la parte de la cadena de entrada cubierta por cada posible elemento gramatical. Para ello se utiliza un conjunto de ítems de la forma

$$[A, \gamma, i, j \mid B, p, q]$$

que representan uno de los siguientes tipos de derivaciones:

- $A[\gamma] \overset{*}{\Rightarrow} a_{i+1} \dots a_p \; B[\,] \; a_{q+1} \dots a_j$ si y sólo si $(B, p, q) \neq (-, -, -)$ y donde $B[\,]$ es un descendiente dependiente de $A[\gamma]$.

- $A[\,] \overset{*}{\Rightarrow} a_{i+1} \dots a_j$ si y sólo si $\gamma = -$ y $(B, p, q) = (-, -, -)$.

Estos ítems son ligeramente diferentes de los propuestos por Vijay-Shanker y Weir en [213], pues estos últimos presentaban la forma $[A, \gamma, i, j \mid B, \eta, p, q]$, con $\eta \in V_I$. El elemento η es redundante ya que por la propiedad de independencia del contexto de las gramáticas lineales de índices (definición 2.1, página 32) sabemos que si $A[\gamma] \overset{*}{\Rightarrow} a_{i+1} \dots a_p \; B[\,] \; a_{q+1} \dots a_j$ entonces para cualquier α se cumple que $A[\alpha\gamma] \overset{*}{\Rightarrow} a_{i+1} \dots a_p \; B[\alpha] \; a_{q+1} \dots a_j$. La descomposición de α que realizan Vijay-Shanker y Weir al considerar $\alpha = \alpha'\eta$ es innecesaria y el almacenamiento de η lo único que provoca es un aumento en el número de ítems necesarios para representar una derivación, puesto que deberemos utilizar un ítem $[A, \gamma, i, j \mid B, \eta, p, q]$ diferente para cada valor de η, cuando con un solo ítem $[A, \gamma, i, j \mid B, p, q]$ es suficiente.

Esquema de análisis sintáctico 4.1 El sistema de análisis $\mathbb{P}_{\text{CYK}}$ que se corresponde con el algoritmo de análisis de tipo CYK para una gramática lineal de índices $\mathcal{L}$ y una cadena de entrada $a_1 \dots a_n$ se define como sigue:

$$\mathcal{I}_{\text{CYK}} = \left\{ \; [A, \gamma, i, j \mid B, p, q] \mid A, B \in V_N, \;\; \gamma \in V_I, \;\; 0 \leq i \leq j \;, \quad (p, q) \leq (i, j) \; \right\}$$

$$\mathcal{H}_{\text{CYK}} = \left\{ \; [a, i-1, i] \mid a = a_i, \;\; 1 \leq i \leq n \; \right\}$$

$$\mathcal{D}_{\text{CYK}}^{\text{Scan}} = \frac{[a, j, j+1]}{[A, -, j, j+1 \mid -, -, -]} \quad A[\,] \to a \in P$$

$$\mathcal{D}_{\text{CYK}}^{[\circ\circ\gamma][\,][\circ\circ]} = \frac{\begin{array}{c}[B, -, i, k \mid -, -, -],\\ [C, \eta, k, j \mid D, p, q]\end{array}}{[A, \gamma, i, j \mid C, k, j]} \quad A[\circ\circ\gamma] \to B[\,] \; C[\circ\circ] \in P$$

$$\mathcal{D}_{\text{CYK}}^{[\circ\circ\gamma][\circ\circ][\,]} = \frac{\begin{array}{c}[B, \eta, i, k \mid D, p, q],\\ [C, -, k, j \mid -, -, -]\end{array}}{[A, \gamma, i, j \mid B, i, k]} \quad A[\circ\circ\gamma] \to B[\circ\circ] \; C[\,] \in P$$

$$\mathcal{D}_{\text{CYK}}^{[\circ\circ][\,][\circ\circ]} = \frac{\begin{array}{c}[B, -, i, k \mid -, -, -],\\ [C, \eta, k, j \mid D, p, q]\end{array}}{[A, \eta, i, j \mid D, p, q]} \quad A[\circ\circ] \to B[\,] \; C[\circ\circ] \in P$$

$$\mathcal{D}_{\text{CYK}}^{[\circ\circ][\circ\circ][\,]} = \frac{\begin{array}{c}[B, \eta, i, k \mid D, p, q],\\ [C, -, k, j \mid -, -, -]\end{array}}{[A, \eta, i, j \mid D, p, q]} \quad A[\circ\circ] \to B[\circ\circ] \; C[\,] \in P$$

$$\mathcal{D}_{\text{CYK}}^{[\circ\circ][\,][\circ\circ\gamma]} = \frac{\begin{array}{c}[B, -, i, k \mid -, -, -],\\ [C, \gamma, k, j \mid D, p, q],\\ [D, \eta, p, q \mid E, r, s]\end{array}}{[A, \eta, i, j \mid E, r, s]} \quad A[\circ\circ] \to B[\,] \; C[\circ\circ\gamma] \in P$$

$$\mathcal{D}_{\text{CYK}}^{[\circ\circ][\circ\circ\gamma][\,]} = \frac{\begin{array}{l}[B,\gamma,i,k \mid D,p,q],\\ [C,-,k,j \mid -,-,-],\\ [D,\eta,p,q \mid E,r,s]\end{array}}{[A,\eta,i,j \mid E,r,s]} \quad A[\circ\circ] \to B[\circ\circ\gamma]\; C[\,] \in P$$

$$\mathcal{D}_{\text{CYK}} = \mathcal{D}_{\text{CYK}}^{\text{Scan}} \cup \mathcal{D}_{\text{CYK}}^{[\circ\circ\gamma][\,][\circ\circ]} \cup \mathcal{D}_{\text{CYK}}^{[\circ\circ\gamma][\circ\circ][\,]} \cup \mathcal{D}_{\text{CYK}}^{[\circ\circ][\,][\circ\circ]} \cup \mathcal{D}_{\text{CYK}}^{[\circ\circ][\circ\circ][\,]} \cup \mathcal{D}_{\text{CYK}}^{[\circ\circ][\,][\circ\circ\gamma]} \cup \mathcal{D}_{\text{CYK}}^{[\circ\circ][\circ\circ\gamma][\,]}$$

$$\mathcal{F}_{\text{CYK}} = \{ \ [S,-,0,n \mid -,-,-] \ \}$$

§

La definición de las hipótesis realizada en este sistema de análisis sintáctico se corresponde con la estándar y es la misma que se utilizará en los restantes sistemas de análisis del capítulo. Por consiguiente, no nos volveremos a referir explícitamente a ellas.

Los pasos $\mathcal{D}_{\text{CYK}}^{\text{Scan}}$ son los encargados de iniciar el procesamiento ascendente de la cadena de entrada. Los demás pasos se encargan de combinar los ítems correspondientes a los elementos del lado derecho de una producción para generar el ítem correspondiente al lado izquierdo de dicha producción.

La complejidad espacial del algoritmo con respecto a la longitud n de la cadena de entrada es $\mathcal{O}(n^4)$ puesto que cada ítem almacena cuatro posiciones de la cadena de entrada. La complejidad temporal con respecto a la cadena de entrada es $\mathcal{O}(n^6)$ y viene dada por los pasos deductivos $\mathcal{D}_{\text{CYK}}^{[\circ\circ][\,][\circ\circ\gamma]}$ y $\mathcal{D}_{\text{CYK}}^{[\circ\circ][\circ\circ\gamma][\,]}$. Aunque dichos pasos manipulan en principio 7 posiciones de la cadena de entrada, mediante aplicación parcial cada uno de ellos se puede descomponer en una sucesión de pasos que manipulan a lo sumo 6 posiciones de la cadena de entrada.

Vijay-Shanker y Weir describen en [214] un algoritmo de tipo CYK generalizado para gramáticas lineales de índices que manipulan más de un índice de la pila de índices en cada producción. La misma generalización puede ser aplicada al esquema de análisis sintáctico propuesto. Schabes extiende en [170] el algoritmo CYK para permitir el cálculo de probabilidades requerido para el tratamiento de LIG estocásticas.

4.2 Algoritmo de tipo Earley ascendente

El algoritmo de tipo CYK presenta una limitación muy importante: sólo es aplicable a gramáticas lineales de índices cuyas producciones tienen a lo sumo dos elementos en el lado derecho. Para evitar esta limitación vamos a considerar la extensión del algoritmo Earley ascendente al caso de las gramáticas lineales de índices. Debemos reseñar que no conocemos ninguna adaptación anterior de este algoritmo para LIG.

Como primer paso para la definición de un algoritmo de tipo Earley ascendente para LIG debemos proceder a la introducción de un punto en las producciones, que nos permitirá distinguir la parte de la producción ya reconocida de aquella que resta por reconocer. Con respecto a la notación utilizada, utilizaremos $\boldsymbol{A}$ para referirnos al elemento LIG de una producción constituido por el no-terminal A y una pila de índices asociada, cuando la forma de dicha pila sea irrelevante en el contexto de utilización. En consecuencia, la aparición de diferentes $\boldsymbol{A}$ en un ítem o paso deductivo indica que en todos los casos de trata de elementos LIG con el mismo no-terminal A pero que puede que tengan asociadas distintas pilas de índices.

Los ítems utilizados en el algoritmo de análisis sintáctico de tipo Earley ascendente para LIG tienen la forma

$$[\boldsymbol{A} \to \Upsilon_1 \bullet \Upsilon_2, \gamma, i, j \mid B, p, q]$$

y representan alguno de los siguientes tipos de derivaciones:

- $A[\gamma] \Rightarrow \Upsilon_1 \Upsilon_2 \stackrel{*}{\Rightarrow} a_{i+1} \dots a_p \; B[\,] \; a_{q+1} \dots a_j \; \Upsilon_2$ si y sólo si $(B,p,q) \neq (-,-,-)$, donde $B[\,]$ es un descendiente dependiente de $A[\gamma]$.

- $\Upsilon_1 \stackrel{*}{\Rightarrow} a_{i+1} \dots a_j$ si y sólo si $\gamma = -$ y $(B,p,q) = (-,-,-)$. Si Υ_1 incluye al hijo dependiente entones las pilas asociadas a $\boldsymbol{A}$ y al hijo dependiente están vacías.

Podemos observar que los ítems del nuevo esquema de análisis sintáctico, que denominaremos **buE**, son un refinamiento de los ítems de **CYK**. Sobre los pasos deductivos aplicaremos también un refinamiento puesto que los pasos de tipo $\mathcal{D}_{\text{CYK}}^{[\circ\circ\gamma][\,][\circ\circ]}$, $\mathcal{D}_{\text{CYK}}^{[\circ\circ\gamma][\circ\circ][\,]}$, $\mathcal{D}_{\text{CYK}}^{[\circ\circ][\,][\circ\circ]}$, $\mathcal{D}_{\text{CYK}}^{[\circ\circ][\circ\circ][\,]}$, $\mathcal{D}_{\text{CYK}}^{[\circ\circ][\,][\circ\circ\gamma]}$ y $\mathcal{D}_{\text{CYK}}^{[\circ\circ][\circ\circ\gamma][\,]}$ serán separados en diferentes tipos de pasos Init y Comp. Finalmente se realizará una *extensión* del dominio de las producciones para permitir gramáticas lineales de índices con producciones de longitud arbitraria.

Esquema de análisis sintáctico 4.2 El sistema de análisis $\mathbb{P}_{\text{buE}}$ que se corresponde con el algoritmo de análisis de tipo Earley ascendente para una gramática lineal de índices $\mathcal{L}$ y una cadena de entrada $a_1 \dots a_n$ se define como sigue:

$$\mathcal{I}_{\text{buE}} = \left\{ \begin{array}{ll} [\boldsymbol{A} \rightarrow \Upsilon_1 \bullet \Upsilon_2, \gamma, i, j \mid B, p, q] \mid & \boldsymbol{A} \rightarrow \Upsilon_1 \Upsilon_2 \in P, \; B \in V_N, \; \gamma \in V_I, \\ & 0 \leq i \leq j \;, \quad (p,q) \leq (i,j) \end{array} \right\}$$

$$\mathcal{D}_{\text{buE}}^{\text{Init}} = \frac{}{[\boldsymbol{A} \rightarrow \bullet \Upsilon, -, i, i \mid -,-,-]}$$

$$\mathcal{D}_{\text{buE}}^{\text{Scan}} = \frac{\begin{array}{l} [A[\,] \rightarrow \bullet a, -, j, j \mid -,-,-], \\ [a, j, j+1] \end{array}}{[A[\,] \rightarrow a \bullet, -, j, j+1 \mid -,-,-]}$$

$$\mathcal{D}_{\text{buE}}^{\text{Comp}[\,]} = \frac{\begin{array}{l} [\boldsymbol{A} \rightarrow \Upsilon_1 \bullet B[\,] \; \Upsilon_2, \gamma, i, k \mid C, p, q], \\ [\boldsymbol{B} \rightarrow \Upsilon_3 \bullet, -, k, j \mid -,-,-] \end{array}}{[\boldsymbol{A} \rightarrow \Upsilon_1 \; B[\,] \bullet \Upsilon_2, \gamma, i, j \mid C, p, q]}$$

$$\mathcal{D}_{\text{buE}}^{\text{Comp}[\circ\circ\gamma][\circ\circ]} = \frac{\begin{array}{l} [A[\circ\circ\gamma] \rightarrow \Upsilon_1 \bullet B[\circ\circ] \; \Upsilon_2, -, i, k \mid -,-,-], \\ [\boldsymbol{B} \rightarrow \Upsilon_3 \bullet, \eta, k, j \mid C, p, q] \end{array}}{[A[\circ\circ\gamma] \rightarrow \Upsilon_1 \; B[\circ\circ] \bullet \Upsilon_2, \gamma, i, j \mid B, k, j]}$$

$$\mathcal{D}_{\text{buE}}^{\text{Comp}[\circ\circ][\circ\circ]} = \frac{\begin{array}{l} [A[\circ\circ] \rightarrow \Upsilon_1 \bullet B[\circ\circ] \; \Upsilon_2, -, i, k \mid -,-,-], \\ [\boldsymbol{B} \rightarrow \Upsilon_3 \bullet, \eta, k, j \mid C, p, q] \end{array}}{[A[\circ\circ] \rightarrow \Upsilon_1 \; B[\circ\circ] \bullet \Upsilon_2, \eta, i, j \mid C, p, q]}$$

$$\mathcal{D}_{\text{buE}}^{\text{Comp}[\circ\circ][\circ\circ\gamma]} = \frac{\begin{array}{l} [A[\circ\circ] \rightarrow \Upsilon_1 \bullet B[\circ\circ\gamma] \; \Upsilon_2 \;, -, i, k \mid -,-,-], \\ [\boldsymbol{B} \rightarrow \Upsilon_3 \bullet \;, \gamma, k, j \mid C, p, q], \\ [\boldsymbol{C} \rightarrow \Upsilon_4 \bullet, \eta, p, q \mid D, r, s] \end{array}}{[A[\circ\circ] \rightarrow \Upsilon_1 \; B[\circ\circ\gamma] \bullet \Upsilon_2, \eta, i, j \mid D, r, s]}$$

$$\mathcal{D}_{\text{buE}} = \mathcal{D}_{\text{buE}}^{\text{Init}} \cup \mathcal{D}_{\text{buE}}^{\text{Scan}} \cup \mathcal{D}_{\text{buE}}^{\text{Comp}[\,]} \cup \mathcal{D}_{\text{buE}}^{\text{Comp}[\circ\circ\gamma][\circ\circ]} \cup \mathcal{D}_{\text{buE}}^{\text{Comp}[\circ\circ][\circ\circ]} \cup \mathcal{D}_{\text{buE}}^{\text{Comp}[\circ\circ][\circ\circ\gamma]}$$

$$\mathcal{F}_{\text{buE}} = \left\{ \; [\boldsymbol{S} \rightarrow \Upsilon \bullet \;, -, 0, n \mid -,-,-] \; \right\}$$

§

Proposición 4.1 **CYK** $\stackrel{\text{ir}}{\Longrightarrow}$ **CYK′** $\stackrel{\text{sr}}{\Longrightarrow}$ **ECYK** $\stackrel{\text{ext}}{\Longrightarrow}$ **buE**.

Demostración:

Como primer paso definiremos el sistema de análisis $\mathbb{P}_{\text{CYK}'}$ para una gramática lineal de índices $\mathcal{L}$ y una cadena de entrada $a_1 \dots a_n$.

$$\mathcal{I}_{\text{CYK}'} = \left\{ \begin{array}{ll} [\boldsymbol{A} \to \Upsilon_1 \bullet \Upsilon_2, \gamma, i, j \mid B, p, q] \mid & \boldsymbol{A} \to \Upsilon_1 \Upsilon_2 \in P,\ B \in V_N,\ \gamma \in V_I, \\ & 0 \leq i \leq j\ ,\ \ (p,q) \leq (k,j) \end{array} \right\}$$

$$\mathcal{D}^{\text{Scan}}_{\text{CYK}'} = \frac{[a, j, j+1]}{[A[\,] \to a\bullet, -, j, j+1 \mid -, -, -]}$$

$$\mathcal{D}^{[\circ\circ\gamma][\,][\circ\circ]}_{\text{CYK}'} = \frac{\begin{array}{c}[\boldsymbol{B} \to \Upsilon_1\bullet, -, i, k \mid -, -, -],\\ [\boldsymbol{C} \to \Upsilon_2\bullet, \eta, k, j \mid D, p, q]\end{array}}{[A[\circ\circ\gamma] \to B[\,]\ C[\circ\circ]\bullet, \gamma, i, j \mid C, k, j]}$$

$$\mathcal{D}^{[\circ\circ\gamma][\circ\circ][\,]}_{\text{CYK}'} = \frac{\begin{array}{c}[\boldsymbol{B} \to \Upsilon_1\bullet, \eta, i, k \mid D, p, q],\\ [\boldsymbol{C} \to \Upsilon_2\bullet, -, k, j \mid -, -, -]\end{array}}{[A[\circ\circ\gamma] \to B[\circ\circ]\ C[\,]\bullet, \gamma, i, j \mid B, i, k]}$$

$$\mathcal{D}^{[\circ\circ][\,][\circ\circ]}_{\text{CYK}'} = \frac{\begin{array}{c}[\boldsymbol{B} \to \Upsilon_1\bullet, -, i, k \mid -, -, -],\\ [\boldsymbol{C} \to \Upsilon_2\bullet, \eta, k, j \mid D, p, q]\end{array}}{[A[\circ\circ] \to B[\,]\ C[\circ\circ]\bullet, \eta, i, j \mid D, p, q]}$$

$$\mathcal{D}^{[\circ\circ][\circ\circ][\,]}_{\text{CYK}'} = \frac{\begin{array}{c}[\boldsymbol{B} \to \Upsilon_1\bullet, \eta, i, k \mid D, p, q],\\ [\boldsymbol{C} \to \Upsilon_2\bullet, -, k, j \mid -, -, -]\end{array}}{[A[\circ\circ] \to B[\circ\circ]\ C[\,]\bullet, \eta, i, j \mid D, p, q]}$$

$$\mathcal{D}^{[\circ\circ][\,][\circ\circ\gamma]}_{\text{CYK}'} = \frac{\begin{array}{c}[\boldsymbol{B} \to \Upsilon_1\bullet, -, i, k \mid -, -, -],\\ [\boldsymbol{C} \to \Upsilon_2\bullet, \gamma, k, j \mid D, p, q],\\ [\boldsymbol{D} \to \Upsilon_3\bullet, \eta, p, q \mid E, r, s]\end{array}}{[A[\circ\circ] \to B[\,]\ C[\circ\circ\gamma]\bullet, \eta, i, j \mid E, r, s]}$$

$$\mathcal{D}^{[\circ\circ][\circ\circ\gamma][\,]}_{\text{CYK}'} = \frac{\begin{array}{c}[\boldsymbol{B} \to \Upsilon_1\bullet, \gamma, i, k \mid D, p, q],\\ [\boldsymbol{C} \to \Upsilon_2\bullet, -, k, j \mid -, -, -],\\ [\boldsymbol{D} \to \Upsilon_3\bullet, \eta, p, q \mid E, r, s]\end{array}}{[A[\circ\circ] \to B[\circ\circ\gamma]\ C[\,]\bullet, \eta, i, j \mid E, r, s]}$$

$$\mathcal{D}_{\text{CYK}'} = \mathcal{D}^{\text{Scan}}_{\text{CYK}'} \cup \mathcal{D}^{[\circ\circ\gamma][\,][\circ\circ]}_{\text{CYK}'} \cup \mathcal{D}^{[\circ\circ\gamma][\circ\circ][\,]}_{\text{CYK}'} \cup \mathcal{D}^{[\circ\circ][\,][\circ\circ]}_{\text{CYK}'} \cup \mathcal{D}^{[\circ\circ][\circ\circ][\,]}_{\text{CYK}'} \cup \mathcal{D}^{[\circ\circ][\,][\circ\circ\gamma]}_{\text{CYK}'} \cup \mathcal{D}^{[\circ\circ][\circ\circ\gamma][\,]}_{\text{CYK}'}$$

$$\mathcal{F}_{\text{CYK}'} = \{\ [\boldsymbol{S} \to \Upsilon\bullet, -, 0, n \mid -, -, -]\ \}$$

Para demostrar que **CYK** $\stackrel{\text{ir}}{\Longrightarrow}$ **CYK′**, definiremos la siguiente función

$$f([\boldsymbol{A} \to \Upsilon\bullet, \gamma, i, j \mid C, p, q]) = [A, \gamma, i, j \mid C, p, q]$$

de la cual se obtiene directamente que $\mathcal{I}_{\text{CYK}} = f(\mathcal{I}_{\text{CYK}'})$ y que $\Delta_{\text{CYK}} = f(\Delta_{\text{CYK}'})$ por inducción en la longitud de las secuencias de derivación. En consecuencia, $\mathbb{P}_{\text{CYK}} \stackrel{\text{ir}}{\longrightarrow} \mathbb{P}_{\text{CYK}'}$, con lo que hemos probado lo que pretendíamos.

Definiremos ahora el sistema de análisis sintáctico $\mathbb{P}_{\text{ECYK}}$ para una gramática lineal de índices $\mathcal{L}$ cuyas producciones tienen a lo sumo dos elementos en el lado derecho y una cadena de entrada $a_1 \dots a_n$:

$$\mathcal{I}_{\text{ECYK}} = \mathcal{I}_{\text{CYK}'} = \mathcal{I}_{\text{buE}} =$$
$$\mathcal{D}_{\text{ECYK}}^{\text{Init}} = \mathcal{D}_{\text{buE}}^{\text{Init}}$$
$$\mathcal{D}_{\text{ECYK}}^{\text{Scan}} = \mathcal{D}_{\text{buE}}^{\text{Scan}}$$
$$\mathcal{D}_{\text{ECYK}}^{\text{Comp}[\,]} = \mathcal{D}_{\text{buE}}^{\text{Comp}[\,]}$$
$$\mathcal{D}_{\text{ECYK}}^{\text{Comp}[\circ\circ\gamma][\circ\circ]} = \mathcal{D}_{\text{buE}}^{\text{Comp}[\circ\circ\gamma][\circ\circ]}$$
$$\mathcal{D}_{\text{ECYK}}^{\text{Comp}[\circ\circ][\circ\circ]} = \mathcal{D}_{\text{buE}}^{\text{Comp}[\circ\circ][\circ\circ]}$$
$$\mathcal{D}_{\text{ECYK}}^{\text{Comp}[\circ\circ][\circ\circ\gamma]} = \mathcal{D}_{\text{buE}}^{\text{Comp}[\circ\circ][\circ\circ\gamma]}$$
$$\mathcal{F}_{\text{ECYK}} = \mathcal{F}_{\text{buE}}$$

Para demostrar que $\mathbf{CYK}' \stackrel{\mathbf{sr}}{\Longrightarrow} \mathbf{ECYK}$, deberemos demostrar que para todo sistema de análisis $\mathbb{P}_{\text{CYK}'}$ y $\mathbb{P}_{\text{ECYK}}$ se cumple que $\mathcal{I}_{\text{CYK}'} \subseteq \mathcal{I}_{\text{ECYK}}$ y $\stackrel{*}{\vdash}_{\text{CYK}'} \subseteq \stackrel{*}{\vdash}_{\text{ECYK}}$. Lo primero es cierto por definición, puesto que $\mathcal{I}_{\text{CYK}'} = \mathcal{I}_{\text{ECYK}}$. Para lo segundo debemos mostrar que $\mathcal{D}_{\text{CYK}'} \supseteq \stackrel{*}{\vdash}_{\text{ECYK}}$. Consideremos caso por caso:

- Un paso deductivo $\mathcal{D}_{\text{CYK}'}^{\text{Scan}}$ es equivalente a la secuencia de pasos deductivos constituida por la aplicación de un paso $\mathcal{D}_{\text{ECYK}}^{\text{Init}}$ y un paso $\mathcal{D}_{\text{ECYK}}^{\text{Scan}}$:

$$\frac{}{[A[\,] \to \bullet a, -, j, j \mid -, -, -]}$$

$$\frac{[A[\,] \to \bullet a, -, j, j \mid -, -, -],\ [a, j, j+1]}{[A[\,] \to a \bullet, -, j, j+1 \mid -, -, -]}$$

- Un paso deductivo $\mathcal{D}_{\text{CYK}'}^{[\circ\circ\gamma][\,][\circ\circ]}$ es equivalente a la secuencia de pasos deductivos constituida por la aplicación de un paso $\mathcal{D}_{\text{ECYK}}^{\text{Init}}$, un paso $\mathcal{D}_{\text{ECYK}}^{\text{Comp}[\,]}$ y un paso $\mathcal{D}_{\text{ECYK}}^{[\circ\circ\gamma][\circ\circ]}$:

$$\frac{}{[A[\circ\circ\gamma] \to \bullet B[\,]\ C[\circ\circ], -, i, i \mid -, -, -]}$$

$$\frac{[A[\circ\circ\gamma] \to \bullet B[\,]\ C[\circ\circ], -, i, i \mid -, -, -],\ [\boldsymbol{B} \to \Upsilon_1 \bullet, -, i, k \mid -, -, -]}{[A[\circ\circ\gamma] \to B[\,] \bullet C[\circ\circ], -, i, k \mid -, -, -]}$$

$$\frac{[A[\circ\circ\gamma] \to B[\,] \bullet C[\circ\circ], -, i, k \mid -, -, -],\ [\boldsymbol{C} \to \Upsilon_2 \bullet, \eta, k, j \mid D, p, q]}{[A[\circ\circ\gamma] \to B[\,]\ C[\circ\circ] \bullet, \gamma, i, j \mid C, k, j]}$$

- Un paso deductivo $\mathcal{D}_{\text{CYK}'}^{[\circ\circ\gamma][\circ\circ][\,]}$ es equivalente a la secuencia formada por un paso $\mathcal{D}_{\text{ECYK}}^{\text{Init}}$, un paso $\mathcal{D}_{\text{ECYK}}^{\text{Comp}[\circ\circ\gamma][\circ\circ]}$ y un paso $\mathcal{D}_{\text{ECYK}}^{[\,]}$:

$$\frac{}{[A[\circ\circ\gamma] \to \bullet B[\circ\circ]\ C[\,], -, i, i \mid -, -, -]}$$

$$\frac{[A[\circ\circ\gamma] \to \bullet B[\circ\circ]\ C[\,], -, i, i \mid -, -, -],\ [\boldsymbol{B} \to \Upsilon_1 \bullet, \eta, i, k \mid D, p, q]}{[A[\circ\circ\gamma] \to B[\circ\circ] \bullet C[\,], \gamma, i, k \mid B, i, k]}$$

$$\frac{[A[\circ\circ\gamma] \to B[\circ\circ] \bullet C[\,], -, i, k \mid B, i, k],\ [\boldsymbol{C} \to \Upsilon_2 \bullet, -, k, j \mid -, -, -]}{[A[\circ\circ\gamma] \to B[\circ\circ]\ C[\,] \bullet, \gamma, i, j \mid B, i, k]}$$

- Un paso $\mathcal{D}_{\mathrm{CYK}'}^{[\circ\circ][\,][\circ\circ]}$ es equivalente a una secuencia de pasos formada por un paso $\mathcal{D}_{\mathrm{ECYK}}^{\mathrm{Init}}$, un paso $\mathcal{D}_{\mathrm{ECYK}}^{\mathrm{Comp}[\,]}$ y un paso $\mathcal{D}_{\mathrm{ECYK}}^{\mathrm{Comp}[\circ\circ][\circ\circ]}$:

$$\frac{}{[A[\circ\circ] \to \bullet B[\,]\ C[\circ\circ], -, i, i \mid -, -, -]}$$

$$\frac{[A[\circ\circ] \to \bullet B[\,]\ C[\circ\circ], -, i, i \mid -, -, -],\ [\boldsymbol{B} \to \Upsilon_1 \bullet, -, i, k \mid -, -, -]}{[A[\circ\circ] \to B[\,] \bullet C[\circ\circ], -, i, k \mid -, -, -]}$$

$$\frac{[A[\circ\circ] \to B[\,] \bullet C[\circ\circ], -, i, k \mid -, -, -],\ [\boldsymbol{C} \to \Upsilon_2 \bullet, \eta, k, j \mid D, p, q]}{[A[\circ\circ] \to B[\,]\ C[\circ\circ] \bullet, \eta, i, j \mid D, p, q]}$$

- Un paso $\mathcal{D}_{\mathrm{CYK}'}^{[\circ\circ][\,][\circ\circ]}$ es equivalente a una secuencia de pasos formada por un paso $\mathcal{D}_{\mathrm{ECYK}}^{\mathrm{Init}}$, un paso $\mathcal{D}_{\mathrm{ECYK}}^{\mathrm{Comp}[\circ\circ][\circ\circ]}$ y un paso $\mathcal{D}_{\mathrm{ECYK}}^{\mathrm{Comp}[\,]}$:

$$\frac{}{[A[\circ\circ] \to \bullet B[\circ\circ]; C[\,], -, i, i \mid -, -, -]}$$

$$\frac{[A[\circ\circ] \to \bullet B[\circ\circ]\ C[\,], -, i, i \mid -, -, -],\ [\boldsymbol{B} \to \Upsilon_1 \bullet, \eta, i, k \mid D, p, q]}{[A[\circ\circ] \to B[\,] \bullet C[\circ\circ], \eta, i, k \mid D, p, q]}$$

$$\frac{[A[\circ\circ] \to B[\,] \bullet C[\circ\circ], \eta, i, k \mid D, p, q],\ [\boldsymbol{C} \to \Upsilon_2 \bullet, -, k, j \mid -, -, -]}{[A[\circ\circ] \to B[\,]\ C[\circ\circ] \bullet, \eta, i, j \mid D, p, q]}$$

- Un paso $\mathcal{D}_{\mathrm{CYK}'}^{[\circ\circ][\,][\circ\circ\gamma]}$ es equivalente a la secuencia de pasos formada por un paso $\mathcal{D}_{\mathrm{ECYK}}^{\mathrm{Init}}$, un paso $\mathcal{D}_{\mathrm{ECYK}}^{\mathrm{Comp}[\,]}$ y un paso $\mathcal{D}_{\mathrm{ECYK}}^{\mathrm{Comp}[\circ\circ][\circ\circ\gamma]}$:

$$\frac{}{[A[\circ\circ] \to \bullet B[\,]\ C[\circ\circ\gamma], -, i, i \mid -, -, -]}$$

$$\frac{[A[\circ\circ] \to \bullet B[\,]\ C[\circ\circ\gamma], -, i, i \mid -, -, -],\ [\boldsymbol{B} \to \Upsilon_1 \bullet, -, i, k \mid -, -, -],}{[A[\circ\circ] \to B[\,] \bullet C[\circ\circ\gamma], -, i, k \mid -, -, -]}$$

$$\frac{\begin{array}{l}[A[\circ\circ] \to B[\,] \bullet C[\circ\circ\gamma], -, i, k \mid -, -, -],\\ [\boldsymbol{C} \to \Upsilon_2 \bullet, \gamma, k, j \mid D, p, q],\\ [\boldsymbol{D} \to \Upsilon_3 \bullet, \eta, p, q \mid E, r, s]\end{array}}{[A[\circ\circ] \to B[\,] \bullet C[\circ\circ\gamma], \eta, i, j \mid E, r, s]}$$

- Un paso deductivo $\mathcal{D}_{\mathrm{CYK}'}^{[\circ\circ][\circ\circ\gamma][\,]}$ es equivalente a la secuencia de pasos deductivo constituida por un paso $\mathcal{D}_{\mathrm{ECYK}}^{\mathrm{Init}}$, un paso $\mathcal{D}_{\mathrm{ECYK}}^{\mathrm{Comp}[\circ\circ][\circ\circ\gamma]}$ y un paso $\mathcal{D}_{\mathrm{ECYK}}^{\mathrm{Comp}[\,]}$:

$$\frac{}{[A[\circ\circ] \to \bullet B[\circ\circ\gamma]; C[\,], -, i, i \mid -, -, -]}$$

$$\frac{\begin{array}{l}[A[\circ\circ] \to \bullet B[\circ\circ\gamma]; C[\,], -, i, i \mid -, -, -, -],\\ [\boldsymbol{B} \to \Upsilon_1 \bullet, \gamma, i, k \mid D, p, q],\\ [\boldsymbol{D} \to \Upsilon_2 \bullet, \eta, p, q \mid E, r, s]\end{array}}{[A[\circ\circ] \to B[\circ\circ\gamma] \bullet C[\,], \eta, i, k \mid E, r, s]}$$

$$\frac{[A[\circ\circ] \to B[\circ\circ\gamma] \bullet C[\,], \eta, i, k \mid E, r, s],\ [\boldsymbol{C} \to \Upsilon_3 \bullet, -, k, j \mid -, -, -],}{[A[\circ\circ] \to B[\circ\circ\gamma]\ C[\,] \bullet, \eta, i, j \mid E, r, s]}$$

El esquema de análisis sintáctico **ECYK** está definido para gramáticas lineales de índices en las cuales ningún ninguna producción puede tener más de dos elementos en su lado derecho mientras que el esquema de análisis **buE** está definido para cualquier LIG. Es fácil mostrar que **ECYK** $\stackrel{\text{ext}}{\Longrightarrow}$ **buE** puesto que **ECYK**$(\mathcal{L}) =$ **buE**$(\mathcal{L})$ es cierto para toda gramática lineal de índices ya que por definición $\mathbb{P}_{\text{ECYK}} = \mathbb{P}_{\text{buE}}$. □

La complejidad espacial con respecto a la cadena de entrada del algoritmo definido por el esquema de análisis **buE** es $\mathcal{O}(n^4)$ puesto que cada ítem almacena 4 posiciones de la cadena de entrada. La complejidad temporal con respecto a la cadena de entrada es $\mathcal{O}(n^6)$ y viene dada por los pasos $\mathcal{D}_{\text{buE}}^{\text{Comp}[\circ\circ][\circ\circ\gamma]}$. Aunque dichos pasos manipulan 7 posiciones de la cadena de entrada, mediante aplicación parcial cada uno de ellos puede descomponerse en una secuencia de pasos, cada uno de ellos manipulando a lo sumo 6 posiciones con respecto a la cadena de entrada.

4.3 Algoritmo de tipo Earley sin la propiedad del prefijo válido

El algoritmo descrito por el esquema de análisis sintáctico **buE** es totalmente ascendente en el sentido de que no toma en consideración si la parte de la cadena de entrada que se reconoce en cada ítem es derivable del axioma de la gramática. Los algoritmos de tipo Earley limitan el número de ítems generados mediante la utilización de predicción, que permite determinar qué producciones son candidatas a formar parte de la derivación atendiendo a la derivabilidad a partir del axioma.

En primer lugar, consideraremos que la predicción se realiza únicamente atendiendo al esqueleto independiente del contexto de la gramática lineal de índices, obteniendo un esquema de análisis que denominaremos **E** y que se deriva del esquema **buE** mediante la aplicación de un filtrado dinámico:

- El paso deductivo Init sólo contendrá producciones cuyo lado izquierdo se refiera al axioma de la gramática.
- En lugar de generar ítems de la forma $[\boldsymbol{A} \to \bullet\Upsilon, -, i, i \mid -, -, -]$ para todas las posibles $\boldsymbol{A} \to \Upsilon \in P$ y todas las posibles posiciones i y j de la cadena de entrada, se generarán únicamente aquellos ítems que involucren producciones cuyo esqueleto independiente del contexto sea relevante durante el proceso de análisis. Dicha tarea será encomendada al conjunto de pasos deductivos Pred.

En lo que respecta a los ítems, los del esquema **E** se definen como los del esquema **buE**.

Esquema de análisis sintáctico 4.3 El sistema de análisis $\mathbb{P}_{\text{E}}$ que se corresponde con el algoritmo de análisis de tipo Earley para una gramática lineal de índices $\mathcal{L}$ y una cadena de entrada $a_1 \dots a_n$ se define como sigue:

$$\mathcal{I}_{\text{E}} = \mathcal{I}_{\text{buE}}$$

$$\mathcal{D}_{\text{E}}^{\text{Init}} = \frac{}{[\boldsymbol{S} \to \bullet\Upsilon, -, 0, 0 \mid -, -, -]}$$

$$\mathcal{D}_{\text{E}}^{\text{Scan}} = \mathcal{D}_{\text{buE}}^{\text{Scan}}$$

$$\mathcal{D}_{\text{E}}^{\text{Pred}} = \frac{[\boldsymbol{A} \to \Upsilon_1 \bullet \boldsymbol{B}\, \Upsilon_2, \gamma, i, j \mid C, p, q]}{[\boldsymbol{B} \to \bullet\Upsilon_3, -, j, j \mid -, -, -]}$$

$$\mathcal{D}_{\mathrm{E}}^{\mathrm{Comp}[\,]} = \mathcal{D}_{\mathrm{buE}}^{\mathrm{Comp}[\,]}$$
$$\mathcal{D}_{\mathrm{E}}^{\mathrm{Comp}[\circ\circ\gamma][\circ\circ]} = \mathcal{D}_{\mathrm{buE}}^{\mathrm{Comp}[\circ\circ\gamma][\circ\circ]}$$
$$\mathcal{D}_{\mathrm{E}}^{\mathrm{Comp}[\circ\circ][\circ\circ]} = \mathcal{D}_{\mathrm{buE}}^{\mathrm{Comp}[\circ\circ][\circ\circ]}$$
$$\mathcal{D}_{\mathrm{E}}^{\mathrm{Comp}[\circ\circ][\circ\circ\gamma]} = \mathcal{D}_{\mathrm{buE}}^{\mathrm{Comp}[\circ\circ][\circ\circ\gamma]}$$
$$\mathcal{D}_{\mathrm{E}} = \mathcal{D}_{\mathrm{E}}^{\mathrm{Init}} \cup \mathcal{D}_{\mathrm{E}}^{\mathrm{Scan}} \cup \mathcal{D}_{\mathrm{E}}^{\mathrm{Pred}} \cup \mathcal{D}_{\mathrm{E}}^{\mathrm{Comp}[\,]} \cup \mathcal{D}_{\mathrm{E}}^{\mathrm{Comp}[\circ\circ\gamma][\circ\circ]} \cup \mathcal{D}_{\mathrm{E}}^{\mathrm{Comp}[\circ\circ][\circ\circ]} \cup \mathcal{D}_{\mathrm{E}}^{\mathrm{Comp}[\circ\circ][\circ\circ\gamma]}$$
$$\mathcal{F}_{\mathrm{E}} = \mathcal{F}_{\mathrm{buE}}$$

§

Proposición 4.2 **buE** $\stackrel{\mathbf{df}}{\Longrightarrow}$ **E**.

Demostración:

Para demostrar que el esquema de análisis sintáctico **E** es el resultado de aplicar un filtrado dinámico al esquema de análisis **buE**, debemos demostrar que $\mathcal{I}_{\mathrm{buE}} \supseteq \mathcal{I}_{\mathrm{E}}$ y que $\vdash_{\mathrm{buE}} \supseteq \vdash_{\mathrm{E}}$ para los sistemas de análisis sintáctico $\mathbb{P}_{\mathrm{buE}}$ y $\mathbb{P}_{\mathrm{E}}$. Lo primero es cierto por definición puesto que $\mathcal{I}_{\mathrm{buE}} = \mathcal{I}_{\mathrm{E}}$. Respecto a lo segundo es suficiente con mostrar que $\vdash_{\mathrm{buE}} \supseteq \mathcal{D}_{\mathrm{E}}$.

Los pasos deductivos en $\mathcal{D}_{\mathrm{E}}^{\mathrm{Scan}}$, $\mathcal{D}_{\mathrm{E}}^{\mathrm{Comp}[\,]}$, $\mathcal{D}_{\mathrm{E}}^{\mathrm{Comp}[\circ\circ\gamma][\circ\circ]}$, $\mathcal{D}_{\mathrm{E}}^{\mathrm{Comp}[\circ\circ][\circ\circ]}$ y $\mathcal{D}_{\mathrm{E}}^{\mathrm{Comp}[\circ\circ][\circ\circ\gamma]}$ son idénticos a sus homónimos del sistema de análisis sintáctico $\mathbb{P}_{\mathrm{buE}}$. El conjunto de pasos deductivos $\mathcal{D}_{\mathrm{E}}^{\mathrm{Init}}$ es un subconjunto de $\mathcal{D}_{\mathrm{bu}}^{\mathrm{Init}}$. Con respecto a los pasos deductivos en $\mathcal{D}_{\mathrm{E}}^{\mathrm{Pred}}$ tenemos que dado un paso deductivo

$$\frac{[\boldsymbol{A} \to \Upsilon_1 \bullet \boldsymbol{B}\ \Upsilon_2, \gamma, i, j \mid C, p, q]}{[\boldsymbol{B} \to \bullet\Upsilon_3, -, j, j \mid -, -, -]} \quad \in \mathcal{D}_{\mathrm{E}}^{\mathrm{Pred}}$$

existe un paso deductivo

$$\frac{}{[\boldsymbol{B} \to \bullet\Upsilon_3, -, j, j \mid -, -, -]} \quad \in \mathcal{D}_{\mathrm{buE}}^{\mathrm{Pred}}$$

y por tanto existe la inferencia

$$[\boldsymbol{A} \to \Upsilon_1 \bullet \boldsymbol{B}\ \Upsilon_2, \gamma, i, j \mid C, p, q] \ \vdash_{\mathrm{buE}} \ [\boldsymbol{B} \to \bullet\Upsilon_3, -, j, j \mid -, -, -]$$

□

El algoritmo descrito por el esquema de análisis sintáctico **E**, que mantiene una complejidad espacial $\mathcal{O}(n^4)$ y una complejidad temporal $\mathcal{O}(n^6)$ con respecto a la cadena de entrada, está muy relacionado con el algoritmo de tipo Earley descrito por Schabes y Shieber en [175] aunque este último sólo es aplicable a una clase específica de gramáticas lineales de índices obtenida a partir de una gramática de adjunción de árboles. Sin embargo, ambos comparten una característica muy importante, como es que el tipo de predicción realizado es muy poco potente puesto que no toma en consideración el contenido de la pila de índices. Aunque aparentemente el algoritmo propuesto por Schabes y Shieber en [175] utiliza información de la pila de índices para realizar la predicción, un análisis más profundo nos lleva a la conclusión de que esto no es realmente así. Lo que realmente ocurre es que dichos autores optaron, a la hora de definir la traducción de TAG a LIG, por almacenar el esqueleto independiente del contexto de los árboles elementales en las pilas de índices, reduciendo el conjunto de no-terminales de la LIG resultante a $\{t, b\}$. En la tabla 4.1 se muestran las producciones propuestas por Schabes y Shieber junto con su equivalente que utiliza el conjunto de nodos de la gramática de adjunción original como conjunto de no-terminales. Más precisamente, por cada nodo elemental η definimos dos no-terminales η^t y η^b.

Producción	**Original**	**Equivalente**
dom. inmediata (espina)	$b[\circ\circ\eta] \to t[\eta_1] \dots t[\circ\circ\eta_s] \dots t[\eta_n]$	$\eta^b[\circ\circ] \to \eta_1^t[\,] \dots \eta_s^t[\circ\circ] \dots \eta_n^t[\,]$
dom. inmediata (no espina)	$b[\eta] \to t[\eta_1] \dots t[\eta_n]$	$\eta^b[\,] \to \eta_1^t[\,] \dots \eta_n^t[\,]$
No adjunción	$t[\circ\circ\eta] \to b[\circ\circ\eta]$	$\eta^t[\circ\circ] \to \eta^b[\circ\circ]$
Adjunción predicativa	$t[\circ\circ\eta] \to t[\circ\circ\eta\eta_r]$	$\eta^t[\circ\circ] \to \eta_r^t[\circ\circ\eta]$
Adjunción de modificador	$b[\circ\circ\eta] \to t[\circ\circ\eta\eta_r]$	$\eta^b[\circ\circ] \to \eta_r^t[\circ\circ\eta]$
Pie	$b[\circ\circ\eta\eta_f] \to b[\circ\circ\eta]$	$\eta_f^b[\circ\circ\eta] \to \eta^b[\circ\circ]$
Sustitución	$t[\eta] \to t[n_r]$	$\eta^t[\,] \to n_r^t[\,]$

Tabla 4.1: Producciones propuestas por Schabes y Shieber

4.4 Algoritmos de tipo Earley con la propiedad del prefijo válido

Por la definición del formalismo de las gramáticas lineales de índices sabemos que una producción con $A[\circ\circ\gamma]$ como elemento del lado izquierdo sólo será útil en una derivación si se cumple la condición

$$S[\,] \stackrel{*}{\Rightarrow} w\ A[\alpha\gamma]\ \Upsilon$$

donde $\alpha \in V_I^*$ y $w \in V_T^*$. La comprobación de esta condición en los pasos Pred restringiría mucho más el número de ítems que contienen producciones con punto de la forma $\boldsymbol{A} \to \bullet\Upsilon$. Un resultado importante es que aquellos algoritmos que no verifiquen el cumplimiento de dicha condición en el momento de realizar la predicción no poseerán la propiedad del prefijo válido[1]. En consecuencia, tanto el algoritmo descrito por **E** como el algoritmo descrito por Schabes y Shieber no poseen dicha propiedad.

Para obtener un algoritmo de tipo Earley con la propiedad del prefijo válido es preciso modificar los pasos Pred de modo que predigan información acerca de las pilas de índices. Para ello también será necesario modificar la forma de los ítems para permitir seguir el rastro de las pilas de índices que se van prediciendo. Definiremos por tanto un nuevo conjunto de ítems de la forma

$$[E, h \mid \boldsymbol{A} \to \Upsilon_1 \bullet \Upsilon_2, \gamma, i, j \mid B, p, q]$$

que representan invariablemente alguno de los siguientes tipos de derivaciones:

- $S[\,] \stackrel{*}{\Rightarrow} a_1 \dots a_h\, E[\alpha]\, \Upsilon_4 \stackrel{*}{\Rightarrow} a_1 \dots a_h \dots a_i\, A[\alpha\gamma]\, \Upsilon_3 \Upsilon_4 \stackrel{*}{\Rightarrow} a_1 \dots a_h \dots a_i \dots a_p\, B[\alpha]\, a_{q+1} \dots a_j\, \Upsilon_2 \Upsilon_3 \Upsilon_4$ si y sólo si $(B, p, q) \neq (-, -, -)$, donde $A[\alpha\gamma]$ es un descendiente dependiente de $E[\alpha]$ y $B[\alpha]$ es un descendiente dependiente de $A[\alpha\gamma]$. Este tipo de derivación se corresponde con la compleción del hijo dependiente de una regla que tiene el no-terminal A como lado izquierdo. Además, la pila asociada a dicho no-terminal no debe estar vacía.

- $S[\,] \stackrel{*}{\Rightarrow} a_1 \dots a_h\, E[\alpha]\, \Upsilon_4 \stackrel{*}{\Rightarrow} a_1 \dots a_h \dots a_i\ A[\alpha\gamma]\ \Upsilon_3 \Upsilon_4 \stackrel{*}{\Rightarrow} a_1 \dots a_h \dots a_i \dots a_j\ \Upsilon_2 \Upsilon_3 \Upsilon_4$ si y sólo si $(E, h) \neq (-, -)$ y $(B, p, q) = (-, -, -)$, donde $A[\alpha\gamma]$ es un descendiente dependiente de $E[\alpha]$ y Υ_1 no contiene el hijo dependiente de $A[\alpha\gamma]$. Este tipo de derivación se refiere a la predicción del no-terminal A con una pila de índices no vacía.

[1]La propiedad del prefijo válido se discute en la sección 3.4.

- $S[\ ] \overset{*}{\Rightarrow} a_1 \ldots a_i A[\ ]\Upsilon_4 \overset{*}{\Rightarrow} a_1 \ldots a_i \ldots a_j\, \Upsilon_2\Upsilon_4$ si y sólo si $(E,h) = (-,-)$, $\gamma = -$ y $(B,p,q) = (-,-,-)$. Si Υ_1 incluye al hijo dependiente de $A[\,]$ entones las pilas asociadas a $A[\,]$ y al hijo dependiente están vacías. Este tipo de derivación se refiere a la predicción o compleción del no-terminal A con una pila de índices vacía.

Observamos que el nuevo conjunto de ítems así definido es un refinamiento de los ítems del esquema **E**, de tal modo que el elemento γ se utiliza para almacenar la cima de la pila de índices predicha en el caso de que el ítem represente una predicción (recordemos que en el esquema **E** los ítems resultado de una predicción tenían $\gamma = -$). Por otra parte, el par de elementos (E,h) permite seguir la traza del ítem involucrado en la predicción. En principio podríamos suponer que para guardar dicha traza necesitaríamos almacenar una cuádrupla (E,η,h,k). Sin embargo, por la propiedad de independencia del contexto de LIG (definición 2.1, página 32) no es necesario almacenar η puesto que la derivación será válida independiente del resto de la pila de índices. Respecto al índice k, no es necesario puesto que al ser todos los ítems predichos en el esquema **E** de la forma $[\boldsymbol{A} \to \bullet\Upsilon, h, h \mid B, p, q]$, su presencia sería redundante.

Con respecto a los pasos deductivos, será necesario adaptar los pasos de compleción para que manipulen adecuadamente los nuevos componentes E y h y refinar los pasos predictivos con el fin de diferenciar los diferentes casos que se pueden dar.

Esquema de análisis sintáctico 4.4 El sistema de análisis $\mathbb{P}_{\text{Earley}_1}$ que se corresponde con el algoritmo de análisis de tipo Earley que preserva la propiedad del prefijo válido para una gramática lineal de índices $\mathcal{L}$ y una cadena de entrada $a_1 \ldots a_n$ se define como sigue:

$$\mathcal{I}_{\text{Earley}_1} = \left\{ \begin{array}{ll} [E,h \mid \boldsymbol{A} \to \Upsilon_1 \bullet \Upsilon_2, \gamma, i, j \mid B,p,q] \mid & \boldsymbol{A} \to \Upsilon_1\Upsilon_2 \in P,\ B, C \in V_N,\ \gamma \in V_I, \\ & 0 \le h \le i \le j\ ,\ \ (p,q) \le (i,j) \end{array} \right\}$$

$$\mathcal{D}^{\text{Init}}_{\text{Earley}_1} = \frac{}{[-,- \mid \boldsymbol{S} \to \bullet\Upsilon, -, 0, 0 \mid -,-,-]}$$

$$\mathcal{D}^{\text{Scan}}_{\text{Earley}_1} = \frac{\begin{array}{c}[-,- \mid A[\,] \to \bullet a, -, j, j \mid -,-,-], \\ [a, j, j+1]\end{array}}{[-,- \mid A[\,] \to a\bullet, -, j, j+1 \mid -,-,-]}$$

$$\mathcal{D}^{\text{Pred}[\,]}_{\text{Earley}_1} = \frac{[E,h \mid \boldsymbol{A} \to \Upsilon_1 \bullet B[\,]\ \Upsilon_2, \gamma, i, j \mid C,p,q]}{[-,- \mid \boldsymbol{B} \to \bullet\Upsilon_3, -, j, j \mid -,-,-]} \quad \boldsymbol{B} \in \{B[\circ\circ], B[\,]\}$$

$$\mathcal{D}^{\text{Pred}[\circ\circ\gamma][\circ\circ]}_{\text{Earley}_1} = \frac{\begin{array}{c}[E,h \mid A[\circ\circ\gamma] \to \Upsilon_1 \bullet B[\circ\circ]\ \Upsilon_2, \gamma, i, j \mid -,-,-], \\ [M,m \mid \boldsymbol{E} \to \bullet\Upsilon_3, \gamma', h, h \mid -,-,-]\end{array}}{[M,m \mid \boldsymbol{B} \to \bullet\Upsilon_4, \gamma', j, j \mid -,-,-]} \quad \begin{array}{l}\boldsymbol{B} \in \{B[\circ\circ\gamma'], B[\circ\circ]\} \text{ sii } \gamma' \neq - \\ \boldsymbol{B} = B[\,] \text{ sii } \gamma' = -\end{array}$$

$$\mathcal{D}^{\text{Pred}[\circ\circ][\circ\circ]}_{\text{Earley}_1} = \frac{[E,h \mid A[\circ\circ] \to \Upsilon_1 \bullet B[\circ\circ]\ \Upsilon_2, \gamma, i, j \mid -,-,-]}{[E,h \mid \boldsymbol{B} \to \bullet\Upsilon_3, \gamma, j, j \mid -,-,-]} \quad \begin{array}{l}\boldsymbol{B} \in \{B[\circ\circ\gamma], B[\circ\circ]\} \text{ sii } \gamma \neq - \\ \boldsymbol{B} = B[\,] \text{ sii } \gamma = -\end{array}$$

$$\mathcal{D}^{\text{Pred}[\circ\circ][\circ\circ\gamma]}_{\text{Earley}_1} = \frac{[E,h \mid A[\circ\circ] \to \Upsilon_1 \bullet B[\circ\circ\gamma]\ \Upsilon_2, \gamma', i, j \mid -,-,-]}{[A,i \mid \boldsymbol{B} \to \bullet\Upsilon_3, \gamma, j, j \mid -,-,-]} \quad \boldsymbol{B} \in \{B[\circ\circ\gamma], B[\circ\circ]\}$$

$$\mathcal{D}^{\text{Comp}[\,]}_{\text{Earley}_1} = \frac{\begin{array}{l}[E,h \mid \boldsymbol{A} \to \Upsilon_1 \bullet B[\,]\ \Upsilon_2, \gamma, i, j \mid C, p, q],\\ [-,- \mid \boldsymbol{B} \to \Upsilon_3 \bullet, -, j, k \mid -,-,-]\end{array}}{[E,h \mid \boldsymbol{A} \to \Upsilon_1\ B[\,] \bullet \Upsilon_2, \gamma, i, k \mid C, p, q]}$$

$$\mathcal{D}^{\text{Comp}[\circ\circ\gamma][\circ\circ]}_{\text{Earley}_1} = \frac{\begin{array}{l}[E,h \mid A[\circ\circ\gamma] \to \Upsilon_1 \bullet B[\circ\circ]\ \Upsilon_2, \gamma, i, j \mid -,-,-],\\ [M,m \mid \boldsymbol{E} \to \bullet \Upsilon_3, \gamma', h, h \mid -,-,-],\\ [M,m \mid \boldsymbol{B} \to \Upsilon_4 \bullet, \gamma', j, k \mid C, p, q]\end{array}}{[E,h \mid A[\circ\circ\gamma] \to \Upsilon_1\ B[\circ\circ] \bullet \Upsilon_2, \gamma, i, k \mid B, j, k]}$$

$$\mathcal{D}^{\text{Comp}[\circ\circ][\circ\circ]}_{\text{Earley}_1} = \frac{\begin{array}{l}[E,h \mid A[\circ\circ] \to \Upsilon_1 \bullet B[\circ\circ]\ \Upsilon_2, \gamma, i, j \mid -,-,-],\\ [E,h \mid \boldsymbol{B} \to \Upsilon_3 \bullet, \gamma, j, k \mid C, p, q]\end{array}}{[E,h \mid A[\circ\circ] \to \Upsilon_1\ B[\circ\circ] \bullet \Upsilon_2, \gamma, i, k \mid C, p, q]}$$

$$\mathcal{D}^{\text{Comp}[\circ\circ][\circ\circ\gamma]}_{\text{Earley}_1} = \frac{\begin{array}{l}[E,h \mid A[\circ\circ] \to \Upsilon_1 \bullet B[\circ\circ\gamma]\ \Upsilon_2\ , \gamma', i, j \mid -,-,-],\\ [A,i \mid \boldsymbol{B} \to \Upsilon_3 \bullet\ , \gamma, j, k \mid C, p, q],\\ [E,h \mid \boldsymbol{C} \to \Upsilon_4 \bullet, \gamma', p, q \mid D, r, s]\end{array}}{[E,h \mid A[\circ\circ] \to \Upsilon_1\ B[\circ\circ\gamma] \bullet \Upsilon_2, \gamma', i, k \mid D, r, s]}$$

$$\begin{aligned}\mathcal{D}_{\text{Earley}_1} = {} & \mathcal{D}^{\text{Init}}_{\text{Earley}_1} \cup \mathcal{D}^{\text{Scan}}_{\text{Earley}_1} \cup \mathcal{D}^{\text{Pred}[\,]}_{\text{Earley}_1} \cup \mathcal{D}^{\text{Pred}[\circ\circ\gamma][\circ\circ]}_{\text{Earley}_1} \cup \mathcal{D}^{\text{Pred}[\circ\circ][\circ\circ]}_{\text{Earley}_1} \cup \mathcal{D}^{\text{Pred}[\circ\circ][\circ\circ\gamma]}_{\text{Earley}_1} \cup \\ & \mathcal{D}^{\text{Comp}[\,]}_{\text{Earley}_1} \cup \mathcal{D}^{\text{Comp}[\circ\circ\gamma][\circ\circ]}_{\text{Earley}_1} \cup \mathcal{D}^{\text{Comp}[\circ\circ][\circ\circ]}_{\text{Earley}_1} \cup \mathcal{D}^{\text{Comp}[\circ\circ][\circ\circ\gamma]}_{\text{Earley}_1}\end{aligned}$$

$$\mathcal{F}_{\text{Earley}_1} = \{\ [-,- \mid \boldsymbol{S} \to \Upsilon \bullet\ , -, 0, n \mid -,-,-]\ \}$$

§

Proposición 4.3 *El algoritmo representado por el esquema de análisis sintáctico* **Earley₁** *es correcto y completo.*

Demostración:

La demostración de la corrección se obtiene verificando mediante inducción en la longitud de las secuencias deductivas que los ítems siempre mantienen la invariante. El caso base viene definido por el conjunto de pasos deductivos Init, que trivialmente satisfacen la invariante. El paso de inducción se realiza verificando que, para cada paso deductivo, si se cumple la invariante para los ítems antecedentes entonces también se cumple para el ítem consecuente.

La completud se demuestra verificando que dada una derivación más a la izquierda de la gramática existe una secuencia deductiva en el esquema de análisis que representa dicha derivación. Para ello aplicamos inducción sobre la longitud de la derivación. Como caso base tenemos que una derivación de longitud 1 se corresponde con la aplicación de un paso Init donde $\Upsilon = \epsilon$. Como hipótesis de inducción suponemos que para toda derivación más a la izquierda de longitud l existe una secuencia deductiva que la representa. Como paso de inducción tenemos que la secuencia deductiva de toda derivación más a la izquierda de longitud $l+1$ se obtiene a partir de la secuencia deductiva correspondiente a la derivación de longitud l más un conjunto de pasos deductivos de tipo Pred y/o Scan y/o Comp. □

Proposición 4.4 *El algoritmo representado por el esquema de análisis sintáctico* **Earley$_1$** *satisface la propiedad del prefijo válido.*

Demostración:

La prueba de esta proposición es un corolario de la prueba de la corrección y completud: si un ítem $[E, h \mid \boldsymbol{A} \rightarrow \Upsilon_1 \bullet \Upsilon_2, \gamma, i, j \mid B, p, q]$ ha sido generado entonces es que existe una derivación $S[\,] \stackrel{*}{\Rightarrow} a_1 \dots a_j \Upsilon_2 \Upsilon_3 \Upsilon_4$ y por tanto $S[\,] \stackrel{*}{\Rightarrow} a_1 \dots a_j w$, donde w se obtiene de una derivación $\Upsilon_2 \Upsilon_3 \Upsilon_4 \stackrel{*}{\Rightarrow} w$. □

Proposición 4.5 $\mathbf{E} \stackrel{\mathbf{sr}}{\Longrightarrow} \mathbf{E'} \stackrel{\mathbf{ir}}{\Longrightarrow} \stackrel{\mathbf{df}}{\Longrightarrow}$ **Earley$_1$**.

Demostración:

Como primer paso definiremos el esquema de análisis sintáctico $\mathbf{E'}$ que se obtiene a partir de $\mathbf{E}$ rompiendo el conjunto de pasos deductivos $\mathcal{D}_{\mathrm{E}}^{\mathrm{Pred}}$ en cuatro conjuntos $\mathcal{D}_{\mathrm{E'}}^{\mathrm{Pred}[\,]}$, $\mathcal{D}_{\mathrm{E'}}^{\mathrm{Pred}[\circ\circ\gamma][\circ\circ]}$, $\mathcal{D}_{\mathrm{E'}}^{\mathrm{Pred}[\circ\circ][\circ\circ]}$y $\mathcal{D}_{\mathrm{E'}}^{\mathrm{Pred}[\circ\circ][\circ\circ\gamma]}$. El sistema de análisis correspondiente se describe a continuación:

$$\mathcal{I}_{\mathrm{E'}} = \mathcal{I}_{\mathrm{E}}$$

$$\mathcal{D}_{\mathrm{E'}}^{\mathrm{Init}} = \mathcal{D}_{\mathrm{E}}^{\mathrm{Init}}$$

$$\mathcal{D}_{\mathrm{E'}}^{\mathrm{Scan}} = \mathcal{D}_{\mathrm{E}}^{\mathrm{Scan}}$$

$$\mathcal{D}_{\mathrm{E'}}^{\mathrm{Pred}[\,]} = \frac{[\boldsymbol{A} \rightarrow \Upsilon_1 \bullet B[\,]\ \Upsilon_2, -, i, j \mid C, p, q]}{[\boldsymbol{B} \rightarrow \bullet \Upsilon_3, -, j, j \mid -, -, -]}$$

$$\mathcal{D}_{\mathrm{E'}}^{\mathrm{Pred}[\circ\circ\gamma][\circ\circ]} = \frac{\begin{array}{l}[A[\circ\circ\gamma] \rightarrow \Upsilon_1 \bullet B[\circ\circ]\ \Upsilon_2, -, i, j \mid -, -, -],\\ [\boldsymbol{E} \rightarrow \bullet \Upsilon_3, -, h, h \mid -, -, -]\end{array}}{[\boldsymbol{B} \rightarrow \bullet \Upsilon_4, -, j, j \mid -, -, -]}$$

$$\mathcal{D}_{\mathrm{E'}}^{\mathrm{Pred}[\circ\circ][\circ\circ]} = \frac{[A[\circ\circ] \rightarrow \Upsilon_1 \bullet B[\circ\circ]\ \Upsilon_2, -, i, j \mid -, -, -]}{[\boldsymbol{B} \rightarrow \bullet \Upsilon_3, -, j, j \mid -, -, -]}$$

$$\mathcal{D}_{\mathrm{E'}}^{\mathrm{Pred}[\circ\circ][\circ\circ\gamma]} = \frac{[A[\circ\circ] \rightarrow \Upsilon_1 \bullet B[\circ\circ\gamma]\ \Upsilon_2, -, i, j \mid -, -, -]}{[\boldsymbol{B} \rightarrow \bullet \Upsilon_3, -, j, j \mid -, -, -]}$$

$$\mathcal{D}_{\mathrm{E'}}^{\mathrm{Comp}[\,]} = \mathcal{D}_{\mathrm{E}}^{\mathrm{Comp}[\,]}$$

$$\mathcal{D}_{\mathrm{E'}}^{\mathrm{Comp}[\circ\circ\gamma][\circ\circ]} = \mathcal{D}_{\mathrm{E}}^{\mathrm{Comp}[\circ\circ\gamma][\circ\circ]}$$

$$\mathcal{D}_{\mathrm{E'}}^{\mathrm{Comp}[\circ\circ][\circ\circ]} = \mathcal{D}_{\mathrm{E}}^{\mathrm{Comp}[\circ\circ][\circ\circ]}$$

$$\mathcal{D}_{\mathrm{E'}}^{\mathrm{Comp}[\circ\circ][\circ\circ\gamma]} = \mathcal{D}_{\mathrm{E}}^{\mathrm{Comp}[\circ\circ][\circ\circ\gamma]}$$

$$\mathcal{D}_{\mathrm{E'}} = \mathcal{D}_{\mathrm{E'}}^{\mathrm{Init}} \cup \mathcal{D}_{\mathrm{E'}}^{\mathrm{Scan}} \cup \mathcal{D}_{\mathrm{E'}}^{\mathrm{Pred}} \cup \mathcal{D}_{\mathrm{E'}}^{\mathrm{Comp}[\,]} \cup \mathcal{D}_{\mathrm{E'}}^{\mathrm{Comp}[\circ\circ\gamma][\circ\circ]} \cup \mathcal{D}_{\mathrm{E'}}^{\mathrm{Comp}[\circ\circ][\circ\circ]} \cup \mathcal{D}_{\mathrm{E'}}^{\mathrm{Comp}[\circ\circ][\circ\circ\gamma]}$$

$$\mathcal{F}_{\mathrm{E'}} = \mathcal{F}_{\mathrm{E}}$$

Las condiciones a verificar son que $\mathcal{I}_{\mathrm{E}} \subseteq \mathcal{I}_{\mathrm{E'}}$ y que $\stackrel{*}{\vdash}_{\mathrm{E}} \subseteq \stackrel{*}{\vdash}_{\mathrm{E'}}$. La primera condición se verifica por la propia definición de los ítems mientras que la segunda se obtiene fácilmente puesto que los diferentes pasos predicitvos lo único que hacen es especificar las diferentes formas que puede tener la producción que aparece en el ítem antecedente de cada uno de ellos. En el caso concreto de los pasos en $\mathcal{D}_{\mathrm{E'}}^{\mathrm{Pred}[\circ\circ\gamma][\circ\circ]}$, el segundo ítem antecedente es totalmente redundante en este esquema de análisis puesto que su existencia se deriva de la existencia del primer ítem antecedente.

Para demostrar que el esquema de análisis **Earley₁** es derivable del esquema de análisis **E′** mediante refinamiento de los ítems y un filtro dinámico definiremos la siguiente función f de tal modo que

$$f([E,h \mid \boldsymbol{A} \to \Upsilon_1 \bullet \Upsilon_2, \gamma, i, j \mid -,-,-]) = [\boldsymbol{A} \to \Upsilon_1 \bullet \Upsilon_2, -, i, j \mid B, p, q]$$

mientras que en cualquier otro caso

$$f([E,h \mid \boldsymbol{A} \to \Upsilon_1 \bullet \Upsilon_2, \gamma, i, j \mid B,p,q]) = [\boldsymbol{A} \to \Upsilon_1 \bullet \Upsilon_2, \gamma, i, j \mid B, p, q]$$

De la definición de f se obtiene directamente que $\mathcal{I}_{\mathrm{E}'} = f(\mathcal{I}_{\mathrm{Earley}_1})$ y que $\Delta_{\mathrm{E}'} = f(\Delta_{\mathrm{Earley}_1})$ por inducción en la longitud de las secuencias de derivación. En consecuencia, $\mathbb{P}_{\mathrm{E}'} \xrightarrow{\mathbf{ir}} \mathbb{P}_{\mathrm{Earley}_1}$, con lo que hemos probado lo que pretendíamos.

Con respecto al filtrado dinámico, se cumple que $\vdash_{\mathrm{E}'} \supseteq \mathcal{D}_{\mathrm{Earley}_1}$ por la propia definición de los pasos deductivos en **Earley₁**. □

La complejidad espacial con respecto a la longitud n de la cadena de entrada del algoritmo descrito por el esquema de análisis **Earley₁** es $\mathcal{O}(n^5)$ puesto que cada ítem hace uso de 5 posiciones. La complejidad temporal con respecto a la cadena de entrada es $\mathcal{O}(n^7)$ y viene dada por el conjunto de pasos deductivos $\mathcal{D}_{\mathrm{Earley}_1}^{\mathrm{Comp}[\circ\circ][\circ\circ\gamma]}$. Los ítem involucrados en dichos pasos hacen referencia a 8 posiciones de la cadena de entrada. Mediante aplicación parcial podemos reducir la complejidad a $\mathcal{O}(n^7)$, que sin embargo resulta superior a la complejidad $\mathcal{O}(n^6)$ obtenida para los algoritmos presentados anteriormente. Para rebajar la complejidad temporal del algoritmo deberemos recurrir a una técnica más sofisticada, similar a la utilizada por Nederhof en [125] para reducir la complejidad de su algoritmo de tipo Earley con la propiedad del prefijo válido para el análisis de TAG. En el caso que nos ocupa, dividiremos cada paso deductivo de $\mathcal{D}_{\mathrm{Earley}_1}^{\mathrm{Comp}[\circ\circ][\circ\circ\gamma]}$ en dos pasos de tal forma que la complejidad de cada uno de ellos sea a lo sumo $\mathcal{O}(n^6)$:

$$\mathcal{D}_{\mathrm{Earley}}^{\mathrm{Comp}[\circ\circ][\circ\circ\gamma]^0} = \frac{\begin{array}{c}[A,i \mid \boldsymbol{B} \to \Upsilon_3 \bullet\ , \gamma, j, k \mid C,p,q],\\ [E,h \mid \boldsymbol{C} \to \Upsilon_4 \bullet, \gamma', p, q \mid D,r,s]\end{array}}{[[\boldsymbol{B} \to \Upsilon_3 \bullet\ , \gamma, j, k \mid D,r,s]]}$$

$$\mathcal{D}_{\mathrm{Earley}}^{\mathrm{Comp}[\circ\circ][\circ\circ\gamma]^1} = \frac{\begin{array}{l}[[\boldsymbol{B} \to \Upsilon_3 \bullet\ , \gamma, j, k \mid D,r,s]],\\ [E,h \mid A[\circ\circ] \to \Upsilon_1 \bullet B[\circ\circ\gamma]\ \Upsilon_2\ , \gamma', i, j \mid -,-,-],\\ [E,h \mid \boldsymbol{C} \to \Upsilon_4 \bullet, \gamma', p, q \mid D,r,s]\end{array}}{[E,h \mid A[\circ\circ] \to \Upsilon_1\ B[\circ\circ\gamma] \bullet \Upsilon_2, \gamma', i, k \mid D,r,s]}$$

El primer paso generará un pseudo-ítem intermedio de la forma $[[\boldsymbol{B} \to \Upsilon_3 \bullet\ , \gamma, j, k \mid D,r,s]]$ que será utilizado como antecedente del segundo paso deductivo y que representa una derivación

$$B[\gamma'\gamma] \stackrel{*}{\Rightarrow} a_{j+1} \dots a_p\, C[\gamma']\, a_{s+1} \dots a_q \stackrel{*}{\Rightarrow} a_{j+1} \dots a_p \dots a_r\ D[\,]\ a_{s+1} \dots a_q \dots a_k$$

para algún γ', p y q. Los pasos de $\mathcal{D}_{\mathrm{Earley}}^{\mathrm{Comp}[\circ\circ][\circ\circ\gamma]^1}$ combinan dicho pseudo-ítem con el ítem $[E,h \mid A[\circ\circ] \to \Upsilon_1 \bullet B[\circ\circ\gamma]\ \Upsilon_2\ , \gamma', i, j \mid -,-,-]$ que representa una derivación

$$S[\,] \stackrel{*}{\Rightarrow} a_1 \dots a_h\, E[\alpha]\, \Upsilon_5 \stackrel{*}{\Rightarrow} a_1 \dots a_h \dots a_i\ A[\alpha\gamma']\ \Upsilon_3\Upsilon_5 \stackrel{*}{\Rightarrow} a_1 \dots a_h \dots a_i \dots a_j\ B[\alpha\gamma'\gamma]\ \Upsilon_2\Upsilon_3\Upsilon_5$$

y con el ítem $[E,h \mid \boldsymbol{C} \to \Upsilon_4 \bullet, \gamma', p, q \mid D,r,s]$ que representa una derivación

$$S[\,] \stackrel{*}{\Rightarrow} a_1 \dots a_h\, E[\alpha]\, \Upsilon_5 \stackrel{*}{\Rightarrow} a_1 \dots a_h \dots a_p\ C[\alpha\gamma']\ \Upsilon_4\Upsilon_5 \stackrel{*}{\Rightarrow} a_1 \dots a_h \dots a_p \dots a_r\ D[\alpha]\ a_{s+1} \dots a_q\ \Upsilon_4\Upsilon_5$$

con lo cual podemos generar un ítem de la forma $[E,h \mid A[\circ\circ] \to \Upsilon_1\ B[\circ\circ\gamma] \bullet \Upsilon_2, \gamma', i, k \mid D, r, s]$ que representa la existencia de una derivación

$$\begin{aligned} S[\,] &\stackrel{*}{\Rightarrow} a_1 \ldots a_h\, E[\alpha]\, \Upsilon_5 \\ &\stackrel{*}{\Rightarrow} a_1 \ldots a_h \ldots a_i\ A[\alpha\gamma']\ \Upsilon_3\Upsilon_5 \\ &\stackrel{*}{\Rightarrow} a_1 \ldots a_h \ldots a_i \ldots a_j\ B[\alpha\gamma'\gamma]\ \Upsilon_2\Upsilon_3\Upsilon_5 \\ &\stackrel{*}{\Rightarrow} a_1 \ldots a_h \ldots a_i \ldots a_j \ldots a_p\ C[\alpha\gamma']\ a_{q+1} \ldots a_k\ \Upsilon_2\Upsilon_3\Upsilon_5 \\ &\stackrel{*}{\Rightarrow} a_1 \ldots a_h \ldots a_i \ldots a_j \ldots a_p \ldots a_r\ D[\alpha]\ a_{s+1} \ldots a_{q+1} \ldots a_k\ \Upsilon_2\Upsilon_3\Upsilon_5 \end{aligned}$$

Esquema de análisis sintáctico 4.5 El sistema de análisis $\mathbb{P}_{\text{Earley}}$ que se corresponde con el algoritmo de análisis de tipo Earley que preserva la propiedad del prefijo válido para una gramática lineal de índices $\mathcal{L}$ y una cadena de entrada $a_1 \ldots a_n$ se define como sigue:

$$\mathcal{I}_{\text{Earley}^{(1)}} = \left\{ \begin{array}{ll} [E,h \mid \boldsymbol{A} \to \Upsilon_1 \bullet \Upsilon_2, \gamma, i, j \mid B, p, q] \mid & \boldsymbol{A} \to \Upsilon_1\Upsilon_2 \in P,\ B, C \in V_N,\ \gamma \in V_I, \\ & 0 \le h \le i \le j\ ,\ \ (p,q) \le (i,j) \end{array} \right\}$$

$$\mathcal{I}_{\text{Earley}^{(2)}} = \left\{\ [[\boldsymbol{A} \to \Upsilon\bullet, \gamma, i, j \mid B, p, q]] \mid \boldsymbol{A} \to \Upsilon \in P,\ B \in V_N,\ \gamma \in V_I,\ i \le j\ ,\ \ (p,q) \le (i,j)\ \right\}$$

$$\mathcal{I}_{\text{Earley}} = \mathcal{I}_{\text{Earley}^{(1)}} \cup \mathcal{I}_{\text{Earley}^{(2)}}$$

$$\mathcal{D}^{\text{Init}}_{\text{Earley}} = \mathcal{D}^{\text{Init}}_{\text{Earley}_1}$$

$$\mathcal{D}^{\text{Scan}}_{\text{Earley}} = \mathcal{D}^{\text{Scan}}_{\text{Earley}_1}$$

$$\mathcal{D}^{\text{Pred}[\,]}_{\text{Earley}} = \mathcal{D}^{\text{Pred}[\,]}_{\text{Earley}_1}$$

$$\mathcal{D}^{\text{Pred}[\circ\circ\gamma][\circ\circ]}_{\text{Earley}} = \mathcal{D}^{\text{Pred}[\circ\circ\gamma][\circ\circ]}_{\text{Earley}_1}$$

$$\mathcal{D}^{\text{Pred}[\circ\circ][\circ\circ]}_{\text{Earley}} = \mathcal{D}^{\text{Pred}[\circ\circ][\circ\circ]}_{\text{Earley}_1}$$

$$\mathcal{D}^{\text{Pred}[\circ\circ][\circ\circ\gamma]}_{\text{Earley}} = \mathcal{D}^{\text{Pred}[\circ\circ][\circ\circ\gamma]}_{\text{Earley}_1}$$

$$\mathcal{D}^{\text{Comp}[\,]}_{\text{Earley}} = \mathcal{D}^{\text{Comp}[\,]}_{\text{Earley}_1}$$

$$\mathcal{D}^{\text{Comp}[\circ\circ\gamma][\circ\circ]}_{\text{Earley}} = \mathcal{D}^{\text{Comp}[\circ\circ\gamma][\circ\circ]}_{\text{Earley}_1}$$

$$\mathcal{D}^{\text{Comp}[\circ\circ][\circ\circ]}_{\text{Earley}} = \mathcal{D}^{\text{Comp}[\circ\circ][\circ\circ]}_{\text{Earley}_1}$$

$$\mathcal{D}^{\text{Comp}[\circ\circ][\circ\circ\gamma]^0}_{\text{Earley}} = \frac{\begin{array}{c} [A, i \mid \boldsymbol{B} \to \Upsilon_3\bullet\ , \gamma, j, k \mid C, p, q], \\ [E, h \mid \boldsymbol{C} \to \Upsilon_4\bullet, \gamma', p, q \mid D, r, s] \end{array}}{[[\boldsymbol{B} \to \Upsilon_3\bullet\ , \gamma, j, k \mid D, r, s]]}$$

$$\mathcal{D}_{\text{Earley}}^{\text{Comp}[\circ\circ][\circ\circ\gamma]^1} = \frac{\begin{array}{l}[[\boldsymbol{B} \to \Upsilon_3 \bullet\ , \gamma, j, k \mid D, r, s]], \\ [E, h \mid A[\circ\circ] \to \Upsilon_1 \bullet B[\circ\circ\gamma]\ \Upsilon_2\ , \gamma', i, j \mid -, -, -], \\ [E, h \mid \boldsymbol{C} \to \Upsilon_4 \bullet, \gamma', p, q \mid D, r, s]\end{array}}{[E, h \mid A[\circ\circ] \to \Upsilon_1\ B[\circ\circ\gamma] \bullet \Upsilon_2, \gamma', i, k \mid D, r, s]}$$

$$\begin{aligned}\mathcal{D}_{\text{Earley}} = \ & \mathcal{D}_{\text{Earley}}^{\text{Init}} \cup \mathcal{D}_{\text{Earley}}^{\text{Scan}} \cup \mathcal{D}_{\text{Earley}}^{\text{Pred}[\,]} \cup \mathcal{D}_{\text{Earley}}^{\text{Pred}[\circ\circ\gamma][\circ\circ]} \cup \mathcal{D}_{\text{Earley}}^{\text{Pred}[\circ\circ][\circ\circ]} \cup \mathcal{D}_{\text{Earley}}^{\text{Pred}[\circ\circ][\circ\circ\gamma]} \cup \\ & \mathcal{D}_{\text{Earley}}^{\text{Comp}[\,]} \cup \mathcal{D}_{\text{Earley}}^{\text{Comp}[\circ\circ\gamma][\circ\circ]} \cup \mathcal{D}_{\text{Earley}}^{\text{Comp}[\circ\circ][\circ\circ]} \cup \mathcal{D}_{\text{Earley}}^{\text{Comp}[\circ\circ][\circ\circ\gamma]^0} \cup \mathcal{D}_{\text{Earley}}^{\text{Comp}[\circ\circ][\circ\circ\gamma]^1}\end{aligned}$$

$$\mathcal{F}_{\text{Earley}} = \mathcal{F}_{\text{Earley}_1}$$

§

Proposición 4.6 Earley$_1$ $\overset{\text{sr}}{\Longrightarrow}$ Earley.

Demostración:

Para demostrar que el esquema de análisis **Earley** puede ser obtenido mediante un refinamiento de los pasos deductivos del esquema **Earley$_1$** debemos probar que para todo sistema de análisis $\mathbb{P}_{\text{Earley}_1}$ y $\mathbb{P}_{\text{Earley}}$ se cumple que $\mathbb{P}_{\text{Earley}_1} \overset{\text{sr}}{\Longrightarrow} \mathbb{P}_{\text{Earley}}$. Ello conlleva demostrar que $\mathcal{I}_{\text{Earley}_1} \subseteq \mathcal{I}_{\text{Earley}}$ y que $\overset{*}{\vdash}_{\text{Earley}_1} \subseteq \overset{*}{\vdash}_{\text{Earley}}$. Lo primero se obtiene directamente puesto que $\mathcal{I}_{\text{Earley}_1} = \mathcal{I}_{\text{Earley}}$ por definición de los sistemas de análisis. Lo segundo se obtiene demostrando que $\mathcal{D}_{\text{Earley}_1} \subseteq \overset{*}{\vdash}_{\text{Earley}}$. El únicos conjunto de pasos deductivos de $\mathbb{P}_{\text{Earley}_1}$ que no se han incorporado directamente en $\mathbb{P}_{\text{Earley}}$ es $\mathcal{D}_{\text{Earley}_1}^{\text{Comp}[\circ\circ][\circ\circ\gamma]}$ pero todo paso perteneciente a ese conjunto es equivalente a la la aplicación de un paso deductivo de $\mathcal{D}_{\text{Earley}}^{\text{Comp}[\circ\circ][\circ\circ\gamma]^0}$ seguido de un paso deductivo del conjunto $\mathcal{D}_{\text{Earley}}^{\text{Comp}[\circ\circ][\circ\circ\gamma]^1}$:

$$\frac{[A, i \mid \boldsymbol{B} \to \Upsilon_3 \bullet\ , \gamma, j, k \mid C, p, q],\ [E, h \mid \boldsymbol{C} \to \Upsilon_4 \bullet, \gamma', p, q \mid D, r, s]}{[[\boldsymbol{B} \to \Upsilon_3 \bullet\ , \gamma, j, k \mid D, r, s]]}$$

$$\frac{\begin{array}{l}[[\boldsymbol{B} \to \Upsilon_3 \bullet\ , \gamma, j, k \mid D, r, s]], \\ [E, h \mid A[\circ\circ] \to \Upsilon_1 \bullet B[\circ\circ\gamma]\ \Upsilon_2\ , \gamma', i, j \mid -, -, -], \\ [E, h \mid \boldsymbol{C} \to \Upsilon_4 \bullet, \gamma', p, q \mid D, r, s]\end{array}}{[E, h \mid A[\circ\circ] \to \Upsilon_1\ B[\circ\circ\gamma] \bullet \Upsilon_2, \gamma', i, k \mid D, r, s]}$$

□

4.5 El bosque de análisis

Los algoritmos anteriores tal y como han sido descritos son realmente reconocedores y no analizadores sintácticos puesto que no construyen una representación de los árboles derivados o *bosque compartido de análisis sintáctico*, que debe satisfacer las siguientes propiedades [34]:

1. Debe contener en forma compacta todos los árboles de análisis. En particular, su tamaño debe ser de orden polinomial con respecto a la longitud de la cadena de entrada.

2. La recuperación de cada árbol de análisis individual debe poder realizarse en tiempo lineal con respecto al tamaño del bosque de análisis.

Billot y Lang [30] definen el bosque compartido correspondiente a una gramática independiente del contexto $\mathcal{G} = (V_T, V_N, P, S)$ y a una cadena de entrada $a_1 \dots a_n$ como una gramática independiente del contexto en la cual los no-terminales son de la forma $\langle A, i, j \rangle$, donde $A \in V_N$ y $i, j \in 0..n$, y las producciones presentan al forma

$$\langle A_0, j_0, j_m \rangle \to w_0 \ \langle A_1, j_0, j_1 \rangle \ w_1 \ \langle A_2, j_1, j_2 \rangle \ \dots \ w_{m-1} \ \langle A_m, j_{m-1}, j_m \rangle \ w_m$$

donde $A_0 \to w_0 A_1 w_1 A_2 \dots w_{m-1} A_m w_m \in P$ y $w_i \in V_T^*$, mediante las cuales se expresa que A_0 reconoce la subcadena $a_{j_0+1} \dots a_{j_m}$ mediante la aplicación de la producción $A_0 \to w_0 A_1 w_1 A_2 \dots w_{m-1} A_m w_m$ y donde cada A_i reconoce una subcadena $a_{j_{i-1}+1} \dots a_{j_i}$.

Podemos extender el concepto de bosque de análisis compartido para definir el concepto de *bosque de LIG* (*LIGed forest*) [215, 31]. Dado el bosque compartido de la gramática independiente del contexto que constituye el esqueleto de una LIG, siempre que una producción LIG $A_0[\circ\circ\gamma] \to A_1[\,] \dots A_d[\circ\circ\gamma'] \dots A_m[\,]$ participe en una derivación se añadirá la producción

$$\langle A_0, j_0, j_m \rangle [\circ\circ\gamma] \to \langle A_1, j_0, j_1 \rangle \ \dots \ \langle A_d, j_{d-1}, j_d \rangle [\circ\circ\gamma'] \ \dots \ \langle A_m, j_{m-1}, j_m \rangle$$

al bosque de LIG para indicar que A_0 reconoce la subcadena $a_{j_0+1} \dots a_{j_m}$ mediante la aplicación de la producción $A_0[\circ\circ\gamma] \to A_1 \dots A_d[\circ\circ\gamma'] \dots A_m$ donde cada A_i reconoce una subcadena $a_{j_{i-1}+1} \dots a_{j_i}$ y la pila de índices es transmitida de A_0 a A_d pero reemplazando la cima γ por γ'. Desgraciadamente, la gramática lineal de índices construida de esta manera no satisface una de las propiedades que debe poseer un bosque de análisis compartido, puesto que no es posible extraer cada uno de los árboles de análisis individuales en tiempo lineal con respecto al tamaño del bosque. Vijay-Shanker y Weir [215] intentan resolver este problema mediante la definición de un autómata finito no-determinista que se encarga de comprobar si un símbolo $\langle A, i, j \rangle [\alpha]$ del bosque deriva una cadena de terminales. Nederhof define en [126] un autómata finito similar al de Vijay-Shanker y Weir junto con un método de transformación gramatical, de tal modo que la gramática lineal de índices obtenida garantiza que todos los símbolos LIG que puedan aparecer en una derivación derivan a su vez una cadena de terminales.

Los enfoques anteriores siguen el criterio expuesto por Lang en [108] de considerar que la salida de un analizador sintáctico debe ser la intersección entre la gramática de entrada y la cadena a analizar. Boullier presenta en [32] un enfoque alternativo según el cual el bosque de análisis compartido para una LIG $\mathcal{L} = (V_T, V_N, V_I, P, S)$ y una cadena de entrada w se define mediante una *gramática de derivación lineal*, una gramática independiente del contexto que reconoce el lenguaje definido por las secuencias de producciones LIG de $\mathcal{L}$ que podrían ser utilizadas para derivar w. Con anterioridad a la construcción de la gramática de derivación lineal es preciso obtener el cierre transitivo de una serie de relaciones sobre $V_N \times V_N$.

Con el fin de evitar el uso de estructuras de datos adicionales, tales como máquinas de estado finito o relaciones precalculadas, nos hemos inspirado en la utilización de gramáticas independientes del contexto como representación del bosque de análisis compartido para gramáticas de adjunción de árboles [215] para tratar de capturar la independencia al contexto de la aplicación de producciones en el caso de LIG. Dada una gramática lineal de índices $\mathcal{L} = (V_T, V_N, V_I, P, S)$ y una cadena de entrada $w = a_1 \dots a_n$, el bosque compartido para $\mathcal{L}$ y w es una gramática independiente del contexto $\mathcal{L}^w = (V_T, V_N^w, P^w, S^w)$ en la que los elementos en V_N^w tienen la forma $\langle A, \gamma, i, j, B, p, q \rangle$, donde $A, B \in V_N$, $\gamma \in V_I$ y $i, j, p, q \in 0 \dots n$. El axioma S^w es el no-terminal $\langle S, -, 0, n, -, -, -, \rangle$. Las producciones en P^w se construyen tal y como se muestra a continuación:

Caso 1a: si $A[\,] \Rightarrow a_j$ entonces se añade la producción

$$\langle A, -, j-1, j, -, -, - \rangle \to a_j$$

Caso 1b: Si $A[\,] \Rightarrow \epsilon$ entonces se añade la producción

$$\langle A, -, j, j, -, -, - \rangle \to \epsilon$$

Caso 2a: Si $A[\circ\circ\gamma] \to B[\,]\ C[\circ\circ] \in P$, $B[\,] \overset{*}{\Rightarrow} a_{i+1} \dots a_k$ y $C[\alpha\eta] \overset{*}{\Rightarrow} a_{k+1} \dots a_p\, D[\alpha]\, a_{q+1} \dots a_j$ entonces se añade la producción

$$\langle A, \gamma, i, j, C, k, j \rangle \to \langle B, -, i, k, -, -, - \rangle\ \langle C, \eta, k, j, D, p, q \rangle$$

Caso 2b: Si $A[\circ\circ\gamma] \to B[\circ\circ]\ C[\,] \in P$, $B[\alpha\eta] \overset{*}{\Rightarrow} a_{i+1} \dots a_p\, D[\alpha]\, a_{q+1} \dots a_k$ y $C[\,] \overset{*}{\Rightarrow} a_{k+1} \dots a_j$ entonces se añade la producción

$$\langle A, \gamma, i, j, B, i, k \rangle \to \langle B, \eta, i, k, D, p, q \rangle\ \langle C, -, k, j, -, -, - \rangle$$

Caso 3a: Si $A[\circ\circ] \to B[\,]\ C[\circ\circ] \in P$, $B[\,] \overset{*}{\Rightarrow} a_{i+1} \dots a_k$ y $C[\alpha\eta] \overset{*}{\Rightarrow} a_{k+1} \dots a_p\, D[\alpha]\, a_{q+1} \dots a_j$ entonces se añade la producción

$$\langle A, \eta, i, j, D, p, q \rangle \to \langle B, -, i, k, -, -, - \rangle\ \langle C, \eta, k, j, D, p, q \rangle$$

Caso 3b: Si $A[\circ\circ] \to B[\circ\circ]\ C[\,] \in P$, $B[\alpha\eta] \overset{*}{\Rightarrow} a_{i+1} \dots a_p\, D[\alpha]\, a_{q+1} \dots a_k$ y $C[\,] \overset{*}{\Rightarrow} a_{k+1} \dots a_j$ entonces se añade la producción

$$\langle A, \eta, i, j, D, p, q \rangle \to \langle B, \eta, i, k, D, p, q \rangle\ \langle C, -, k, j, -, -, - \rangle$$

Caso 4a: Si $A[\circ\circ] \to B[\,]\ C[\circ\circ\gamma] \in P$, $B[\,] \overset{*}{\Rightarrow} a_{i+1} \dots a_k$, $C[\alpha\eta\gamma] \overset{*}{\Rightarrow} a_{k+1} \dots a_p\, D[\alpha\eta]\, a_{q+1} \dots a_j$ y $D[\alpha\eta] \overset{*}{\Rightarrow} a_{p+1} \dots a_r\, E[\alpha]\, a_{s+1} \dots a_q$ entonces se añade la producción

$$\langle A, \eta, i, j, E, r, s \rangle \to \langle B, -, i, k, -, -, - \rangle\ \langle C, \gamma, k, j, D, p, q \rangle\ \langle D, \eta, p, q, E, r, s \rangle$$

donde las derivaciones que comienzan en $\langle D, \eta, p, q, E, r, s \rangle$ posibilitan la recuperación de la parte restante de la pila de índices perteneciente a A.

Caso 4b: Si $A[\circ\circ] \to B[\circ\circ\gamma]\ C[\,] \in P$, $B[\alpha\eta\gamma] \overset{*}{\Rightarrow} a_{i+1} \dots a_p\, D[\alpha\eta]\, a_{q+1} \dots a_k$, $C[\,] \overset{*}{\Rightarrow} a_{k+1} \dots a_j$ y $D[\alpha\eta] \overset{*}{\Rightarrow} a_{p+1} \dots a_r\, E[\alpha]\, a_{s+1} \dots a_q$ entonces se añade la producción

$$\langle A, \eta, i, j, E, r, s \rangle \to \langle B, \gamma, i, k, D, p, q \rangle\ \langle C, -, k, j, -, -, - \rangle\ \langle D, \eta, p, q, E, r, s \rangle$$

donde las derivaciones que comienzan en $\langle D, \eta, p, q, E, r, s \rangle$ posibilitan la recuperación de la parte restante de la pila de índices perteneciente a A.

Caso 5: Si $A[\circ\circ\gamma] \to B[\circ\circ] \in P$ y $B[\alpha\gamma] \overset{*}{\Rightarrow} a_{i+1} \dots a_p\, D[\alpha]\, a_{q+1} \dots a_j$ entonces se añade la producción

$$\langle A, \gamma, i, j, B, i, j \rangle \to \langle B, \eta, i, j, D, p, q \rangle$$

Caso 6: Si $A[\circ\circ] \to B[\circ\circ] \in P$ y $B[\alpha\gamma] \overset{*}{\Rightarrow} a_{i+1} \dots a_p\, D[\alpha]\, a_{q+1} \dots a_j$ entonces se añade la producción

$$\langle A, \gamma, i, j, D, p, q \rangle \to \langle B, \gamma, i, j, D, p, q \rangle$$

Caso 7: Si $A[\circ\circ] \rightarrow B[\circ\circ\gamma] \in P$, $B[\alpha\eta\gamma] \overset{*}{\Rightarrow} a_{i+1} \ldots a_p \, D[\alpha\eta] \, a_{q+1} \ldots a_j$ y $D[\alpha\eta] \overset{*}{\Rightarrow} a_{p+1} \ldots a_r \, E[\alpha] \, a_{s+1} \ldots a_q$ entonces se añade la producción

$$\langle A, \eta, i, j, E, r, s\rangle \rightarrow \langle B, \gamma, i, j, D, p, q\rangle \; \langle D, \eta, p, q, E, r, s\rangle$$

donde las derivaciones que comienzan en $\langle D, \eta, p, q, E, r, s\rangle$ posibilitan la recuperación de la parte restante de la pila de índices perteneciente a A.

Es importante reseñar que estamos asumiendo gramáticas lineales de índices cuyas producciones tienen a lo sumo dos elementos en el lado derecho. Este hecho no representa un problema puesto que cualquier producción LIG $A_0[\circ\circ\gamma] \rightarrow A_1 \ldots a_d[\circ\circ\gamma'] \ldots A_m[\,]$ puede ser implícitamente binarizada en un conjunto de producciones LIG

$$\begin{gathered}
\nabla_0[\,] \rightarrow \epsilon \\
\nabla_1[\circ\circ] \rightarrow \nabla_0[\circ\circ] \; A_1[\,] \\
\vdots \\
\nabla_{d-1}[\circ\circ] \rightarrow \nabla_{d-2}[\circ\circ] \; A_{d-1}[\,] \\
\nabla_d[\circ\circ\gamma] \rightarrow \nabla_{d-1}[\,] \; A_d[\circ\circ\gamma'] \\
\nabla_{d+1}[\circ\circ] \rightarrow \nabla_d[\circ\circ] \; A_{d+1}[\,] \\
\vdots \\
\nabla_m[\circ\circ] \rightarrow \nabla_{m-1}[\circ\circ] \; A_m[\,] \\
A_0[\circ\circ] \rightarrow \nabla_m[\circ\circ]
\end{gathered}$$

donde los ∇_i son símbolos no-terminales nuevos que representan el reconocimiento parcial de la producción original. De hecho, un símbolo ∇_i es equivalente a una producción con punto que tenga el punto situado justo antes del no-terminal A_{i+1} ó con el punto al final del lado derecho en el caso de ∇_m.

Existe una correspondencia directa entre el caso 1 y los pasos de tipo Scan del esquema de análisis sintáctico **CYK**, y entre los casos 2, 3 y 4 y el resto de pasos del mismo esquema. Existe también una correspondencia similar para los demás esquemas de análisis, donde el caso 1 se corresponde con los pasos de tipo Scan y los demás casos se corresponden con los diferentes pasos de compleción.

Es interesante señalar que el conjunto de no-terminales es un subconjunto del conjunto de ítems de los esquemas de análisis **CYK**, **buE** y **E**. El caso del esquema de análisis **Earley** es ligeramente diferente, puesto que cada no-terminal $\langle A, \gamma, i, j, B, p, q\rangle$ representa la clase de ítems $[E, h \mid A, \gamma, i, j \mid D, p, q]$ para cualquier valor de E y de h.

Al igual que ocurre cuando se utilizan gramáticas independientes del contexto para representar el bosque de análisis compartido en el caso de TAG [215], las derivaciones de $\mathcal{L}^w$ codifican las derivaciones de la cadena de entrada w según $\mathcal{L}$ pero el conjunto específico de cadenas terminales generadas por $\mathcal{L}^w$ no es relevante. Lo importante es que $L(\mathcal{L}^w)$, el lenguaje generado por $\mathcal{L}^w$, es no vacío si y sólo si w pertenece a $L(\mathcal{L})$, el lenguaje generado por $\mathcal{L}$. Si podamos $\mathcal{L}^w$ reteniendo los símbolos útiles de tal modo que podamos garantizar que cada no-terminal genera una cadena terminal, las derivaciones de w en la LIG original pueden ser obtenidas a partir de las derivaciones de w en $\mathcal{L}^w$.

El número de posibles producciones en $\mathcal{L}^w$ es $\mathcal{O}(n^7)$. Esta complejidad puede ser reducida a $\mathcal{O}(n^6)$ mediante la transformación de las producciones que presentan la forma $A[\circ\circ] \rightarrow$

$B[\,]\ C[\circ\circ\gamma]$ en dos producciones

$$A[\circ\circ] \rightarrow B[\,]\ C^\gamma[\circ\circ]$$

$$C^\gamma[\circ\circ] \rightarrow C[\circ\circ\gamma]$$

donde C^γ es un nuevo no-terminal. De modo análogo, las producciones $A[\circ\circ] \rightarrow B[\circ\circ\gamma]\ C[\,]$ se pueden transformar en

$$A[\circ\circ] \rightarrow B^\gamma[\circ\circ]\ C[\,]$$

$$B^\gamma[\circ\circ] \rightarrow B[\circ\circ\gamma]$$

donde B^γ es un nuevo no-terminal.

4.6 Comparación entre los algoritmos de análisis sintáctico para LIG y TAG

Podemos apreciar que existe una gran similitud entre los esquemas de análisis **CYK**, **buE** y **E** definidos para gramáticas lineales de índices en este capítulo y los esquemas homónimos definidos para gramáticas de adjunción de árboles en el capítulo 3, así como entre el esquema **Earley** para LIG y el esquema **Nederhof** para TAG.

En lo que respecta a los ítems tenemos que en ambos casos se almacenan las mismas posiciones de la cadena de entrada y una producción con punto. Las diferencias surgen en aquella información referida al esqueleto independiente del contexto que es preciso guardar.

En las gramáticas de adjunción todo nodo forma parte de un árbol elemental y por lo tanto conocemos directamente el nodo raíz y el nodo pie del árbol al que pertenece. La operación de adjunción de un árbol β en un nodo N^γ se traduce en que el análisis continúa en la raíz del árbol β, propagando a través de la espina la pila de adjunciones no terminadas, la más reciente de las cuales es la que ha sido realizada sobre N^γ. El árbol γ será retomado en dicho nodo cuando se alcance el nodo pie de β. En consecuencia, un ítem representando el estado del proceso de análisis en un nodo arbitrario M^β tendría la forma siguiente:

$$[M^\beta \rightarrow \delta \bullet \nu, N^\gamma, i, j \mid \mathbf{F}^\beta, p, q]$$

Vemos que la información proporcionada por la aparición de $\mathbf{F}^\beta$ es redundante y puede ser eliminada. Aunque aparentemente la información proporcionada N^γ es necesaria, podemos ver que es redundante si consideramos que en lugar de N^γ podría aparecer cualquier otro nodo de cualquier otro árbol elemental que admita la adjunción de β. Si eliminamos N^γ de los ítems ganaremos en compartición y reduciremos la complejidad con respecto al tamaño de la gramática, que consideramos constante. Efectivamente, la desaparición de dicho elemento se suple por la adición de condiciones de aplicación que comprueban que se cumplan las restricciones de adjunción.

Con respecto a los ítems utilizados en aquellos algoritmos de análisis sintáctico de TAG que preservan la propiedad del prefijo válido, tras las consideraciones anteriores tendrían que tener la forma

$$[\mathbf{R}^\beta, h \mid M^\beta \rightarrow \delta \bullet \nu, i, j \mid p, q]$$

donde observamos que $\mathbf{R}^\beta$ es redundante, puesto que el nodo padre del árbol β es siempre conocido.

Con respecto a los pasos deductivos, la tabla 4.2 muestra la equivalencia entre los pasos utilizados por los algoritmos para el análisis de LIG y aquellos utilizados para el análisis de TAG. Vemos que la mayor diferencia radica en que en el caso de LIG debemos diferenciar entre aquellas predicciones y compleciones del esqueleto independiente del contexto que se realizan

TAG	**LIG**
Init	Init
Scan	Scan
Pred (fuera de la espina)	Pred[ ]
Comp (fuera de la espina)	Comp[ ]
Pred (en la espina)	Pred[$\circ\circ$][$\circ\circ$]
Comp (en la espina)	Comp[$\circ\circ$][$\circ\circ$]
AdjPred	Pred[$\circ\circ$][$\circ\circ\gamma$]
FootPred	Pred[$\circ\circ\gamma$][$\circ\circ$]
FootComp	Comp[$\circ\circ\gamma$][$\circ\circ$]
AdjComp	Comp[$\circ\circ$][$\circ\circ\gamma$]

Tabla 4.2: Correspondencia de los pasos deductivos para TAG y LIG

fuera de la espina y aquellas que se realizan en la espina. Esto se debe a que en el caso de LIG la pila de índices debe propagarse explícitamente a través de la espina, mientras que en el caso de TAG la pila de adjunciones es propagada implícitamente a través de la espina y por eso no es necesario diferenciar ambos casos.

Como caso especialmente interesante tenemos el algoritmo para el análisis sintáctico de LIG definido por el esquema **Earley** y el algoritmo para el análisis sintáctico de TAG definido por el esquema **Nederhof**. En ambos casos se implementa una estrategia que aplica predicción tanto sobre el esqueleto independiente del contexto como sobre la pila de índices o adjunciones. En ambos casos se obtiene inicialmente un algoritmo con complejidad superior a $\mathcal{O}(n^6)$ pero se reduce a esta última mediante la división del paso más complejo en una serie de pasos que se combinan entre sí. El interés de estos algoritmos viene determinado porque si bien **Nederhof** representa el algoritmo para el análisis de TAG tal y como fue descrito por Nederhof [125], la estrategia adoptada en el esquema **Earley** para LIG muestra un carácter más general, aplicable no sólo al caso de LIG sino también a la tabulación de diversos modelos e autómatas para esta clase de lenguajes [53, 14] y a la tabulación de algoritmos para TAG. Aplicada a este caso, el conjunto de pasos $\mathcal{D}^{\text{AdjComp}}_{\text{Nederhof}}$ se descompondría en:

$$\mathcal{D}^{\text{AdjComp}^0}_{\text{Nederhof}'} = \mathcal{D}^{\text{AdjComp}^0}_{\text{Nederhof}} = \frac{\begin{array}{c}[j, \top \to \mathbf{R}^\beta \bullet, j, m \mid k, l],\\ [h, M^\gamma \to \upsilon \bullet, k, l \mid p, q],\end{array}}{[[M^\gamma \to \upsilon \bullet, j, m \mid p, q]]} \quad \beta \in \text{adj}(M^\gamma)$$

$$\mathcal{D}^{\text{AdjComp}^1}_{\text{Nederhof}'} = \frac{\begin{array}{c}[[M^\gamma \to \upsilon \bullet, j, m \mid p, q]],\\ [h, M^\gamma \to \upsilon \bullet, k, l \mid p, q],\\ [h, N^\gamma \to \delta \bullet M^\gamma \nu, i, j \mid p', q']\end{array}}{[h, N^\gamma \to \delta M^\gamma \bullet \nu, i, m \mid p \cup p', q \cup q']}$$

a partir de los cuales podemos definir un nuevo esquema de análisis sintáctico $\mathbb{P}_{\text{Nederhof}'}$ para TAG.

Esquema de análisis sintáctico 4.6 El sistema de análisis $\mathbb{P}_{\text{Nederhof}'}$ que se corresponde con una variación el algoritmo de análisis sintáctico de tipo Earley para TAG propuesto por Nederhof, dada una gramática de adjunción de árboles $\mathcal{T}$ y una cadena de entrada $A_1 \dots a_n$ se define como sigue:

$$\mathcal{I}_{\text{Nederhof}'} = \mathcal{I}_{\text{Nederhof}}$$

$$\mathcal{D}^{\text{Init}}_{\text{Nederhof}'} = \mathcal{D}^{\text{Init}}_{\text{Nederhof}}$$

$$\mathcal{D}^{\text{Scan}}_{\text{Nederhof}'} = \mathcal{D}^{\text{Scan}}_{\text{Nederhof}}$$

$$\mathcal{D}^{\text{Pred}}_{\text{Nederhof}'} = \mathcal{D}^{\text{Pred}}_{\text{Nederhof}}$$

$$\mathcal{D}^{\text{Comp}}_{\text{Nederhof}'} = \mathcal{D}^{\text{Comp}}_{\text{Nederhof}}$$

$$\mathcal{D}^{\text{AdjPred}}_{\text{Nederhof}'} = \mathcal{D}^{\text{AdjPred}}_{\text{Nederhof}}$$

$$\mathcal{D}^{\text{FootPred}}_{\text{Nederhof}'} = \mathcal{D}^{\text{FootPred}}_{\text{Nederhof}}$$

$$\mathcal{D}^{\text{FootComp}}_{\text{Nederhof}'} = \mathcal{D}^{\text{FootComp}}_{\text{Nederhof}}$$

$$\mathcal{D}^{\text{AdjComp}^0}_{\text{Nederhof}'} = \mathcal{D}^{\text{AdjComp}^0}_{\text{Nederhof}}$$

$$\mathcal{D}^{\text{AdjComp}^1}_{\text{Nederhof}'} = \frac{\begin{array}{l}[[M^\gamma \to \upsilon\bullet, j, m \mid p, q]],\\ [h, M^\gamma \to \upsilon\bullet, k, l \mid p, q],\\ [h, N^\gamma \to \delta \bullet M^\gamma \nu, i, j \mid p', q']\end{array}}{[h, N^\gamma \to \delta M^\gamma \bullet \nu, i, m \mid p \cup p', q \cup q']} \quad \beta \in \text{adj}(M^\gamma)$$

$$\begin{aligned}\mathcal{D}_{\text{Nederhof}'} = \;& \mathcal{D}^{\text{Init}}_{\text{Nederhof}'} \cup \mathcal{D}^{\text{Scan}}_{\text{Nederhof}'} \cup \mathcal{D}^{\text{Pred}}_{\text{Nederhof}'} \cup \mathcal{D}^{\text{Comp}}_{\text{Nederhof}'} \cup \mathcal{D}^{\text{AdjPred}}_{\text{Nederhof}'} \cup \mathcal{D}^{\text{FootPred}}_{\text{Nederhof}'}\\ & \cup \mathcal{D}^{\text{FootComp}}_{\text{Nederhof}'} \cup \mathcal{D}^{\text{AdjComp}^0}_{\text{Nederhof}'} \cup \mathcal{D}^{\text{AdjComp}^1}_{\text{Nederhof}'}\end{aligned}$$

$$\mathcal{F}_{\text{Nederhof}'} = \mathcal{F}_{\text{Nederhof}}$$

§

Proposición 4.7 $\mathbf{E} \stackrel{\text{df}}{\Longrightarrow} \mathbf{Ear}' \stackrel{\text{ir}}{\Longrightarrow} \mathbf{Earley}' \stackrel{\text{sr}}{\Longrightarrow} \mathbf{Nederhof}'$.

4.7 Otros algoritmos de análisis sintáctico para LIG

A diferencia del caso de las gramáticas de adjunción de árboles, donde encontramos un elevado número de algoritmos diferentes, en el caso de las gramáticas lineales de índices prácticamente todos los algoritmos de análisis sintáctico descritos en la literatura son variaciones de los algoritmos CYK, con la salvedad de los algoritmos que se describen a continuación.

4.7.1 Algoritmo bidireccional

Schneider define en [181, 182] una extensión del algoritmo de tipo *head-corner* para gramáticas independientes del contexto presentado por Sikkel en [189]. El resultado es un algoritmo bidireccional que procede ascendentemente guiado por los hijos dependientes de las producciones de la gramática. Los resultados intermedios del proceso de análisis se almacenan en ítems con producciones anotadas por dos puntos, de la forma

$$[\boldsymbol{A} \to \Upsilon_1 \bullet \Upsilon_2 \bullet \Upsilon_3, \gamma, i, j \mid B, p, q]$$

que representan los siguientes tipos de derivaciones:

- $\Upsilon_2 \overset{*}{\Rightarrow} \Upsilon_{2'}\ E[\gamma]\ \Upsilon_{2''} \overset{*}{\Rightarrow} a_{i+1} \dots a_p\ B[\,]\ a_{q+1} \dots a_j$ si y sólo si $(B,p,q) \neq (-,-,-)$ y donde $B[\,]$ es un descendiente dependiente de $E[\gamma]$.
- $\Upsilon_2 \overset{*}{\Rightarrow} a_{i+1} \dots a_j$ si y sólo si $\gamma = -$ y $(B,p,q) = (-,-,-)$.

La cadena de entrada $a_1 \dots a_n$ ha sido reconocida cuando se obtiene un ítem de la forma $[\boldsymbol{S} \to \bullet\Upsilon\bullet\ , -, 0, n \mid -,-,-]$. A continuación mostramos el esquema de análisis correspondiente a este algoritmo.

Esquema de análisis sintáctico 4.7 El sistema de análisis $\mathbb{P}_{\mathrm{HC}}$ que se corresponde con el algoritmo de análisis bidireccional ascendente para una gramática lineal de índices $\mathcal{L}$ y una cadena de entrada $a_1 \dots a_n$ se define como sigue:

$$\mathcal{I}_{\mathrm{HC}} = \left\{ \begin{array}{ll} [\boldsymbol{A} \to \Upsilon_1 \bullet \Upsilon_2 \bullet \Upsilon_3, \gamma, i, j \mid B, p, q] \mid & \boldsymbol{A} \to \Upsilon_1\Upsilon_2\Upsilon_3 \in P,\ B \in V_N,\ \gamma \in V_I, \\ & 0 \leq i \leq j\ ,\ \ (p,q) \leq (i,j) \end{array} \right\}$$

$$\mathcal{D}_{\mathrm{HC}}^{\mathrm{Scan}} = \frac{[a, j, j+1]}{[A[\,] \to \bullet a \bullet, -, j, j+1 \mid -,-,-]}$$

$$\mathcal{D}_{\mathrm{HC}}^{\mathrm{LComp}[\,]} = \frac{\begin{array}{c}[\boldsymbol{A} \to \Upsilon_1\ B[\,] \bullet \Upsilon_2 \bullet \Upsilon_3, \gamma, k, j \mid C, p, q], \\ [\boldsymbol{B} \to \bullet\Upsilon_4\bullet, -, i, k \mid -,-,-]\end{array}}{[\boldsymbol{A} \to \Upsilon_1 \bullet B[\,]\ \Upsilon_2 \bullet \Upsilon_3, \gamma, i, j \mid C, p, q]}$$

$$\mathcal{D}_{\mathrm{HC}}^{\mathrm{RComp}[\,]} = \frac{\begin{array}{c}[\boldsymbol{A} \to \Upsilon_1 \bullet \Upsilon_2 \bullet B[\,]\ \Upsilon_3, \gamma, i, k \mid C, p, q], \\ [\boldsymbol{B} \to \bullet\Upsilon_4\bullet, -, k, j \mid -,-,-]\end{array}}{[\boldsymbol{A} \to \Upsilon_1 \bullet \Upsilon_2\ B[\,] \bullet \Upsilon_3, \gamma, i, j \mid C, p, q]}$$

$$\mathcal{D}_{\mathrm{HC}}^{\mathrm{HC}[\circ\circ\gamma][\circ\circ]} = \frac{[\boldsymbol{B} \to \bullet\Upsilon_3\bullet, \eta, i, j \mid C, p, q]}{[A[\circ\circ\gamma] \to \Upsilon_1 \bullet B[\circ\circ] \bullet \Upsilon_2, \gamma, i, j \mid B, i, j]}$$

$$\mathcal{D}_{\mathrm{HC}}^{\mathrm{HC}[\circ\circ][\circ\circ]} = \frac{[\boldsymbol{B} \to \bullet\Upsilon_3\bullet, \eta, i, j \mid C, p, q]}{[A[\circ\circ] \to \Upsilon_1 \bullet B[\circ\circ] \bullet \Upsilon_2, \eta, i, j \mid C, p, q]}$$

$$\mathcal{D}_{\mathrm{HC}}^{\mathrm{HC}[\circ\circ][\circ\circ\gamma]} = \frac{\begin{array}{c}[\boldsymbol{B} \to \bullet\Upsilon_3\bullet, \gamma, i, j \mid C, p, q] \\ [\boldsymbol{C} \to \bullet\Upsilon_4\bullet, \eta, p, q \mid D, r, s]\end{array}}{[A[\circ\circ] \to \Upsilon_1 \bullet B[\circ\circ\gamma] \bullet \Upsilon_2, \eta, i, j \mid D, r, s]}$$

$$\mathcal{D}_{\mathrm{HC}} = \mathcal{D}_{\mathrm{HC}}^{\mathrm{Scan}} \cup \mathcal{D}_{\mathrm{HC}}^{\mathrm{LComp}[\,]} \cup \mathcal{D}_{\mathrm{HC}}^{\mathrm{RComp}[\,]} \cup \mathcal{D}_{\mathrm{HC}}^{\mathrm{HC}[\circ\circ\gamma][\circ\circ]} \cup \mathcal{D}_{\mathrm{HC}}^{\mathrm{HC}[\circ\circ][\circ\circ]} \cup \mathcal{D}_{\mathrm{HC}}^{\mathrm{HC}[\circ\circ][\circ\circ\gamma]}$$

$$\mathcal{F}_{\mathrm{HC}} = \left\{\ [\boldsymbol{S} \to \bullet\Upsilon\bullet\ , -, 0, n \mid -,-,-]\ \right\}$$

§

4.7.2 Algoritmo de reconocimiento de Boullier

Boullier presenta en [31] un reconocedor para gramáticas lineales de índices que divide el proceso de análisis en las dos fases siguientes:

1. Análisis de la gramática independiente del contexto que constituye el esqueleto de la gramática lineal de índices original con el fin de obtener un bosque compartido de todos los posibles análisis.

2. Comprobación de que las condiciones impuestas por la gramática lineal de índices se cumplen a través de las espinas definidas en el bosque de análisis resultante de la primera fase.

La segunda fase constituye la parte fundamental del algoritmo y está basada en el hecho de que si consideramos la evolución individual de cada espina vemos que todo índice que es apilado en un punto debe ser sacado de la pila en otro punto de la espina y viceversa, cada vez que un índice es sacado de una pila en un punto de la espina tenemos la seguridad de que ha sido apilado en algún punto anterior de la misma. Por lo tanto, para verificar las condiciones impuestas por una LIG sobre las espinas no es necesario reconstruir las pilas de índices sino verificar que se cumplen los pares apilar/sacar. En el caso del reconocimiento la tarea se ve simplificada por el hecho de que no nos interesa indicar todas las posibles espinas entre dos elementos LIG sino simplemente verificar la existencia de una de tales espinas.

Para ello utilizaremos las relaciones $\prec\!\!\succ$, $\overset{\gamma}{\prec}$ y $\overset{\gamma}{\succ}$ que se definen en la tabla 4.3, donde o_1 se refiere al elemento LIG constituido por A y su pila de índices asociada y o_2 se refiere a B y su pila de índices. Estas relaciones indican, para cada producción de la gramática, la operación que se debe aplicar a la pila asociada al elemento del lado izquierdo para obtener la pila asociada al hijo dependiente, de tal modo que $\prec\!\!\succ$ indica que no se hacen cambios, $\overset{\gamma}{\prec}$ que se apila el índice γ y $\overset{\gamma}{\succ}$ que se saca de la pila el índice γ.

$\prec\!\!\succ$	$=$	$\{(o_1, o_2) \mid A[\circ\circ] \rightarrow \Upsilon_1\ B[\circ\circ]\ \Upsilon_2 \in P\}$
$\overset{\gamma}{\prec}$	$=$	$\{(o_1, o_2) \mid A[\circ\circ] \rightarrow \Upsilon_1\ B[\circ\circ\gamma]\ \Upsilon_2 \in P\}$
$\overset{\gamma}{\succ}$	$=$	$\{(o_1, o_2) \mid A[\circ\circ\gamma] \rightarrow \Upsilon_1\ B[\circ\circ]\ \Upsilon_2 \in P\}$

Tabla 4.3: Relaciones binarias definidas por el reconocedor de Boullier

El algoritmo de reconocimiento de Boullier utiliza esta tabla para inicializar las relaciones y aplica las reglas de composición descritas en la tabla 4.4 para obtener los valores finales de dichas relaciones. Utilizando estas relaciones podemos calcular la relación $\approx$ que calcula porciones válidas de una espina y que se define recursivamente como

$$\approx \;=\; \prec\!\!\succ \cup \overset{\gamma}{\prec}\overset{*}{\approx}\overset{\gamma}{\succ}$$

En este punto estamos en disposición de calcular la relación $\bowtie$ de espinas válidas que se define como

$$\bowtie \;=\; \{(o_1, o_2) \mid o_1 \overset{+}{\approx} o_2\}$$

tal que o_1 es de la forma $A[\alpha]$ y aparece en la parte izquierda de una producción y o_2 es de la forma $B[\alpha']$ y aparece en la parte derecha de una producción.

si	$o_1 \overset{\gamma}{\prec} o_2$	y	$o_2 \mathrel{\prec\!\!\!\succ} o_3$	entonces	$o_1 \overset{\gamma}{\prec} o_3$
si	$o_1 \overset{\gamma}{\prec} o_2$	y	$o_2 \overset{\gamma}{\succ} o_3$	entonces	$o_1 \mathrel{\prec\!\!\!\succ} o_3$
si	$o_1 \mathrel{\prec\!\!\!\succ} o_2$	y	$o_2 \overset{\gamma}{\prec} o_3$	entonces	$o_1 \overset{\gamma}{\prec} o_3$
si	$o_1 \mathrel{\prec\!\!\!\succ} o_2$	y	$o_2 \mathrel{\prec\!\!\!\succ} o_3$	entonces	$o_1 \mathrel{\prec\!\!\!\succ} o_3$
si	$o_1 \mathrel{\prec\!\!\!\succ} o_2$	y	$o_2 \overset{\gamma}{\succ} o_3$	entonces	$o_1 \overset{\gamma}{\succ} o_3$
si	$o_1 \overset{\gamma}{\succ} o_2$	y	$o_2 \mathrel{\prec\!\!\!\succ} o_3$	entonces	$o_1 \overset{\gamma}{\succ} o_3$

Tabla 4.4: Reglas de composición de las relaciones del reconocedor de Boullier

Esta última relación se utiliza para eliminar del bosque de análisis obtenido en la primera fase aquellos nodos que no cumplen las condiciones impuestas sobre las pilas de índices por la gramática. Si el nodo correspondiente al axioma de la gramática no es eliminado entonces podemos afirmar que la cadena de entrada pertenece al lenguaje definido por la gramática. El algoritmo trabaja con una complejidad temporal $\mathcal{O}(n^6)$, donde n es la longitud de la cadena de entrada, si se restringe la forma de las gramáticas de tal modo que la parte derecha de cada producción posea a lo sumo dos elementos.

4.7.3 Algoritmo de análisis sintáctico de Boullier

Boullier presenta en [32] una extensión del algoritmo precedente que lo convierte en un algoritmo de análisis sintáctico, ya que permite obtener todos los análisis posibles de una cadena de entrada de acuerdo con una gramática lineal de índices, manteniendo la complejidad $\mathcal{O}(n^6)$ para gramáticas con a lo sumo dos elementos en la parte derecha de las producciones.

El algoritmo se basa en la construcción de una *gramática de derivación lineal*, esencialmente una gramática independiente del contexto que define un lenguaje cuyas cadenas son las secuencias de producciones que constituyen derivaciones de una gramática lineal de índices para una cadena de entrada dada.

Para ello necesitamos definir las relaciones $\underset{1}{\prec\!\!\!\succ}$, $\underset{1}{\overset{\gamma}{\prec}}$ y $\underset{1}{\overset{\gamma}{\succ}}$ de la tabla 4.5, que como podemos observar son muy similares a las relaciones $\prec\!\!\!\succ$, $\overset{\gamma}{\prec}$ y $\overset{\gamma}{\succ}$. Adicionalmente, definimos la relación $\underset{+}{\prec\!\!\!\succ}$ que selecciona pares de no-terminales que forman parte de una espina y cuyas pilas de índices están vacías. A partir de estas relaciones definimos las relaciones $\underset{+}{\overset{\gamma}{\succ}}$ y $\approx$ como

$$\underset{+}{\overset{\gamma}{\succ}} \;=\; \underset{1}{\overset{\gamma}{\succ}} \cup \underset{+}{\prec\!\!\!\succ}\underset{1}{\overset{\gamma}{\succ}}$$

$$\approx \;=\; \bigcup_{\gamma \in V_I} \underset{1}{\overset{\gamma}{\prec}}\underset{+}{\overset{\gamma}{\succ}}$$

por lo que se cumple que

$$\underset{+}{\prec\!\!\!\succ} \;=\; \underset{1}{\prec\!\!\!\succ} \cup \approx \cup \underset{1}{\prec\!\!\!\succ}\underset{+}{\prec\!\!\!\succ} \cup \approx \underset{+}{\prec\!\!\!\succ}$$

Dada una gramática lineal de índices (V_N, V_T, V_I, P, S) definimos su gramática de derivación lineal como la gramática independiente del contexto $D = (V_N^D, V_T^D, P^D, S^D)$ donde:

- $V_N^D = \{[A] \mid A \in V_N\} \cup \{[A\rho B] \mid A, B \in V_N,\ \rho \in \mathcal{R}\}$, donde $\mathcal{R} = \{\underset{1}{\overset{\gamma}{\prec}}, \underset{1}{\prec\!\!\!\succ}, \underset{1}{\overset{\gamma}{\succ}}, \underset{+}{\prec\!\!\!\succ}, \approx, \underset{+}{\overset{\gamma}{\succ}}\}$

$$
\begin{array}{|lcl|}
\hline
\underset{1}{\diamond} & = & \{(A,B) \mid A[\circ\circ] \rightarrow \Upsilon_1 \, B[\circ\circ] \, \Upsilon_2 \in P\} \\
\overset{\gamma}{\underset{1}{\prec}} & = & \{(A,B) \mid A[\circ\circ] \rightarrow \Upsilon_1 \, B[\circ\circ\gamma] \, \Upsilon_2 \in P\} \\
\overset{\gamma}{\underset{1}{\succ}} & = & \{(A,B) \mid A[\circ\circ\gamma] \rightarrow \Upsilon_1 \, B[\circ\circ] \, \Upsilon_2 \in P\} \\
\underset{+}{\diamond} & = & \{(A,B) \mid A[\,] \stackrel{+}{\Rightarrow} \Upsilon_1 \, B[\,] \, \Upsilon_2 \text{ donde } B[\,] \text{ es el descendiente dependiente de } A[\,]\,\} \\
\hline
\end{array}
$$

Tabla 4.5: Relaciones binarias definidas por el analizador sintáctico de Boullier

- $V_T^D = P$
- $S^D = [S]$
- $$\begin{aligned} P^D \;=\; & \{[A] \rightarrow r \mid r : A[\,] \rightarrow a \in P\} \cup \\ & \{[A] \rightarrow r[A \underset{+}{\diamond} B] \mid r : B[\,] \rightarrow a \in P\} \cup \\ & \{[A \underset{+}{\diamond} C] \rightarrow [\Upsilon_1\Upsilon_2]r \mid r : A[\circ\circ] \rightarrow \Upsilon_1 \, C[\circ\circ] \, \Upsilon_2 \in P\} \cup \\ & \{[A \underset{+}{\diamond} C] \rightarrow [A \approx C]\} \cup \\ & \{[A \underset{+}{\diamond} C] \rightarrow [B \underset{+}{\diamond} C][\Upsilon_1\Upsilon_2]r \mid r : A[\circ\circ] \rightarrow \Upsilon \, B[\circ\circ] \, \Upsilon_2 \in P\} \cup \\ & \{[A \underset{+}{\diamond} C] \rightarrow [B \underset{+}{\diamond} C][A \approx B]\} \cup \\ & \{[A \approx C] \rightarrow [B \overset{\gamma}{\underset{+}{\succ}} C][\Upsilon_1\Upsilon_2]r \mid r : A[\circ\circ] \rightarrow \Upsilon_1 \, B[\circ\circ\gamma] \, \Upsilon_2 \in P\} \cup \\ & \{[A \overset{\gamma}{\underset{+}{\succ}} c] \rightarrow [\Upsilon_1\Upsilon_2]r \mid r : A[\circ\circ\gamma] \rightarrow \Upsilon_1 \, C[\circ\circ] \, \Upsilon_2 \in P\} \cup \\ & \{[A \overset{\gamma}{\underset{+}{\succ}} c] \rightarrow [\Upsilon_1\Upsilon_2]r[A \underset{+}{\diamond} B] \mid r : B[\circ\circ\gamma] \rightarrow \Upsilon_1 \, C[\circ\circ] \, \Upsilon_2 \in P\} \end{aligned}$$

  donde r identifica una producción y $[\Upsilon_1\Upsilon_2]$ denota, o bien al no-terminal $[X]$ cuando $\Upsilon_1\Upsilon_2 = X[\,]$, o bien a la cadena vacía ϵ cuando $\Upsilon_1\Upsilon_2 \in V_T^*$

Denominamos derivación lineal a una derivación a derechas en la cual primero se derivan los hijos no dependiente y por último el hijo dependiente. Las producciones P^D definen todas las posibles *derivaciones lineales* de la LIG original.

El algoritmo de análisis sintáctico de Boullier consta de los siguientes pasos:

1. Construcción de un bosque de análisis para la gramática independiente del contexto que constituye el esqueleto independiente del contexto de la LIG original.

2. Construcción del bosque de análisis para la LIG a partir del bosque de análisis obtenido para su esqueleto independiente del contexto. Dicho bosque de análisis constituye a su vez una LIG, de igual modo que el bosque de análisis de una CFG constituye una CFG.

3. Construcción de la gramática de derivación lineal a partir de la LIG que representa el bosque de análisis obtenido en el paso anterior.

Puesto que cada derivación de la gramática lineal de índices original es una cadena de la gramática de derivación lineal, la extracción de análisis individuales de una LIG ha sido reducida a la derivación de cadenas en una gramática independiente del contexto. En consecuencia, cada derivación se puede obtener en tiempo lineal.

Parte II

Modelos de autómata para los lenguajes de adjunción de árboles

Capítulo 5

Autómatas a pila

Los autómatas a pila son máquinas abstractas que reconocen exactamente la clase de los lenguajes independientes del contexto. En este capítulo introducimos este tipo de autómatas, que servirán de base a los modelos de autómata para lenguajes de adjunción de árboles que serán presentados en los capítulos siguientes.

5.1 Definición

Presentamos dos definiciones diferentes pero equivalentes de los autómatas a pila (*Push-Down Automata*, PDA) [85]. En primer lugar presentaremos la definición clásica, que considera un autómata a pila como una máquina abstracta que consta de tres componentes: una cadena de entrada, un control finito y una pila. Seguidamente presentaremos una definición más moderna en la cual se suprimen las referencias al control finito para centrase en el componente fundamental de este tipo de autómatas: la pila.

5.1.1 Definición con estados

En la definición clásica [85], los autómatas a pila son considerados tuplas $(Q, V_T, V_S, \delta, q_0, \$_0, Q_F)$, donde:

- Q es un conjunto finito de estados.
- V_T es un conjunto finito de símbolos terminales.
- V_S es un conjunto finito de símbolos de pila.
- $q_0 \in Q$ es el estado inicial.
- $\$_0 \in V_S$ es el símbolo inicial de la pila.
- $Q_F \subseteq Q$ es el conjunto de estados finales.
- δ es una relación de $Q \times (V_T \cup \{\epsilon\}) \times V_S$ en subconjuntos finitos de $Q \times V_S^*$ que define los movimientos o transiciones válidos del autómata. Una transición

  $$(q', \beta) \in \delta(q, a, Z)$$

  donde $q, q' \in Q$, $a \in V_T \cup \{\epsilon\}$, $Z \in V_S \cup \{\epsilon\}$, $\beta \in V_S^*$, se interpreta como sigue: si el autómata se encuentra en el estado q, el siguiente terminal en la cadena de entrada es a y el símbolo en la cima de la pila es Z, entonces puede pasar al estado q' y reemplazar la cima de la pila por β.

La *configuración* de un autómata a pila en un momento dado viene definida por el triple (q, α, w), donde q indica el estado en el que se encuentra el autómata, α el contenido de la pila y w la parte de la cadena de entrada que resta por leer. El cambio de una configuración a otra viene determinado por la aplicación de una transición, de tal modo que si $(q, \alpha Z, aw)$ es una configuración y $(q', \beta) \in \delta(q, a, Z)$, entonces el autómata pasará a la nueva configuración $(q', \alpha\beta, w)$, hecho que denotamos mediante $(q, \alpha, aw) \vdash (q', \alpha\beta, w)$. Denotamos por $\overset{*}{\vdash}$ el cierre reflexivo y transitivo de $\vdash$.

El *lenguaje aceptado por estado final* por un autómata a pila viene determinado por el conjunto de cadenas $w \in V_T^*$ tal que $(q_0, \$_0, w) \overset{*}{\vdash} (p, \alpha, \epsilon)$, donde $p \in Q_F$ y $\alpha \in V_S^*$.

El *lenguaje aceptado por pila vacía* por un autómata a pila viene determinado por el conjunto de cadenas $w \in V_T^*$ tal que $(q_0, \$_0, w) \overset{*}{\vdash} (p, \epsilon, \epsilon)$ para cualquier $p \in Q$.

Dado un autómata a pila que reconoce un determinado lenguaje por estado final, es posible construir otro autómata a pila que reconoce el mismo lenguaje por pila vacía y viceversa [85].

5.1.2 Definición sin estados

El control finito de un autómata a pila es un elemento prescindible, puesto que el estado de una configuración dada puede almacenarse en el elemento situado en la cima de la pila [24]. Como consecuencia obtenemos una definición alternativa equivalente [107, 24, 52], que juzgamos más simple y homogénea, según la cual un autómata a pila es una tupla $(V_T, V_S, \Theta, \$_0, \$_f)$, donde

- V_T es un conjunto finito de símbolos terminales.
- V_S es un conjunto finito de símbolos de pila.
- $\$_0 \in V_S$ es el símbolo inicial de la pila.
- $\$_f \in V_S$ es el símbolo final de la pila.
- Θ es un conjunto de transiciones, cada una de las cuales pertenece a uno de los tres tipos siguientes, donde $C, F, G \in V_S$, $\xi \in V_S^*$ y $a \in V_T \cup \{\epsilon\}$:

  SWAP: Transiciones de la forma $C \overset{a}{\longmapsto} F$ que reemplazan el elemento C de la cima de la pila por el elemento F mientras se lee a de la cadena de entrada. El resultado de aplicar una transición de este tipo a una pila ξC es una pila ξF.

  PUSH: Transiciones de la forma $C \overset{a}{\longmapsto} CF$ que apilan un nuevo elemento F en la pila mientras se lee a de la cadena de entrada. El resultado de aplicar una transición de este tipo a una pila ξC es una pila ξCF.

  POP: Transiciones de la forma $CF \overset{a}{\longmapsto} G$ que eliminan los dos elementos C y F de la cima de la pila y los sustituyen por G mientras se lee a de la cadena de entrada. El resultado de aplicar una transición de este tipo a una pila ξCF es una pila ξG, con lo cual el tamaño de la pila decrece en una unidad.

Según la nueva definición, la *configuración* de un autómata a pila en un momento dado viene determinada por el par (ξ, w), donde ξ es el contenido de la pila y w es la parte de la cadena de entrada que resta por leer. Una configuración (ξ, aw) deriva una configuración (ξ', w), hecho que denotamos mediante $(\xi, aw) \vdash (\xi', w)$, si y sólo si existe una transición que aplicada a ξ devuelve ξ' y consume a de la cadena de entrada. En caso de ser necesario identificar una derivación d concreta, utilizaremos la notación $\overset{*}{\vdash}_d$. Denotamos por $\overset{*}{\vdash}$ el cierre reflexivo y transitivo de $\vdash$.

Decimos que una cadena de entrada w es aceptada por un autómata a pila si $(\$_0, w) \overset{*}{\vdash} (\$_0\$_f, \epsilon)$. El *lenguaje aceptado* por un autómata a pila viene determinado por el conjunto de cadenas $w \in V_T^*$ tal que $(\$_0, w) \overset{*}{\vdash} (\$_0\$_f, \epsilon)$.

5.2 Esquemas de compilación

Un esquema de compilación es un conjunto de reglas que permite, a partir de una gramática independiente del contexto y de una estrategia de análisis sintáctico, construir un autómata a pila que describa los cálculos que se pueden realizar con dicha gramática utilizando la estrategia de análisis elegida.

Los esquemas de compilación que se van a mostrar se basan en el paradigma de llamada/retorno [55], utilizando para ello los seis tipos de reglas mostrados en la tabla 5.1. A toda regla [**CALL**] le corresponde una regla [**RET**] y viceversa. Las reglas [**INIT**], [**CALL**] y [**SEL**] definen las transiciones del autómata encargadas de la fase predictiva del algoritmo de análisis mientras que las reglas [**RET**] y [**PUB**] definen las transiciones encargadas de propagar la información en la fase ascendente. Por este motivo la fase descendente o predictiva de una estrategia de análisis cuando es implantada en un autómata a pila recibe el nombre de *fase de llamada*, mientras que la fase ascendente recibe el nombre de *fase de retorno*.

Regla	**Tarea**
[**INIT**]	inicia los cálculos a partir de la pila inicial.
[**CALL**]	requiere el análisis de un no-terminal de una producción.
[**SEL**]	selecciona una producción.
[**PUB**]	determina que una producción ha sido completamente analizada.
[**RET**]	continua el proceso de análisis después de terminar una producción.
[**SCAN**]	reconoce los terminales que componen la cadena de entrada.

Tabla 5.1: Reglas de los esquemas de compilación de gramáticas independientes del contexto

En primer lugar definiremos el esquema de compilación correspondiente a una estrategia genérica basada en el paradigma llamada/retorno, parametrizada con respecto a la información que se predice y propaga en las fases de llamada y de retorno, respectivamente. Utilizaremos la siguiente notación:

- $A_{r,s}$ para referirnos al elemento de la producción r que ocupa la posición s, de modo que para una producción r tenemos que $A_{r,0} \to A_{r,1} \ldots A_{r,n_r}$.
- $\nabla_{r,s}$ para indicar el reconocimiento parcial de una producción, notacionalmente equivalente a una producción con punto $A_{r,0} \to A_{r,1} \ldots A_{r,s} \bullet A_{r,s+1} \ldots A_{r,n_r}$.
- $\overrightarrow{A_{r,s}}$ para referirnos a la predicción de información con respecto a $A_{r,s}$.
- $\overleftarrow{A_{r,s}}$ para representar la información propagada ascendentemente con respecto a $A_{r,s}$.

Con el fin de simplificar al máximo la definición de los esquemas de compilación y sin pérdida de generalidad, supondremos que se cumplen las siguientes condiciones sobre la gramática independiente del contexto:

- El axioma sólo aparece en el lado izquierdo de la producción unitaria 0, que tiene la forma $S \to X$, con $X \in V_N \cup V_T$
- Los terminales sólo aparecen en producciones unitarias de la forma $A_{r,0} \to a$, donde $a \in V_T \cup \{\epsilon\}$.

Esquema de compilación 5.1 El esquema de compilación genérico de una gramática independiente del contexto en un autómata a pila queda definido por el conjunto de reglas mostrado a continuación y por los elementos inicial $\$_0$ y final $\overleftarrow{S}$. La primera columna indica el nombre de la regla, la segunda las transiciones generadas por la misma y la tercera las condiciones, generalmente referidas a la forma de las producciones, que se deben cumplir para que la regla sea aplicable.

[INIT]	$\$_0 \longmapsto \$_0 \ \nabla_{0,0}$	
[CALL]	$\nabla_{r,s} \longmapsto \nabla_{r,s} \ \overrightarrow{A_{r,s+1}}$	
[SEL]	$\overrightarrow{A_{r,0}} \longmapsto \nabla_{r,0}$	$r \neq 0$
[PUB]	$\nabla_{r,n_r} \longmapsto \overleftarrow{A_{r,0}}$	
[RET]	$\nabla_{r,s} \ \overleftarrow{A_{r,s+1}} \longmapsto \nabla_{r,s+1}$	
[SCAN]	$\overrightarrow{A_{r,0}} \overset{a}{\longmapsto} \overleftarrow{A_{r,0}}$	$A_{r,0} \to a$

§

A continuación presentamos tres versiones concretas del esquema de compilación genérico. La primera se corresponde con una estrategia de análisis descendente en la cual los no-terminales se predicen en la fase de llamada pero no se propagan en la fase ascendente. La segunda se corresponde con una estrategia mixta de tipo Earley [69] en la que los no-terminales se predicen en la fase de llamada y se propagan en la fase de retorno. La tercera y última se corresponde con una estrategia ascendente en la cual no hay ningún tipo de predicción en la fase de llamada mientras que en la fase de retorno se propagan los no-terminales analizados.

En la tabla 5.2 se muestran los valores que deben tomar los parámetros de predicción y propagación de información para transformar el esquema de compilación genérico en esquemas correspondientes a las tres estrategias de análisis citadas, donde $\Box$ representa un símbolo de pila nuevo. En el caso de estrategias Earley es necesario distinguir la llamada de un no-terminal $A_{r,s+1}$ de su retorno, para lo cual utilizamos los símbolos $\overline{A_{r,s+1}}$ y $\overline{\overline{A_{r,s+1}}}$, respectivamente.

5.2.1 Estrategia descendente

Esquema de compilación 5.2 El esquema de compilación descendente de una gramática independiente del contexto en un autómata a pila queda definido por el conjunto de reglas mostrado en la tabla 5.3 y por los elementos inicial $\$_0$ y final $\Box$. §

Estrategia	$\overrightarrow{A_{r,s+1}}$	$\overleftarrow{A_{r,s+1}}$
Ascendente	$\square$	$A_{r,s+1}$
Earley	$\overline{A_{r,s+1}}$	$\overline{\overline{A_{r,s+1}}}$
Descendente	$A_{r,s+1}$	$\square$

Tabla 5.2: Parámetros del esquema de compilación genérico de CFG en PDA

Ejemplo 5.1 Consideremos la gramática independiente del contexto definida por las producciones:

$$\begin{array}{ll} (0) & S \rightarrow X \\ (1) & X \rightarrow AXB \\ (2) & X \rightarrow \epsilon \\ (3) & A \rightarrow a \\ (4) & B \rightarrow b \end{array}$$

que genera el lenguaje $\{a^n b^n \mid n \geq 0\}$. En la tabla 5.4 se muestra el conjunto de transiciones del autómata a pila que se obtiene al aplicar el esquema de compilación descendente.

En la tabla 5.5 se muestra la secuencia de configuraciones que sigue el autómata para analizar correctamente la cadena de entrada *aabb*. La primera columna muestra la transición aplicada, la segunda el tipo de la regla que generó dicha transición, la tercera el contenido de la pila y la cuarta la parte que resta por leer de la cadena de entrada. Las configuraciones marcadas con $*$ son aquellas en las que hay más de una opción para proseguir el análisis. Se trata de configuraciones resultado de aplicar una transición de tipo [**CALL**]. Si el elemento apilado coincide con el lado izquierdo de varias producciones, debe proseguirse el análisis por cada una de ellas. En la tabla 5.5 sólo se muestran las configuraciones que llevan al reconocimiento de la cadena de entrada. El resto de las posibles configuraciones llevan al autómata a detenerse en configuraciones que no son finales. ¶

5.2.2 Estrategia Earley

Esquema de compilación 5.3 El esquema de compilación para una estrategia de tipo Earley de una gramática independiente del contexto en un autómata a pila queda definido por el conjunto de reglas mostrado en la tabla 5.6 y por los elementos inicial $\$_0$ y final $\overline{\overline{S}}$. §

5.2.3 Estrategia ascendente

Esquema de compilación 5.4 El esquema de compilación ascendente de una gramática independiente del contexto en un autómata a pila queda definido por el conjunto de reglas mostrado en la tabla 5.7 y por los elementos inicial $\$_0$ y final S. §

[INIT]	$\$_0 \longmapsto \$_0 \ \nabla_{0,0}$	
[CALL]	$\nabla_{r,s} \longmapsto \nabla_{r,s} \ \ A_{r,s+1}$	
[SEL]	$A_{r,0} \longmapsto \nabla_{r,0}$	$r \neq 0$
[PUB]	$\nabla_{r,n_r} \longmapsto \Box$	
[RET]	$\nabla_{r,s} \ \Box \ \longmapsto \nabla_{r,s+1}$	
[SCAN]	$A_{r,0} \overset{a}{\longmapsto} \Box$	$A_{r,0} \rightarrow a$

Tabla 5.3: Reglas del esquema de compilación descendente de CFG en PDA

(a)	**[INIT]**	$\$_0 \longmapsto \$_0 \ \nabla_{0,0}$
(b)	**[CALL]**	$\nabla_{0,0} \longmapsto \nabla_{0,0} \ \ X$
(c)	**[RET]**	$\nabla_{0,0} \ \Box \ \longmapsto \nabla_{0,1}$
(d)	**[PUB]**	$\nabla_{0,1} \longmapsto \Box$
(e)	**[SEL]**	$X \longmapsto \nabla_{1,0}$
(f)	**[CALL]**	$\nabla_{1,0} \longmapsto \nabla_{1,0} \ A$
(g)	**[RET]**	$\nabla_{1,0} \ \Box \ \longmapsto \nabla_{1,1}$
(h)	**[CALL]**	$\nabla_{1,1} \longmapsto \nabla_{1,1} \ X$
(i)	**[RET]**	$\nabla_{1,1} \ \Box \ \longmapsto \nabla_{1,2}$
(j)	**[CALL]**	$\nabla_{1,2} \longmapsto \nabla_{1,2} \ B$
(k)	**[RET]**	$\nabla_{1,2} \ \Box \ \longmapsto \nabla_{1,3}$
(l)	**[PUB]**	$\nabla_{1,3} \longmapsto \Box$
(m)	**[SCAN]**	$X \overset{\epsilon}{\longmapsto} \Box$
(n)	**[SCAN]**	$A \overset{a}{\longmapsto} \Box$
(p)	**[SCAN]**	$B \overset{b}{\longmapsto} \Box$

Tabla 5.4: Transiciones del autómata a pila con estrategia descendente

		$\$_0$					$aabb$	
(a)	**[INIT]**	$\$_0$	$\nabla_{0,0}$				$aabb$	
(b)	**[CALL]**	$\$_0$	$\nabla_{0,0}$	X			$aabb$	$*$
(e)	**[SEL]**	$\$_0$	$\nabla_{0,0}$	$\nabla_{1,0}$			$aabb$	
(f)	**[CALL]**	$\$_0$	$\nabla_{0,0}$	$\nabla_{1,0}$	A		$aabb$	
(n)	**[SCAN]**	$\$_0$	$\nabla_{0,0}$	$\nabla_{1,0}$	$\Box$		abb	
(g)	**[RET]**	$\$_0$	$\nabla_{0,0}$	$\nabla_{1,1}$			abb	
(h)	**[CALL]**	$\$_0$	$\nabla_{0,0}$	$\nabla_{1,1}$	X		abb	$*$
(e)	**[SEL]**	$\$_0$	$\nabla_{0,0}$	$\nabla_{1,1}$	$\nabla_{1,0}$		abb	
(f)	**[CALL]**	$\$_0$	$\nabla_{0,0}$	$\nabla_{1,1}$	$\nabla_{1,0}$	A	abb	
(n)	**[SCAN]**	$\$_0$	$\nabla_{0,0}$	$\nabla_{1,1}$	$\nabla_{1,0}$	$\Box$	bb	
(g)	**[RET]**	$\$_0$	$\nabla_{0,0}$	$\nabla_{1,1}$	$\nabla_{1,1}$		bb	
(h)	**[CALL]**	$\$_0$	$\nabla_{0,0}$	$\nabla_{1,1}$	$\nabla_{1,1}$	X	bb	$*$
(m)	**[SCAN]**	$\$_0$	$\nabla_{0,0}$	$\nabla_{1,1}$	$\nabla_{1,1}$	$\Box$	bb	
(i)	**[RET]**	$\$_0$	$\nabla_{0,0}$	$\nabla_{1,1}$	$\nabla_{1,2}$		bb	
(j)	**[CALL]**	$\$_0$	$\nabla_{0,0}$	$\nabla_{1,1}$	$\nabla_{1,2}$	B	bb	
(p)	**[SCAN]**	$\$_0$	$\nabla_{0,0}$	$\nabla_{1,1}$	$\nabla_{1,2}$	$\Box$	b	
(k)	**[RET]**	$\$_0$	$\nabla_{0,0}$	$\nabla_{1,1}$	$\nabla_{1,3}$		b	
(l)	**[PUB]**	$\$_0$	$\nabla_{0,0}$	$\nabla_{1,1}$	$\Box$		b	
(i)	**[RET]**	$\$_0$	$\nabla_{0,0}$	$\nabla_{1,2}$			b	
(j)	**[CALL]**	$\$_0$	$\nabla_{0,0}$	$\nabla_{1,2}$	B		b	
(p)	**[SCAN]**	$\$_0$	$\nabla_{0,0}$	$\nabla_{1,2}$	$\Box$			
(k)	**[RET]**	$\$_0$	$\nabla_{0,0}$	$\nabla_{1,3}$				
(l)	**[PUB]**	$\$_0$	$\nabla_{0,0}$	$\Box$				
(c)	**[RET]**	$\$_0$	$\nabla_{0,1}$					
(d)	**[PUB]**	$\$_0$	$\Box$					

Tabla 5.5: Configuraciones del autómata a pila descendente durante el análisis de $aabb$

[INIT]	$\$_0 \longmapsto \$_0 \ \nabla_{0,0}$	
[CALL]	$\nabla_{r,s} \longmapsto \nabla_{r,s} \ \overline{A_{r,s+1}}$	
[SEL]	$\overline{A_{r,0}} \longmapsto \nabla_{r,0}$	$r \neq 0$
[PUB]	$\nabla_{r,n_r} \longmapsto \overline{\overline{A_{r,0}}}$	
[RET]	$\nabla_{r,s} \ \overline{\overline{A_{r,s+1}}} \longmapsto \nabla_{r,s+1}$	
[SCAN]	$\overline{A_{r,0}} \overset{a}{\longmapsto} \overline{\overline{A_{r,0}}}$	$A_{r,0} \to a$

Tabla 5.6: Reglas del esquema de compilación Earley de CFG en PDA

[INIT]	$\$_0 \longmapsto \$_0 \ \nabla_{0,0}$	
[CALL]	$\nabla_{r,s} \longmapsto \nabla_{r,s} \ \square$	
[SEL]	$\square \longmapsto \nabla_{r,0}$	$r \neq 0$
[PUB]	$\nabla_{r,n_r} \longmapsto A_{r,0}$	
[RET]	$\nabla_{r,s} \ A_{r,s+1} \longmapsto \nabla_{r,s+1}$	
[SCAN]	$\square \overset{a}{\longmapsto} A_{r,0}$	$A_{r,0} \to a$

Tabla 5.7: Reglas del esquema de compilación ascendente de CFG en PDA

5.3 Tabulación

La independencia del contexto de las transiciones de los autómatas a pila permite tabular la ejecución de los mismos. En esta sección presentamos dos técnicas diferentes para la tabulación de los autómatas a pila, una propuesta por Lang [104, 107] y la otra propuesta por Nederhof [126].

5.3.1 La técnica de Lang

En una gramática independiente del contexto, si $B \overset{*}{\Rightarrow} \delta$ entonces $\alpha B \beta \overset{*}{\Rightarrow} \alpha \delta \beta$ para todo $\alpha, \beta \in (V_N \cup V_T)^*$. Esta misma independencia del contexto se traslada a los autómatas a pila, de tal modo que si

$$(B, a_{i+1} \dots a_j \dots a_n) \overset{*}{\vdash} (BC, a_{j+1} \dots a_n)$$

entonces también se cumple que

$$(\xi B, a_{i+1} \dots a_j \dots a_n) \overset{*}{\vdash} (\xi BC, a_{j+1} \dots a_n)$$

para todo $\xi \in V_S^*$. Denominaremos *derivaciones independientes del contexto* a este tipo de derivaciones. Observamos que este tipo de derivaciones presenta gran semejanza con las transiciones PUSH.

Para representar una derivación independiente del contexto sólo precisamos almacenar los símbolos de pila B y C más las posiciones de la cadena de entrada i y j, puesto que la derivación es independiente de ξ. La técnica de tabulación de autómatas a pila propuesta por Lang [104, 107] se basa precisamente en reemplazar la manipulación de pilas por la manipulación de ítems de la forma

$$[B, i, C, j]$$

que representan de forma compacta el conjunto de derivaciones independientes del contexto que comparten los elementos de la cima de la pila. Los ítems se combinan mediante *reglas de combinación* de la forma $\frac{ants}{cons}\ trans$, donde $cons$ es el ítem consecuente que se obtiene al aplicar la transición $trans$ sobre los ítems antecedentes $ants$. A continuación mostramos las reglas de combinación de ítems para los tres tipos de transiciones:

Transiciones SWAP: la aplicación de una transición $C \overset{a}{\longmapsto} F$ produce la derivación $(\xi C, a_j \ldots a_n) \vdash (\xi F, a_k \ldots a_n)$ en un autómata a pila, donde $k = j$ si $a = \epsilon$ y $k = j+1$ si $a = a_{j+1}$. Dada la derivación independiente del contexto $(\xi' B, a_{i+1} \ldots a_n) \overset{*}{\vdash} (\xi' BC, a_{j+1} \ldots a_n)$ que da lugar a la configuración $(\xi' BC, a_{j+1} \ldots a_n)$, tras la aplicación de la transición obtendremos la derivación independiente del contexto $(\xi' B, a_{i+1} \ldots a_n) \overset{*}{\vdash} (\xi' BF, a_{k+1} \ldots a_n)$. La correspondiente regla de combinación de ítems es

$$\frac{[B, i, C, j]}{[B, i, F, k]} \quad C \overset{a}{\longmapsto} F$$

Transiciones PUSH: la aplicación de una transición $C \overset{a}{\longmapsto} CF$ produce la derivación $(\xi C, a_{j+1} \ldots a_n) \vdash (\xi CF, a_{k+1} \ldots a_n)$, donde $k = j$ si $a = \epsilon$ y $k = j+1$ si $a = a_{j+1}$. Esta derivación es por sí misma una derivación independiente del contexto, por lo que la correspondiente regla de combinación de ítems es

$$\frac{[B, i, C, j]}{[C, j, F, k]} \quad C \overset{a}{\longmapsto} CF$$

Transiciones POP: la aplicación de una transición $CF \overset{a}{\longmapsto} G$ produce la derivación $(\xi CF, a_{k+1} \ldots a_n) \vdash (\xi G, a_{l+1} \ldots a_n)$. La configuración $(\xi CF, a_{k+1} \ldots a_n)$ refleja una derivación independiente del contexto $(\xi C, a_{j+1} \ldots a_n) \overset{*}{\vdash} (\xi CF, a_{k+1} \ldots a_n)$, pero además necesitamos la derivación independiente del contexto $(\xi' B, a_{i+1} \ldots a_n) \overset{*}{\vdash} (\xi' BC, a_{j+1} \ldots a_n)$ que colocó C en la cima de la pila para obtener la derivación independiente del contexto $(\xi' B, a_{i+1} \ldots a_n) \overset{*}{\vdash} (\xi' BG, a_{l+1} \ldots a_n)$ resultante de la aplicación de la transición, donde $l = k$ si $a = \epsilon$ y $l = k+1$ si $a = a_{k+1}$. La correspondiente regla de combinación de ítems es

$$\frac{\begin{array}{c}[C, j, F, k] \\ [B, i, C, j]\end{array}}{[B, i, G, l]} \quad CF \overset{a}{\longmapsto} G$$

La manipulación de configuraciones mediante la aplicación de transiciones es equivalente a la manipulación de ítems mediante las reglas de combinación correspondientes a cada transición [104, 107].

5.3.2 La técnica de Nederhof

Aunque Nederhof no ha presentado una técnica específica para la tabulación de autómatas a pila, la misma se obtiene como un caso especial de la técnica de tabulación propuesta por dicho autor para los autómatas lineales de índices [126].

A diferencia de la técnica de Lang, que traslada la propiedad de independencia del contexto de las gramáticas independientes del contexto a derivaciones con la forma de transiciones de PUSH, la técnica de tabulación de Nederhof traslada dicha propiedad a derivaciones con la forma de transiciones SWAP, puesto que si tenemos una derivación

$$(B, a_i \dots a_j \dots a_n) \overset{*}{\vdash} (C, a_j \dots a_n)$$

entonces también se cumple que

$$(\xi B, a_i \dots a_j \dots a_n) \overset{*}{\vdash} (\xi C, a_j \dots a_n)$$

para todo $\xi \in V_S^*$. Al igual que en la técnica de Lang, estas derivaciones también se denominan *derivaciones independientes del contexto*.

Para representar una derivación independiente del contexto sólo precisamos almacenar B, C y las posiciones i y j, por lo que se utilizarán ítems de la forma

$$[B, i, C, j]$$

Estos ítems se manipulan mediante reglas de combinación, una para cada tipo de transición:

Transiciones SWAP: la aplicación de una transición $C \overset{a}{\longmapsto} F$ produce una derivación $(\xi C, a_j \dots a_n) \vdash (\xi F, a_k \dots a_n)$ en un autómata a pila, donde $k = j$ si $a = \epsilon$ y $k = j + 1$ si $a = a_{j+1}$. Dada la derivación independiente del contexto $(\xi B, a_i \dots a_n) \overset{*}{\vdash} (\xi C, a_j \dots a_n)$, tras la aplicación de la transición obtendremos la derivación independiente del contexto $(\xi B, a_i \dots a_n) \overset{*}{\vdash} (\xi F, a_k \dots a_n)$. La correspondiente regla de combinación de ítems es

$$\frac{[B, i, C, j]}{[B, i, F, k]} \quad C \overset{a}{\longmapsto} F$$

Transiciones PUSH: la aplicación de una transición $C \overset{a}{\longmapsto} CF$ produce la derivación $(\xi C, a_{j+1} \dots a_n) \vdash (\xi CF, a_{k+1} \dots a_n)$, donde $k = j$ si $a = \epsilon$ y $k = j + 1$ si $a = a_{j+1}$. Dada la derivación independiente del contexto $(\xi B, a_i \dots a_n) \overset{*}{\vdash} (\xi C, a_j \dots a_n)$, tras la aplicación de la transición obtendremos la derivación independiente del contexto $(\xi F, a_{k+1} \dots a_n) \overset{0}{\vdash} (\xi F, a_{k+1} \dots a_n)$. La correspondiente regla de combinación de ítems es

$$\frac{[B, i, C, j]}{[F, k, F, k]} \quad C \overset{a}{\longmapsto} CF$$

Transiciones POP: la aplicación de una transición $CF \overset{a}{\longmapsto} G$ produce una derivación $(\xi CF, a_{k+1} \dots a_n) \vdash (\xi G, a_{l+1} \dots a_n)$, donde $l = k$ si $a = \epsilon$ y $l = k + 1$ si $a = a_{k+1}$. La configuración $(\xi CF, a_{k+1} \dots a_n)$ refleja una derivación constituida por

1. una derivación independiente del contexto $(\xi B, a_{i+1} \dots a_n) \overset{*}{\vdash} (\xi C, a_{j+1} \dots a_n)$;
2. un paso de derivación $(\xi C, a_{j+1} \dots a_n) \vdash (\xi CF', a_{k'+1} \dots a_n)$ resultado de la aplicación de una transición $C \overset{b}{\longmapsto} CF'$, donde $k' = j$ si $b = \epsilon$ y $k' = j + 1$ si $b = a_{j+1}$;
3. una derivación independiente del contexto $(\xi CF', a_{k'+1} \dots a_n) \overset{*}{\vdash} (\xi CF, a_{k+1} \dots a_n)$.

En consecuencia, la regla de combinación tendrá la forma:

$$\frac{\begin{array}{c}[F', k', F, k] \\ [B, i, C, j]\end{array}}{[B, i, G, l]} \quad \begin{array}{l} C \overset{b}{\longmapsto} CF' \\ CF \overset{a}{\longmapsto} G \end{array}$$

Capítulo 6

Autómatas a pila embebidos

En este capítulo se presentan los autómatas a pila embebidos, el primer modelo de autómatas descrito en la literatura que reconocía exactamente la clase de los lenguajes de adjunción de árboles. Las principales aportaciones de este capítulo son la redefinición de este tipo de autómatas, eliminando la necesidad de un control finito y modificando en consecuencia la forma de las transiciones, y el establecimiento de una técnica de tabulación que permite su ejecución en tiempo polinomial. Este capítulo está basado en [18].

6.1 Introducción

Un autómata para un formalismo gramatical puede simular una derivación en dicho formalismo básicamente de dos maneras: descendente o ascendente. En el caso descendente, las transiciones del autómata tratan de recorrer el árbol derivado desde la raíz hacia las hojas, mientras comprueba si la cadena de entrada concuerda con la cadena esperada. En el caso ascendente, la cadena de entrada se va introduciendo en el almacenamiento del autómata al tiempo que se trata de recorrer el árbol derivado desde las hojas hacia la raíz. En el caso de los autómatas a pila, que constituyen el modelo de autómata equivalente a las gramáticas independientes del contexto, el comportamiento descendente o ascendente viene especificado por el conjunto de transiciones utilizados, no por la forma en que los autómatas a pila son definidos en tanto que sistema formal. No ocurre lo mismo en el caso de los autómatas a pila embebidos, que constituyen uno de los modelos de autómata equivalentes a las gramáticas de adjunción de árboles. Los autómatas a pila embebidos sólo pueden simular derivaciones de una gramática de adjunción de árboles en las que las adjunciones se reconocen de modo descendente, y no pueden ser utilizados para simular derivaciones en las cuales las derivaciones se reconocen de modo ascendente.

6.2 Autómatas a pila embebidos

Los autómatas a pila embebidos (*Embedded Push-Down Automata*, EPDA) [218, 206] son una extensión de los autómatas a pila que reconocen exactamente la clase de los lenguajes de adjunción de árboles. La principal diferencia entre un PDA y un EPDA estriba en que mientras los primeros trabajan con una pila de símbolos elementales, los segundos trabajan con un pila, que denominamos pila principal, que a su vez contiene pilas.

Un autómata a pila embebido consta de tres componentes: una cadena de entrada, un control finito y una pila que contiene pilas. Su *configuración* en un momento dado viene determinada por el estado del control finito en que se encuentra el autómata, el contenido de la pila y la parte de la cadena de entrada que resta por leer, tal y como se muestra en la figura 6.1. Un conjunto de *transiciones* permite cambiar de configuración. Para ello, las transiciones pueden consultar el estado q del control finito, el siguiente símbolo terminal a de la cadena de entrada y el elemento Z en la cima de la pila $[\alpha Z$ que se encuentra en la cima de la pila principal. Como resultado de la aplicación de una transición, el autómata puede cambiar al estado q', avanzar una posición en la cadena de entrada y reemplazar Z por una secuencia $Z_m \dots Z_1$ de símbolos de pila para dar lugar a una pila $[\alpha'$ en la cima de la pila principal. Adicionalmente, la pila $[\alpha'$ puede ser reemplazada por una secuencia de $k \geq 0$ pilas, que incluirán a la pila $[\alpha'$ si esta no es la vacía. La figura 6.2 muestra el resultado de aplicar una transición

$$(q',\ [\alpha_k \dots [\alpha_{i+1},\ Z_m \dots Z_1,\ [\alpha_i \dots [\alpha_1) \in \delta(q, a, Z)$$

al autómata a pila embebido de la figura 6.1. Esta transición reemplaza el elemento Z de la cima de la pila $[\alpha Z$ por $Z_m \dots Z_1$ para dar lugar a la pila $[\alpha Z_m \dots Z_1$. Bajo dicha pila se sitúan $k - i$ pilas $[\alpha_j$ con $i + 1 \leq j \leq k$. Por encima se sitúan i pilas $[\alpha_j$ con $1 \leq j \leq i$. La cima de $[\alpha_1$ se convierte en la cima de la pila principal. El marcador $[$ no pertenece a V_S y se utiliza simplemente para separar el contenido de las diferentes pilas.

Si como resultado de la aplicación de una transición la pila situada en la cima de la pila principal quedase vacía, dicha pila sería eliminada y la pila situada inmediatamente debajo pasaría a a ocupar la cima de la pila principal.

De acuerdo con Vijay-Shanker en [206], definiremos formalmente un autómata a pila embebido como una tupla $(Q, V_T, V_S, \delta, q_0, Q_F, \$_0)$ donde:

- Q es un conjunto finito de estados.
- V_T es un conjunto finito de símbolos terminales.
- V_S es un conjunto finito de símbolos de pila.
- q_0 es el estado inicial.
- $Q_F \subseteq Q$ es el conjunto de estados finales.
- $\$_0 \in V_S$ es el símbolo inicial de la pila.
- δ es una relación de $Q \times V_T \cup \{\epsilon\} \times V_S$ en subconjuntos finitos de $Q \times ([V_S^*)^* \times V_S^* \times ([V_S^*)^*$, donde $[\ \notin V_S$ es un símbolo utilizado para separar las diferentes pilas que componen la pila principal.

La *configuración* de un autómata a pila embebido en un momento dado viene definida por el triple (q, Υ, w), donde $q \in Q$ indica el estado en el que se encuentra, $\Upsilon \in ([V_S^*)^*$ el contenido de la pila principal y $w \in V_T^*$ la parte de la cadena de entrada que resta por leer. El cambio de una configuración a otra viene determinado por la aplicación de una transición, de tal modo que si $(q, \Upsilon[\alpha Z, aw)$ es una configuración y

$$(q',\ [\alpha_k \dots [\alpha_{i+1},\ Z_m \dots Z_1,\ [\alpha_i \dots [\alpha_1) \in \delta(q, a, Z)$$

es una transición, entonces el autómata a pila embebido pasa a la nueva configuración

$$(q', \Upsilon[\alpha_k \dots [\alpha_{i+1}[\alpha Z_m \dots Z_1[\alpha_i \dots [\alpha_1, w)$$

Este hecho se denota mediante

$$(q, \Upsilon[\alpha Z, aw) \vdash (q', \Upsilon[\alpha_k \dots [\alpha_{i+1}[\alpha Z_m \dots Z_1[\alpha_i \dots [\alpha_1, w)$$

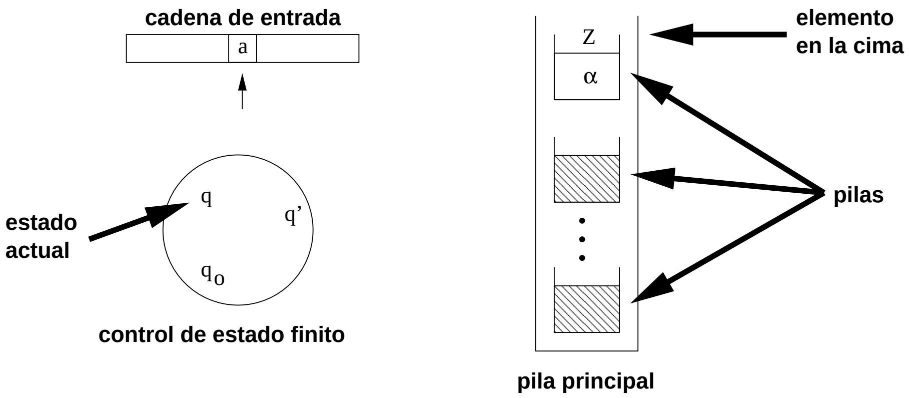

Figura 6.1: Autómata a pila embebido

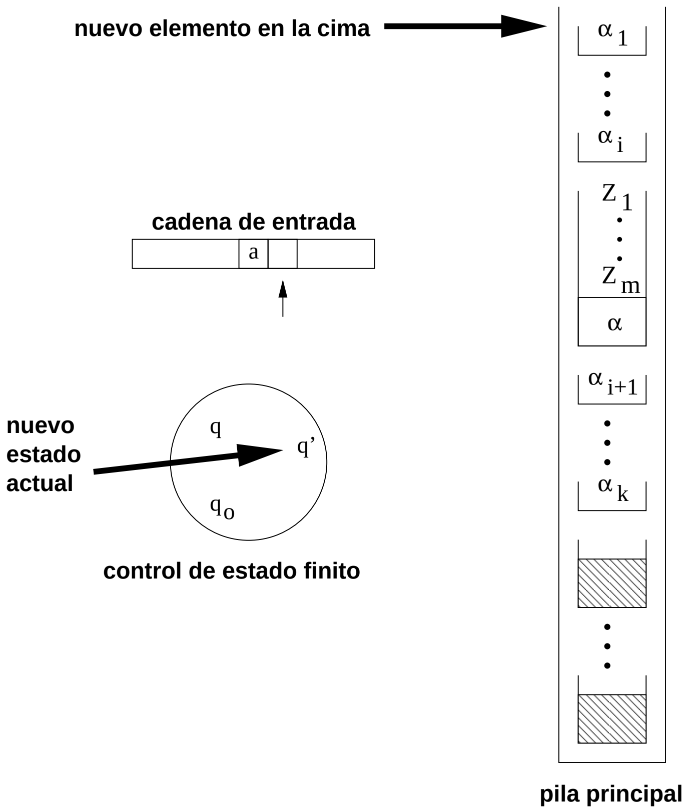

Figura 6.2: Autómata a pila embebido después de una transición

Denotamos por $\overset{*}{\vdash}$ el cierre reflexivo y transitivo de $\vdash$.

El *lenguaje aceptado por estado final* por un autómata a pila embebido viene determinado por el conjunto de cadenas $w \in V_T^*$ tal que $(q_0, [\$_0, w) \overset{*}{\vdash} (p, \Upsilon, \epsilon)$, donde $p \in Q_F$ y $\Upsilon \in ([V_S^*)^*$.

El *lenguaje aceptado por pila vacía* por un autómata a pila embebido viene determinado por el conjunto de cadenas $w \in V_T^*$ tal que $(q_0, [\$_0, w) \overset{*}{\vdash} (q, \epsilon, \epsilon)$ para cualquier $q \in Q$.

Dado un EPDA que reconoce un determinado lenguaje por estado final, es posible construir otro EPDA que reconoce el mismo lenguaje por pila vacía y viceversa [206].

Ejemplo 6.1 El autómata embebido a pila definido por la tupla $(\{q_0, q_1, q_2, q_3\}, \{a, b, c, d\}, \{\$_0, B, C, D\}, \delta, q_0, \emptyset, \$_0)$, donde δ contiene las transiciones mostradas en la tabla 6.1, acepta el lenguaje $\{a^n b^n c^n d^n \mid n \geq 0\}$ por pila vacía. En la tabla 6.1 también se muestra la secuencia de configuraciones que sigue el autómata para analizar correctamente la cadena de entrada *aabbccdd*. La primera columna muestra la transición aplicada, la segunda el estado, la tercera el contenido de la pila y la cuarta la parte que resta por leer de la cadena de entrada. ¶

(a)	$(q_0, [D, B, \epsilon) \in \delta(q_0, a, \$_0)$
(b)	$(q_0, [D, BB, \epsilon) \in \delta(q_0, a, B)$
(c)	$(q_1, [C, \epsilon, \epsilon) \in \delta(q_0, b, B)$
(d)	$(q_1, [C, \epsilon, \epsilon) \in \delta(q_1, b, B)$
(e)	$(q_2, \epsilon, \epsilon, \epsilon) \in \delta(q_1, c, C)$
(f)	$(q_2, \epsilon, \epsilon, \epsilon) \in \delta(q_2, c, C)$
(g)	$(q_3, \epsilon, \epsilon, \epsilon) \in \delta(q_2, d, D)$
(h)	$(q_3, \epsilon, \epsilon, \epsilon) \in \delta(q_3, d, D)$
(i)	$(q_0, \epsilon, \epsilon, \epsilon) \in \delta(q_0, \epsilon, \$_0)$

	q_0	$[\$_0$	$aabbccdd$
(a)	q_0	$[D[B$	$abbccdd$
(b)	q_0	$[D[D[BB$	$bbccdd$
(c)	q_1	$[D[D[C[B$	$bccdd$
(d)	q_1	$[D[D[C[C$	$ccdd$
(e)	q_2	$[D[D[C$	cdd
(f)	q_2	$[D[D$	dd
(g)	q_3	$[D$	d
(h)	q_3		

Tabla 6.1: Transiciones del autómata a pila embebido que acepta $\{a^n b^n c^n d^n \mid n > 0\}$ (izquierda) y configuraciones de dicho autómata durante el análisis de *aabbccdd* (derecha)

Johnson define en [89] una extensión de los autómatas a pila embebidos en la cual los símbolos de pila son reemplazados por términos lógicos de primer orden. Johnson denomina a tal extensión *autómatas lógicos a pila embebidos* (*Logic Embedded Push-Down Automata*, LEPDA) y propone una implementación en el lenguaje de programación lógica Prolog [225].

Weir generaliza en [231] el concepto de autómata a pila embebido al definir una progresión de modelos de autómatas, denominada *autómatas a pila iterados lineales* en la cual los autómatas de nivel 0, que no utilizan pila, aceptan la clase de los lenguajes regulares; los autómatas de nivel 1, que utilizan una pila, aceptan la clase de los lenguajes independientes del contexto; y los autómatas de nivel 2, que hacen uso de una pila de pilas, aceptan la clase de los lenguajes de adjunción de árboles.

6.3 Autómatas a pila embebidos sin estados

Al igual que en el caso de los autómatas a pila, el control finito es un elemento prescindible de los autómatas a pila embebidos, puesto que el estado correspondiente a una configuración puede ser incluido en el elemento de la cima de la pila. Como resultado obtenemos una definición alternativa, que juzgamos más simple y homogénea, según la cual un autómata a pila embebido es una tupla $(V_T, V_S, \Theta, \$_0, \$_f)$ en la cual:

- V_T es un conjunto finito de símbolos terminales.
- V_S es un conjunto finito de símbolos de pila.
- $\$_0 \in V_S$ es el símbolo inicial de pila.
- $\$_f \in V_S$ es el símbolo final de pila.
- Θ es un conjunto de transiciones, cada una de las cuales pertenece a uno de los siguientes tipos, donde $C, F, G \in V_S$, $\Upsilon \in ([\![V_S^*)^*$, $\alpha \in V_S^*$ y $a \in V_T \cup \{\epsilon\}$:

  SWAP: Transiciones de la forma $C \overset{a}{\longmapsto} F$ que reemplazan el elemento C de la cima de la pila por el elemento F mientras se lee a de la cadena de entrada. El resultado de aplicar una transición de este tipo a una pila $\Upsilon[\![\alpha C$ es una pila $\Upsilon[\![\alpha F$.

  PUSH: Transiciones de la forma $C \overset{a}{\longmapsto} CF$ que apilan un nuevo elemento F en la pila mientras se lee a de la cadena de entrada. El resultado de aplicar una transición de este tipo a una pila $\Upsilon[\![\alpha C$ es una pila $\Upsilon[\![\alpha CF$.

  POP: Transiciones de la forma $CF \overset{a}{\longmapsto} G$ que eliminan los dos elementos C y F de la cima de la pila y los sustituyen por G mientras se lee a de la cadena de entrada. El resultado de aplicar una transición de este tipo a una pila $\Upsilon[\![\alpha CF$ es una pila $\Upsilon[\![\alpha G$.

  WRAP-A: Transiciones de la forma $C \overset{a}{\longmapsto} C, [\![F$ que sitúan una nueva pila $[\![F$ en la cima de la pila principal mientras se lee a de la cadena de entrada. El resultado de aplicar una transición de este tipo a una pila $\Upsilon[\![\alpha C$ es una pila $\Upsilon[\![\alpha C[\![F$.

  WRAP-B: Transiciones de la forma $C \overset{a}{\longmapsto} [\![C, F$ que sitúan una nueva pila $[\![C$ bajo la pila que actualmente ocupa la cima de la pila principal y cambian C por F en la cima de dicha pila, todo ello mientras se lee a de la cadena de entrada. El resultado de aplicar una transición de este tipo a una pila $\Upsilon[\![\alpha C$ es una pila $\Upsilon[\![C[\![\alpha F$.

  UNWRAP: Transiciones de la forma $C, [\![F \overset{a}{\longmapsto} G$ que eliminan la pila $[\![F$ de la cima de la pila principal, mientras se lee a de la cadena de entrada. El resultado de aplicar una transición de este tipo a una pila $\Upsilon[\![\alpha C[\![F$ es una pila $\Upsilon[\![\alpha G$.

  Los tres primeros tipos se corresponden con los tipos de transiciones presentes en los autómatas a pila, aunque en este caso trabajan en la pila situada en la cima de la pila principal, tal y como se observa en la figura 6.3. Las transiciones de tipo WRAP-A y WRAP-B permiten situar una nueva pila encima y debajo, respectivamente, de la pila situada en la cima de la pila principal, como se puede observar en la figura 6.4. Las transiciones de tipo UNWRAP permiten eliminar una pila de la cima de la pila principal, como muestra la figura 6.5.

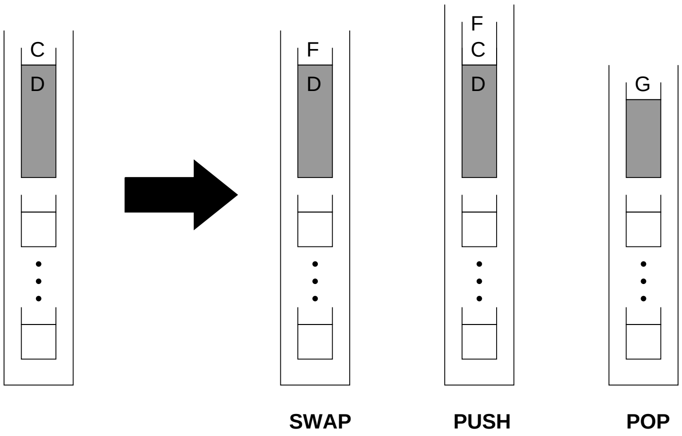

Figura 6.3: Transiciones SWAP, PUSH y POP

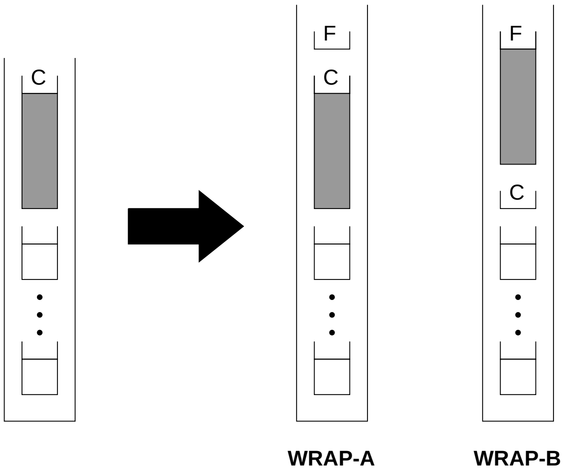

Figura 6.4: Transiciones WRAP-A y WRAP-B

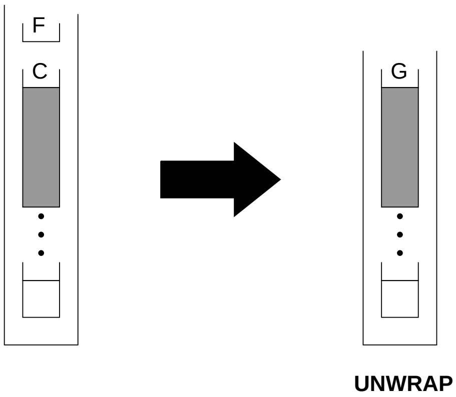

Figura 6.5: Transición UNWRAP

La *configuración* en un momento dado de un autómata a pila embebido sin estados viene determinada por el par (Υ, w), donde Υ es el contenido de la pila y w la parte de la cadena de entrada que resta por leer. Una configuración (Υ, aw) deriva una configuración (Υ', w), hecho que denotamos mediante $(\Upsilon, aw) \vdash (\Upsilon', w)$, si y sólo si existe una transición que aplicada a Υ devuelve Υ' y consume a de la cadena de entrada. En caso de ser necesario identificar una derivación d concreta, utilizaremos la notación $\vdash_d$. Denotamos por $\overset{*}{\vdash}$ el cierre reflexivo y transitivo de $\vdash$.

Decimos que una cadena de entrada w es aceptada por un autómata a pila embebido si $([\$_0, w) \overset{*}{\vdash} ([\$_0[\$_f, \epsilon)$. El *lenguaje aceptado* por un autómata a pila embebido viene determinado por el conjunto de cadenas $w \in V_T^*$ tal que $([\$_0, w) \overset{*}{\vdash} ([\$_0[\$_f, \epsilon)$.

En lo que resta, cuando nos refiramos a EPDA sin especificar si se trata de la versión con estados o sin estados, debe entenderse que nos estamos refiriendo a la versión sin estados que acabamos de definir.

Ejemplo 6.2 El autómata embebido a pila sin estados definido por la tupla $(\{a, b, c, d\}, \{\$_0, \$_f, B, C, D, E\}, \Theta, \$_0, \$_f)$, donde Θ contiene las transiciones mostradas en la tabla 6.2, acepta el lenguaje $\{a^n b^n c^n d^n \mid n \geq 0\}$ por pila vacía. En la tabla 6.2 también se muestra la secuencia de configuraciones que sigue el autómata para analizar correctamente la cadena de entrada *aabbccdd*. La primera columna muestra la transición aplicada, la segunda el contenido de la pila y la tercera la parte que resta por leer de la cadena de entrada. ¶

Teorema 6.1 *Mediante la utilización de un conjunto de transiciones SWAP, PUSH, POP, WRAP-A, WRAP-B y UNWRAP es posible emular transiciones complejas del tipo*

$$DB \overset{a}{\longmapsto} \;\; [F_k \ldots [F_{i+1},\ EC_1 \ldots C_m,\ [F_i \ldots [F_1$$

tal que su aplicación da lugar a un paso de derivación

$$\Upsilon[\alpha DB \;\;\vdash\;\; \Upsilon[F_k \ldots [F_{i+1}[\alpha EC_1 \ldots C_m, [F_i \ldots [F_1$$

donde

- $a \in V_T \cup \{\epsilon\}$.
- $0 \leq m \leq 2$.
- $B, C_1, \ldots, C_2, F_1, \ldots, F_k \in V_S$.
- *Si* $m = 2$ *entonces* $C_1 = B$.
- *Si* $m = 0$ *entonces* $D, E \in V_S$.
- *Si* $m > 0$ *entonces* $D = E = \epsilon$.

Demostración:

Realizaremos una prueba constructiva creando un procedimiento que permita emular el tipo de transiciones propuesto en el teorema 6.1 a partir de las transiciones elementales de los autómatas a pila embebidos sin estados. Para ello precisamos nuevos símbolos de pila ∇_j y X', donde $i + 1 \leq j \leq k$ y $X \in V_S$. La emulación se realizará en tres fases: la primera se encargará de crear las pilas $[F_k \ldots [F_{i+1}$, la segunda de obtener la pila $[\alpha EC_1 \ldots C_m$ y la tercera se encargará de crear las pilas $[F_i \ldots [F_1$.

(a)	$\$_0 \longmapsto \$_0, [D$
(b)	$D \overset{a}{\longmapsto} [D, B$
(c)	$B \longmapsto BD$
(d)	$B \longmapsto BC$
(e)	$C \overset{b}{\longmapsto} [C, E$
(f)	$BE \longmapsto C$
(g)	$C, [C \overset{c}{\longmapsto} C$
(h)	$D, [C \overset{d}{\longmapsto} D$
(i)	$D, [D \overset{d}{\longmapsto} D$
(j)	$D \longmapsto \$_f$

	$[\$_0$	$aabbccdd$
(a)	$[\$_0[D$	$aabbccdd$
(b)	$[\$_0[D[B$	$abbccdd$
(c)	$[\$_0[D[BD$	$abbccdd$
(b)	$[\$_0[D[D[BB$	$bbccdd$
(d)	$[\$_0[D[D[BBC$	$bbccdd$
(e)	$[\$_0[D[D[C[BBE$	$bccdd$
(f)	$[\$_0[D[D[C[BC$	$bccdd$
(e)	$[\$_0[D[D[C[C[BE$	$ccdd$
(f)	$[\$_0[D[D[C[C[C$	$ccdd$
(g)	$[\$_0[D[D[C[C$	cdd
(g)	$[\$_0[D[D[C$	dd
(h)	$[\$_0[D[D$	d
(i)	$[\$_0[D$	
(j)	$[\$_0[\$_f$	

Tabla 6.2: Transiciones del EPDA sin estados que acepta $\{a^n b^n c^n d^n \mid n > 0\}$ (izquierda) y configuraciones de dicho autómata durante el análisis de $aabbccdd$ (derecha)

Fase 1 Partimos de una configuración $(\Upsilon[\alpha DB, aw)$, donde $D \in V_S$ si $m = 0$ y $D = \epsilon$ en otro caso. Comenzamos la emulación mediante la creación de una transición que apila ∇_k:

$$B \longmapsto B\nabla_k$$

Para emular la creación de cada una de las $k-i$ pilas unitarias creamos $k-i$ conjuntos con las tres transiciones siguientes:

$$\nabla_j \longmapsto \nabla_j F_j$$

$$F_j \longmapsto [F_j, F'_j$$

$$\nabla_j F'_j \longmapsto \nabla_{j-1}$$

donde $i+1 \leq j \leq k$.
En el caso de que $k = i$, esto es $[F_k \ldots [F_{i+1} = \epsilon$, esta fase consistiría únicamente de la transición

$$B \longmapsto B\nabla_i$$

Como resultado de la aplicación de esta fase obtendremos una configuración

$$(\Upsilon[F_k \ldots [F_{i+1}[\alpha DB\nabla_i, aw)$$

Fase 2 Las transiciones de esta fase dependen del valor de m, por lo que tenemos tres posibilidades:

- Si $m = 2$ se trata del apilamiento de C_2 sobre B:

$$\nabla_i \longmapsto C'_2$$

- Si $m = 1$ nos encontramos ante el cambio de B por C_1:

$$B\nabla_i \longmapsto C'_1$$

- Si $m = 0$ se trata de eliminar B de la pila:

$$B\nabla_i \longmapsto \nabla''_i$$

$$D\nabla''_i \longmapsto E'$$

Después de esta fase obtenemos una configuración $(\Upsilon[F_k \ldots [F_{i+1}[\alpha' X', aw)$, donde $\alpha' X' = \alpha E'$ si $m = 0$, $\alpha' X' = \alpha C'_1$ si $m = 1$ y $\alpha' X' = \alpha B C'_2$ si $m = 2$.

Fase 3 Para iniciar esta fase precisamos de una transición

$$X' \longmapsto X', [F'_i$$

donde $X' = E'$ si $m = 0$, $X' = C'_1$ si $m = 1$ y $X' = C'_2$ si $m = 2$. Para crear las restantes $i - 1$ pilas unitarias utilizaremos $i - 1$ transiciones de la forma

$$F'_j \longmapsto F'_j, [F'_{j-1}$$

donde $i \leq j < 1$, para finalizar con la transición

$$F'_1 \overset{a}{\longmapsto} F_1$$

En el caso de que $i = 0$, esto es $F_i \ldots [F_1 = \epsilon$, esta fase consistiría únicamente de la siguiente transición

$$X' \overset{a}{\longmapsto} X$$

donde $X = E$ si $m = 0$, $X = C_1$ si $m = 1$ y $X = C_2$ si $m = 2$.
Tras esta fase obtendremos la configuración

$$(\Upsilon[F_k \ldots [F_{i+1}[\alpha X'[F'_i \ldots [F_1, w)$$

Para emular completamente la transición deseada, deberemos crear una transición

$$F'j, [W \longmapsto Y$$

por cada transición $F_j, [W \longmapsto Y$ presente en el EPDA y una transición

$$X', [W \longmapsto Y$$

por cada transición $X, [W \longmapsto Y$ presente en el EPDA, donde $X = E$ si $m = 0$, $X = C_1$ si $m = 1$ y $X = C_2$ si $m = 2$.

□

Ejemplo 6.3 La transición $B \overset{a}{\longmapsto} [E[F, BC, [G[H$ se emula mediante el conjunto de transiciones mostrado en la tabla 6.3 más una transición $C'[X \longmapsto Y$ por cada transición $C[X \longmapsto Y$ presente en el EPDA original. Análogamente, deberemos crear una transición $G'[X \longmapsto Y$ por cada transición $G[X \longmapsto Y$ presente en el EPDA original.

En la misma tabla se muestra el resultado de aplicar las transiciones resultantes a una configuración $(\Upsilon[\alpha B, aw)$. La primera columna muestra la transición aplicada, la segunda muestra el contenido de la pila del EPDA y la tercera la parte de la cadena de entrada que falta por leer. ¶

(a)	$B \longmapsto B\nabla_4$
(b)	$\nabla_4 \longmapsto \nabla_4 E$
(c)	$E \longmapsto [E, E'$
(d)	$\nabla_4 E' \longmapsto \nabla_3$
(e)	$\nabla_3 \longmapsto \nabla_3 F$
(f)	$F \longmapsto [F, F'$
(g)	$\nabla_3 F' \longmapsto \nabla_2$
(h)	$\nabla_2 \longmapsto C'$
(i)	$C' \longmapsto C', [G'$
(j)	$G' \longmapsto G', [H'$
(k)	$H' \overset{a}{\longmapsto} H$

	$\Upsilon[\alpha B$	aw
(a)	$\Upsilon[\alpha B\nabla_4$	aw
(b)	$\Upsilon[\alpha B\nabla_4 E$	aw
(c)	$\Upsilon[E[\alpha B\nabla_4 E'$	aw
(d)	$\Upsilon[E[\alpha B\nabla_3$	aw
(e)	$\Upsilon[E[\alpha B\nabla_3 F$	aw
(f)	$\Upsilon[E[F[\alpha B\nabla_3 F'$	aw
(g)	$\Upsilon[E[F[\alpha B\nabla_2$	aw
(h)	$\Upsilon[E[F[\alpha BC'$	aw
(i)	$\Upsilon[E[F[\alpha BC'[G'$	aw
(j)	$\Upsilon[E[F[\alpha BC'[G'[H'$	aw
(k)	$\Upsilon[E[F[\alpha BC'[G'[H$	w

Tabla 6.3: Normalización de una transición compleja de un EPDA (izquierda) y emulación de la misma (derecha)

6.4 Equivalencia entre autómatas a pila embebidos sin estados y con estados

Para establecer la equivalencia entre la versión con estados y la versión sin estados del EPDA haremos uso de las transiciones complejas para EPDA sin estados definidas en el teorema 6.1 y de la forma normal para los EPDA con estados definida por Vijay-Shanker en [206]. Esta última establece que las transiciones deberán tener la forma

$$(q', [Z'_k \dots [Z'_{i+1}, Z_m \dots Z_1, [Z'_i \dots [Z'_1) \in \delta(q, a, Z)$$

donde $q, q' \in Q$, $a \in V_T \cup \{\epsilon\}$, $Z, Z_1, \dots, Z_m, Z'_1, \dots, Z_k \in V_S$ y $m \leq 2$. En consecuencia, las transiciones podrán apilar un elemento, cambiar el elemento de la cima o sacar el elemento de la cima de la pila que ocupa la cima de la pila principal, y podrán crear nuevas pilas con un único elemento encima y/o debajo de dicha pila.

Teorema 6.2 *Para todo EPDA sin estados $\mathcal{A}$, existe un EPDA con estados $\mathcal{A}'$ tal que el lenguaje aceptado por $\mathcal{A}$ es igual al lenguaje aceptado por $\mathcal{A}'$.*

Demostración:

Sea $\mathcal{A} = (V_T, V_S, \Theta, \$_0, \$_f)$ un EPDA sin estados. El EPDA con estados $\mathcal{A}' = (Q, V_T, V_S, \delta, q, \emptyset, \$_0)$ acepta el mismo lenguaje (por pila vacía) que $\mathcal{A}$ si las transiciones en δ se obtienen mediante una traducción adecuada de las transiciones en Θ. Consideremos todos los posibles tipos de transiciones en un EPDA sin estados:

SWAP: una transición $C \overset{a}{\longmapsto} F$ se traduce por una transición $(q, \epsilon, F, \epsilon) \in \delta(q, a, C)$.

PUSH: una transición $C \overset{a}{\longmapsto} CF$ se traduce por una transición $(q, \epsilon, CF, \epsilon) \in \delta(q, a, C)$.

POP: una transición $CF \overset{a}{\longmapsto} G$ se traduce por una transición $(q', \epsilon, \epsilon, \epsilon) \in \delta(q, a, F)$ más la transición $(q, \epsilon, G, \epsilon) \in \delta(q', \epsilon, C)$, donde q' es un estado que sólo se utiliza en estas dos transiciones.

WRAP-A: una transición $C \overset{a}{\longmapsto} C, [F$ se traduce por una transición $(q, \epsilon, C, [F) \in \delta(q, a, C)$.

WRAP-B: una transición $C \overset{a}{\longmapsto} [C, F$ se traduce por una transición $(q, [C, F, \epsilon) \in \delta(q, a, C)$.

UNWRAP: una transición $C, [F \overset{a}{\longmapsto} G$ se traduce por una transición $(q'', \epsilon, \epsilon, \epsilon) \in \delta(q, a, F)$ más una transición $(q'', \epsilon, G, \epsilon) \in \delta(q, \epsilon, C)$, donde q'' es un estado que sólo se utiliza en estas dos transiciones. Es importante recordar que las pilas vacías son eliminadas automáticamente de la cima en los EPDA con estados.

Adicionalmente, deberemos considerar las dos transiciones siguientes, que vacían la pila cuando se alcanza la configuración final $([\$_0 [\$_f, \epsilon)$:

$$(q', \epsilon, \epsilon, \epsilon) \in \delta(q, \epsilon, \$_f)$$

$$(q, \epsilon, \epsilon, \epsilon) \in \delta(q', \epsilon, \$_0)$$

El conjunto Q estará formado por el estado q y todos los estados q' y q'' utilizados en la traducción de transiciones POP y UNWRAP. □

Teorema 6.3 *Para todo EPDA con estados $\mathcal{A}$, existe un EPDA sin estados $\mathcal{A}'$ tal que el lenguaje aceptado por $\mathcal{A}$ es igual al lenguaje aceptado por $\mathcal{A}'$.*

Demostración:

Dado un EPDA con estados $\mathcal{A} = (Q, V_T, V_S, \delta, q_0, Q_F, \$_0)$ construiremos un EPDA sin estados $\mathcal{A}' = (V_T, V_S', \Theta, \$_0', \$_f')$ que acepte el mismo lenguaje que es aceptado por $\mathcal{A}$ por pila vacía. El conjunto V_S' estará formado por pares $\langle Z, q\rangle$ y $\langle Z^0, q\rangle$, donde $Z \in V_S \cup \{-\}$ y $q \in Q \cup \{-\}$, y por los elementos inicial $\$_0' = \langle \$_0, -\rangle$ y final $\$_f' = \langle -, -\rangle$.

Las transiciones en Θ tendrán el formato de las transiciones descritas en el teorema 6.1 y serán el resultado de traducir las transiciones en δ. Supondremos que las transiciones de $\mathcal{A}$ están en la forma normal definida por Vijay-Shanker en [206]. Consideremos cada uno de los posibles casos:

- Una transición
$$(q', \ [F_k \ldots [F_{i+1}, \ Z_1, \ [F_i \ldots [F_1) \in \delta(q, a, Z)$$
se traduce por una transición
$$\langle Z, q\rangle \overset{a}{\longmapsto} \ [\langle F_k^0, q'\rangle \ldots [\langle F_{i+1}^0, q'\rangle, \ \langle Z_1, q'\rangle, \ [\langle F_i^0, q'\rangle \ldots [\langle F_1^0, q'\rangle$$
- Una transición
$$(q', \ [F_k \ldots [F_{i+1}, \ Z_2 Z_1, \ [F_i \ldots [F_1) \in \delta(q, a, Z)$$
se traduce por una transición
$$\langle Z, q\rangle \overset{a}{\longmapsto} \ [\langle F_k^0, q'\rangle \ldots [\langle F_{i+1}^0, q'\rangle, \ \langle Z_2, q'\rangle\langle Z_1, q'\rangle, \ [\langle F_i^0, q'\rangle \ldots [\langle F_1^0, q'\rangle$$
- Una transición
$$(q', \ [F_k \ldots [F_{i+1}, \ \epsilon, \ [F_i \ldots [F_1) \in \delta(q, a, Z)$$
se traduce por un conjunto de transiciones
$$\langle D, q''\rangle\langle Z, q\rangle \overset{a}{\longmapsto} \ [\langle F_k^0, q'\rangle \ldots [\langle F_{i+1}^0, q'\rangle, \ \langle D, q'\rangle, \ [\langle F_i^0, q'\rangle \ldots [\langle F_1^0, q'\rangle$$
para todo $\langle D, q''\rangle \in V_S'$.

Adicionalmente, incluiremos una transición

$$\langle F^0, q\rangle \longmapsto \langle F^0, q\rangle\langle F, q\rangle$$

por todo $\langle F^0, q\rangle \in V'_S$ tal que $F \in V_S$ y $q \in Q$. Observamos que los pares $\langle F^0, q\rangle$ representan el papel de elemento inicial en cada una de las pilas individuales que componen la pila principal. Dicho elemento es necesario puesto que los EPDA con estados detectan si la pila en la cima de la pila principal está vacía, en cuyo caso la eliminan, todo ello sin intervención de transición alguna. Este comportamiento debe ser definido explícitamente en un EPDA sin estados mediante la creación de transiciones UNWRAP. La utilización de elementos $\langle F^0, q\rangle$ permite hacer corresponder una configuración $(q, \Upsilon [\!\![, w)$ de un EPDA con estados con una configuración $(\Upsilon' [\!\![\langle F^0, q\rangle, w)$ de un EPDA sin estados. En consecuencia, es necesario añadir a Θ una transición UNWRAP

$$\langle D, q'\rangle, [\!\![\langle F^0, q\rangle \longmapsto \langle D, q\rangle$$

por cada par

$$\langle D, q'\rangle \in V'_S - \{\langle \$_0, -\rangle\}$$
$$\langle F^0, q\rangle \in V'_S$$

Las dos transiciones siguientes

$$\langle \$_0, -\rangle \longmapsto \langle \$_0, -\rangle [\!\![\langle \$_0^0, -\rangle$$

$$\langle \$_0^0, -\rangle \longmapsto \langle \$_0^0, -\rangle\langle \$_0, q_0\rangle$$

configuran la pila inicial $\langle \$_0, -\rangle [\!\![\langle \$_0^0, -\rangle\langle \$_0, q_0\rangle$ sobre la cual pueden comenzar a ser aplicadas las transiciones definidas anteriormente.

Una cadena w es reconocida por $\mathcal{A}$ cuando se obtiene una configuración (q, ϵ, ϵ). Dicha configuración tiene su equivalente en al configuración $(\langle \$_0, -\rangle [\!\![\langle \$_0^0, q\rangle, \epsilon)$ de $\mathcal{A}'$. Las transiciones

$$\langle \$_0^0, q\rangle \longmapsto \langle -, -\rangle$$

para todo $q \in Q$ permiten obtener la configuración $(\langle \$_0, -\rangle [\!\![\langle -, -\rangle, \epsilon)$, que es la configuración final de $\mathcal{A}'$. □

Corolario 6.4 *Los EPDA sin estados son equivalentes a los EPDA con estados.*

Demostración:

El resultado enunciado se obtiene directamente a partir de los teorems 6.2 y 6.3. □

6.5 Esquemas de compilación de gramáticas independientes del contexto

Un modo sencillo de realizar un esquema de compilación para gramáticas independientes del contexto en autómatas a pila embebidos consiste en emular el comportamiento de un autómata a pila mediante un EPDA.

Partiendo de la configuración inicial $([\!\![\$_0, w)$ de un EPDA, podríamos pensar en realizar la emulación de un autómata a pila mediante la aplicación de las transiciones SWAP, PUSH y POP de este último sobre la pila $[\!\![\$_0$. Sin embargo, este enfoque impide alcanzar una configuración final del EPDA, puesto que estas tienen la forma $([\!\![\$_0 [\!\![\$_f, \epsilon)$, con dos pilas unitarias almacenadas en la pila principal.

Resolvemos el problema estableciendo que una configuración $(\$_0 B_1 B_2 \ldots B_n, w)$ de un autómata a pila equivale a una configuración $([\$_0 [B_1 [B_1 \ldots [B_1, w)$ de un EPDA, con lo cual las pilas unitarias pasan a representar el papel de los símbolos de pila. Con respecto a las transiciones, tenemos que:

- Las transiciones SWAP $C \longmapsto F$ permanecen sin cambios.
- Las transiciones PUSH $C \longmapsto CF$ del autómata a pila son reemplazadas por transiciones WRAP-A $C \longmapsto C, [F$ en el autómata a pila embebido.
- Las transiciones POP $CF \longmapsto G$ del autómata a pila son reemplazadas por transiciones UNWRAP $C, [F \longmapsto G$ en el autómata a pila embebido.

El caso inverso también es válido, de tal modo que un autómata a pila embebido cuyas configuraciones sólo contienen pilas unitarias y que sólo utiliza transiciones SWAP, WRAP-A y UNWRAP es equivalente a un autómata a pila cuyas configuraciones se obtienen a partir de las configuraciones del EPDA eliminando los símbolos $[$. El conjunto de transiciones contendrá las transiciones SWAP y el resultado de convertir las transiciones WRAP-A en transiciones PUSH y las transiciones UNWRAP en transiciones POP.

A continuación mostramos un esquema de compilación genérico de gramáticas independientes del contexto en autómatas a pila embebidos, obtenido a partir del esquema de compilación 5.1 mediante la aplicación de las transformaciones mencionadas anteriormente.

Esquema de compilación 6.1 El esquema de compilación genérico de una gramática independiente del contexto en un autómata a pila embebido queda definido por el siguiente conjunto de reglas y los elementos inicial $\$_0$ y final $\overleftarrow{S}$.

$$
\begin{array}{lll}
\textbf{[INIT]} & \$_0 \longmapsto \$_0 \ [\nabla_{0,0} & \\
\textbf{[CALL]} & \nabla_{r,s} \longmapsto \nabla_{r,s} \ [\overrightarrow{A_{r,s+1}} & \\
\textbf{[SEL]} & \overrightarrow{A_{r,0}} \longmapsto \nabla_{r,0} & r \neq 0 \\
\textbf{[PUB]} & \nabla_{r,n_r} \longmapsto \overleftarrow{A_{r,0}} & \\
\textbf{[RET]} & \nabla_{r,s} \ [\overleftarrow{A_{r,s+1}} \longmapsto \nabla_{r,s+1} & \\
\textbf{[SCAN]} & \overrightarrow{A_{r,0}} \overset{a}{\longmapsto} \overleftarrow{A_{r,0}} & A_{r,0} \to a
\end{array}
$$

§

6.6 Esquemas de compilación de gramáticas de adjunción de árboles

Un esquema de compilación de una gramática de adjunción de árboles en un autómata a pila embebido debe realizar las tareas siguientes:

- Recorrer los nodos de los árboles elementales.
- En el caso de la adjunción del árbol auxiliar β en un nodo N^γ:
  - Al llegar al nodo N^γ, suspender el recorrido de γ y comenzar el recorrido de β a partir de su raíz.

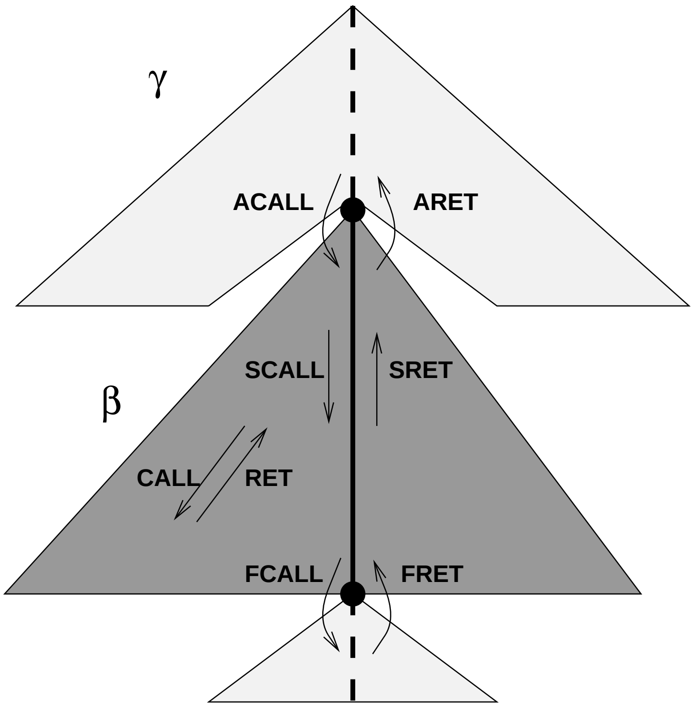

Figura 6.6: Reglas de compilación para TAG

– Al alcanzar el nodo pie de β, suspender el recorrido de este último para continuar el recorrido del subárbol de γ que cuelga del nodo N^γ.

El recorrido de un árbol elemental es equivalente al recorrido del conjunto de producciones independientes del contexto que lo componen, por lo que podemos utilizar para esta tarea las reglas de compilación [**CALL**], [**SEL**], [**PUB**] y [**RET**] definidas en el esquema de compilación 6.1. Al igual que en el capítulo 3, consideraremos una producción adicional $\top^\beta \rightarrow \mathbf{R}^\alpha$ para cada árbol inicial α y dos producciones adicionales para cada árbol auxiliar β: $\top^\beta \rightarrow \mathbf{R}^\beta$ y $\mathbf{F}^\beta \rightarrow \perp^\beta$, donde $\mathbf{R}^\beta$ y $\mathbf{F}^\beta$ se refieren a los nodos raíz y pie de β, respectivamente.

El tratamiento de la adjunción implica suspender el recorrido de un árbol al llegar a la raíz del árbol auxiliar, para retomarlo más tarde el alcanzar el nodo pie de dicho árbol auxiliar. Para ello hemos de poder transmitir, a través de la espina del árbol auxiliar, el nodo en el cual se realizó la adjunción. En el caso de que se realicen adjunciones en nodos de la espina, dichas adjunciones se irán apilando. Utilizaremos las propias pilas del EPDA para almacenar las pilas de adjunciones pendientes. Para ello, dotaremos de la siguiente semántica a las pilas del EPDA: dada una pila $[\alpha B$, el elemento B de la cima nos informa del punto en el que se encuentra el recorrido de un árbol elemental mientras que la parte restante α contiene la pila de adjunciones pendientes en dicho punto. La figura 6.6 muestra de modo intuitivo la misión de cada una de las reglas de compilación:

- La regla de compilación [**SCALL**] será la encargada de crear las transiciones que propaguen la pila de adjunciones pendientes a través de la espina de los árboles auxiliares, desde la raíz al pie.

- La regla [**SRET**] es una regla [**RET**] que trata con elementos de la espina de un árbol auxiliar. Se define con el fin de formar un par [**SCALL**]–[**SRET**] análogo al par [**CALL**]–[**RET**].

- Al llegar al nodo de adjunción N^γ, deberemos apilar dicho nodo en la pila de adjunciones pendientes y pasar a la raíz del árbol auxiliar β. La regla de compilación [**ACALL**] se encargará de crear las transiciones que realicen estas tareas.

- La regla de compilación [**ARET**] es un tipo especial de regla [**RET**] que permite continuar el recorrido de γ una vez que ha terminado la adjunción de β.

- Al llegar al nodo pie de β, deberemos eliminar el nodo N^γ de la pila de adjunciones pendientes y continuar el recorrido en el árbol que cuelga de dicho nodo. De crear las transiciones que realicen estas tareas se encargará la regla de compilación [**FCALL**].

- La regla de compilación [**FRET**] es un tipo especial de regla [**RET**] que permite continuar el recorrido a partir del nodo pie de β.

A continuación definimos un esquema de compilación genérico, derivado del esquema de compilación 6.1, en el cual se ha parametrizado la información concerniente al recorrido de los árboles elementales.

Esquema de compilación 6.2 El esquema de compilación genérico de una gramática de adjunción de árboles en un autómata a pila embebido queda definido por el conjunto de reglas mostrado en la figura 6.4 y los elementos inicial $\$_0$ y final $\overleftarrow{S}$. Es interesante señalar que las pilas de adjunciones pendientes no almacenan directamente los nodos $N^\gamma_{r,s+1}$ en los que se realizaron las adjunciones sino el elemento $\nabla^\gamma_{r,s}$ que indica el punto en el que se lanzó la adjunción, almacenado en las pilas bajo la forma $\Delta^\gamma_{r,s}$ para evitar confusiones. Podemos ver un símbolo Δ como un símbolo ∇ en espera de la finalización de una adjunción.

Las reglas de compilación [**ACALL**] y [**FCALL**] podrían escribirse alternativamente de la siguiente forma:

$$[\mathbf{ACALL}] \quad \nabla^\gamma_{r,s} \longmapsto [\nabla^\gamma_{r,s},\ \Delta^\gamma_{r,s}\ \overrightarrow{\top^\beta} \qquad \beta \in \mathrm{adj}(N^\gamma_{r,s+1})$$

$$[\mathbf{FCALL}] \quad \Delta^\gamma_{r,s}\ \nabla^\beta_{f,0} \longmapsto [\nabla^\beta_{f,0},\ \overrightarrow{N^\gamma_{r,s+1}} \qquad N^\beta_{f,0} = \mathbf{F}^\beta,\ \ \beta \in \mathrm{adj}(N^\gamma_{r,s+1})$$

pero como las transiciones resultantes no son transiciones elementales, hemos tenido que descomponer dichas reglas de compilación en dos pares de reglas: [**ACALL-a**] más [**ACALL-b**] y [**FCALL-a**] más [**FCALL-b**]. §

El esquema de compilación genérico establece que las adjunciones se reconocen de modo descendente, puesto que al pasar a la raíz de un árbol auxiliar se apila el nodo de adjunción en la pila de adjunciones pendientes y al llegar al nodo pie de dicho árbol se saca de la pila de adjunciones pendientes.

Es interesante remarcar que la regla de compilación [**CALL**] utiliza una transición WRAP-A con el significado de "crear una nueva pila que indique un nodo del árbol γ con una pila vacía de adjunciones pendientes", mientras que la regla de compilación [**SCALL**] utiliza una transición WRAP-B a la que dota del significado "Crear una nueva pila que indique el nodo del árbol γ y pasarle la pila de adjunciones pendientes". En la figura 6.4 se observa cómo la parte sombreada (en nuestro caso, la pila de adjunciones pendientes) permanece en su posición original en la operación WRAP-A mientras que en la operación WRAP-B la parte sombreada es pasada a la nueva pila en la cima, al tiempo que la pila que queda debajo sólo conserva el elemento C de su cima. Cuando se continúa el recorrido del árbol auxiliar a partir del pie ya no queda rastro de

[INIT]	$\$_0 \longmapsto \$_0,\ [\nabla^{\alpha}_{0,0}$	$\alpha \in \boldsymbol{I}$
[CALL]	$\nabla^{\gamma}_{r,s} \longmapsto \nabla^{\gamma}_{r,s}\ [\overrightarrow{N^{\gamma}_{r,s+1}}$	$N^{\gamma}_{r,s+1} \not\in \text{espina}(\gamma),\ \textbf{nil} \in \text{adj}(N^{\gamma}_{r,s+1})$
[SCALL]	$\nabla^{\beta}_{r,s} \longmapsto [\nabla^{\beta}_{r,s},\ \overrightarrow{N^{\beta}_{r,s+1}}$	$N^{\beta}_{r,s+1} \in \text{espina}(\beta),\ \textbf{nil} \in \text{adj}(N^{\beta}_{r,s+1})$
[SEL]	$\overrightarrow{N^{\gamma}_{r,0}} \longmapsto \nabla^{\gamma}_{r,0}$	
[PUB]	$\nabla^{\gamma}_{r,n_r} \longmapsto \overleftarrow{N^{\gamma}_{r,0}}$	
[RET]	$\nabla^{\gamma}_{r,s},\ [\overleftarrow{N^{\gamma}_{r,s+1}} \longmapsto \nabla^{\gamma}_{r,s+1}$	$N^{\gamma}_{r,s+1} \not\in \text{espina}(\gamma),\ \textbf{nil} \in \text{adj}(N^{\gamma}_{r,s+1})$
[SRET]	$\nabla^{\beta}_{r,s},\ [\overleftarrow{N^{\beta}_{r,s+1}} \longmapsto \nabla^{\beta}_{r,s+1}$	$N^{\gamma}_{r,s+1} \in \text{espina}(\beta),\ \textbf{nil} \in \text{adj}(N^{\beta}_{r,s+1})$
[SCAN]	$\overrightarrow{N^{\gamma}_{r,0}} \stackrel{a}{\longmapsto} \overleftarrow{N^{\gamma}_{r,0}}$	$N^{\gamma}_{r,0} \rightarrow a$
[ACALL-a]	$\nabla^{\gamma}_{r,s} \longmapsto [\nabla^{\gamma}_{r,s},\ \Delta^{\gamma}_{r,s}$	$\text{adj}(N^{\gamma}_{r,s+1}) \neq \{\textbf{nil}\}$
[ACALL-b]	$\Delta^{\gamma}_{r,s} \longmapsto \Delta^{\gamma}_{r,s}\ \overrightarrow{\top^{\beta}}$	$\beta \in \text{adj}(N^{\gamma}_{r,s+1})$
[ARET]	$\nabla^{\gamma}_{r,s},\ [\overleftarrow{\top^{\beta}} \longmapsto \nabla^{\gamma}_{r,s+1}$	$\beta \in \text{adj}(N^{\gamma}_{r,s+1})$
[FCALL-a]	$\nabla^{\beta}_{f,0} \longmapsto [\nabla^{\beta}_{f,0},\ \bot$	$N^{\beta}_{f,0} = \mathbf{F}^{\beta}$
[FCALL-b]	$\Delta^{\gamma}_{r,s}\ \bot \longmapsto \overrightarrow{N^{\gamma}_{r,s+1}}$	
[FRET]	$\nabla^{\beta}_{f,0},\ [\overleftarrow{N^{\gamma}_{r,s+1}} \longmapsto \nabla^{\beta}_{f,1}$	$N^{\beta}_{f,0} = \mathbf{F}^{\beta},\quad \beta \in \text{adj}(N^{\gamma}_{r,s+1})$

Tabla 6.4: Reglas del esquema de compilación genérico de TAG en EPDA

la pila de adjunciones pendientes, lo que permite utilizar transiciones UNWRAP en las reglas **[RET]** y **[SRET]**.

El esquema de compilación genérico no establece ninguna restricción sobre la estrategia utilizada para recorrer los árboles elementales. A continuación se definen, de acuerdo con la tabla 5.2, los esquemas de compilación correspondientes a tres estrategias particulares aplicadas al recorrido de los árboles elementales: ascendente, Earley y descendente.

6.6.1 Estrategia descendente

Esquema de compilación 6.3 El esquema de compilación de una gramática de adjunción de árboles en un autómata a pila embebido que incorpora una estrategia descendente para el recorrido de los árboles elementales queda definido por el conjunto de reglas mostrado en la tabla 6.5 y los elementos inicial $\$_0$ y final $\Box$. §

6.6.2 Estrategia Earley

Esquema de compilación 6.4 El esquema de compilación de una gramática de adjunción de árboles en un autómata a pila embebido que incorpora una estrategia Earley para el recorrido

[INIT]	$\$_0 \longmapsto \$_0,\ [\nabla^{\alpha}_{0,0}$	$\alpha \in \boldsymbol{I}$
[CALL]	$\nabla^{\gamma}_{r,s} \longmapsto \nabla^{\gamma}_{r,s},\ [N^{\gamma}_{r,s+1}$	$N^{\gamma}_{r,s+1} \notin \text{espina}(\gamma),\ \textbf{nil} \in \text{adj}(N^{\gamma}_{r,s+1})$
[SCALL]	$\nabla^{\beta}_{r,s} \longmapsto [\nabla^{\beta}_{r,s},\ \ N^{\beta}_{r,s+1}$	$N^{\beta}_{r,s+1} \in \text{espina}(\beta),\ \textbf{nil} \in \text{adj}(N^{\beta}_{r,s+1})$
[SEL]	$N^{\gamma}_{r,0} \longmapsto \nabla^{\gamma}_{r,0}$	
[PUB]	$\nabla^{\gamma}_{r,n_r} \longmapsto \square$	
[RET]	$\nabla^{\gamma}_{r,s},\ [\square \longmapsto \nabla^{\gamma}_{r,s+1}$	$N^{\gamma}_{r,s+1} \notin \text{espina}(\gamma),\ \textbf{nil} \in \text{adj}(N^{\gamma}_{r,s+1})$
[SRET]	$\nabla^{\beta}_{r,s},\ [\square \longmapsto \nabla^{\beta}_{r,s+1}$	$N^{\beta}_{r,s+1} \in \text{espina}(\beta),\ \textbf{nil} \in \text{adj}(N^{\beta}_{r,s+1})$
[SCAN]	$N^{\gamma}_{r,0} \overset{a}{\longmapsto} \square$	$N^{\gamma}_{r,0} \to a$
[ACALL-a]	$\nabla^{\gamma}_{r,s} \longmapsto [\nabla^{\gamma}_{r,s},\ \ \Delta^{\gamma}_{r,s}$	$\text{adj}(N^{\gamma}_{r,s+1}) \neq \{\textbf{nil}\}$
[ACALL-b]	$\Delta^{\gamma}_{r,s} \longmapsto \Delta^{\gamma}_{r,s}\ \ \top^{\beta}$	$\beta \in \text{adj}(N^{\gamma}_{r,s+1})$
[ARET]	$\nabla^{\gamma}_{r,s},\ [\square \longmapsto \nabla^{\gamma}_{r,s+1}$	
[FCALL-a]	$\nabla^{\beta}_{f,0} \longmapsto [\nabla^{\beta}_{f,0},\ \ \bot$	$N^{\beta}_{f,0} = \mathbf{F}^{\beta}$
[FCALL-b]	$\Delta^{\gamma}_{r,s}\ \ \bot \longmapsto N^{\gamma}_{r,s+1}$	
[FRET]	$\nabla^{\beta}_{f,0},\ [\square \longmapsto \nabla^{\beta}_{f,1}$	$N^{\beta}_{f,0} = \mathbf{F}^{\beta},\ \ \beta \in \text{adj}(N^{\gamma}_{r,s+1})$

Tabla 6.5: Reglas del esquema de compilación descendente de TAG en EPDA

[INIT]	$\$_0 \longmapsto \$_0,\ [\nabla^{\alpha}_{0,0}$	$\alpha \in \boldsymbol{I}$
[CALL]	$\nabla^{\gamma}_{r,s} \longmapsto \nabla^{\gamma}_{r,s},\ [\overline{N^{\gamma}_{r,s+1}}$	$N^{\gamma}_{r,s+1} \notin \text{espina}(\gamma),\ \textbf{nil} \in \text{adj}(N^{\gamma}_{r,s+1})$
[SCALL]	$\nabla^{\beta}_{r,s} \longmapsto [\nabla^{\beta}_{r,s},\ \ \overline{N^{\beta}_{r,s+1}}$	$N^{\beta}_{r,s+1} \in \text{espina}(\beta),\ \textbf{nil} \in \text{adj}(N^{\beta}_{r,s+1})$
[SEL]	$\overline{N^{\gamma}_{r,0}} \longmapsto \nabla^{\gamma}_{r,0}$	
[PUB]	$\nabla^{\gamma}_{r,n_r} \longmapsto \overline{\overline{N^{\gamma}_{r,0}}}$	
[RET]	$\nabla^{\gamma}_{r,s},\ [\overline{\overline{N^{\gamma}_{r,s+1}}} \longmapsto \nabla^{\gamma}_{r,s+1}$	$N^{\gamma}_{r,s+1} \notin \text{espina}(\gamma),\ \textbf{nil} \in \text{adj}(N^{\gamma}_{r,s+1})$
[SRET]	$\nabla^{\beta}_{r,s},\ [\overline{\overline{N^{\beta}_{r,s+1}}} \longmapsto \nabla^{\beta}_{r,s+1}$	$N^{\beta}_{r,s+1} \in \text{espina}(\beta),\ \textbf{nil} \in \text{adj}(N^{\beta}_{r,s+1})$
[SCAN]	$\overline{N^{\gamma}_{r,0}} \overset{a}{\longmapsto} \overline{\overline{N^{\gamma}_{r,0}}}$	$N^{\gamma}_{r,0} \to a$
[ACALL-a]	$\nabla^{\gamma}_{r,s} \longmapsto [\nabla^{\gamma}_{r,s},\ \ \Delta^{\gamma}_{r,s}$	$\text{adj}(N^{\gamma}_{r,s+1}) \neq \{\textbf{nil}\}$
[ACALL-b]	$\Delta^{\gamma}_{r,s} \longmapsto \Delta^{\gamma}_{r,s}\ \ \overline{\top^{\beta}}$	$\beta \in \text{adj}(N^{\gamma}_{r,s+1})$
[ARET]	$\nabla^{\gamma}_{r,s},\ [\overline{\overline{\top^{\beta}}} \longmapsto \nabla^{\gamma}_{r,s+1}$	$\beta \in \text{adj}(N^{\gamma}_{r,s+1})$
[FCALL-a]	$\nabla^{\beta}_{f,0} \longmapsto [\nabla^{\beta}_{f,0},\ \ \bot$	$N^{\beta}_{f,0} = \mathbf{F}^{\beta}$
[FCALL-b]	$\Delta^{\gamma}_{r,s}\ \ \bot \longmapsto \overline{N^{\gamma}_{r,s+1}}$	
[FRET]	$\nabla^{\beta}_{f,0},\ [\overline{\overline{N^{\gamma}_{r,s+1}}} \longmapsto \nabla^{\beta}_{f,1}$	$N^{\beta}_{f,0} = \mathbf{F}^{\beta},\ \ \beta \in \text{adj}(N^{\gamma}_{r,s+1})$

Tabla 6.6: Reglas del esquema de compilación Earley de TAG en EPDA

[INIT]	$\$_0 \longmapsto \$_0,\ \lbrack\nabla^{\alpha}_{0,0}$	$\alpha \in \boldsymbol{I}$
[CALL]	$\nabla^{\gamma}_{r,s} \longmapsto \nabla^{\gamma}_{r,s},\ \lbrack\Box$	$N^{\gamma}_{r,s+1} \not\in \text{espina}(\gamma),\ \ \mathbf{nil} \in \text{adj}(N^{\gamma}_{r,s+1})$
[SCALL]	$\nabla^{\beta}_{r,s} \longmapsto \lbrack\nabla^{\beta}_{r,s},\ \ \Box$	$N^{\beta}_{r,s+1} \in \text{espina}(\beta),\ \ \mathbf{nil} \in \text{adj}(N^{\beta}_{r,s+1})$
[SEL]	$\Box \longmapsto \nabla^{\gamma}_{r,0}$	
[PUB]	$\nabla^{\gamma}_{r,n_r} \longmapsto N^{\gamma}_{r,0}$	
[RET]	$\nabla^{\gamma}_{r,s},\ \lbrack N^{\gamma}_{r,s+1} \longmapsto \nabla^{\gamma}_{r,s+1}$	$N^{\gamma}_{r,s+1} \not\in \text{espina}(\gamma),\ \ \mathbf{nil} \in \text{adj}(N^{\gamma}_{r,s+1})$
[SRET]	$\nabla^{\beta}_{r,s},\ \lbrack N^{\beta}_{r,s+1} \longmapsto \nabla^{\beta}_{r,s+1}$	$N^{\beta}_{r,s+1} \in \text{espina}(\beta),\ \ \mathbf{nil} \in \text{adj}(N_{r,s+1})$
[SCAN]	$\Box \overset{a}{\longmapsto} N^{\gamma}_{r,0}$	$N^{\gamma}_{r,0} \to a$
[ACALL-a]	$\nabla^{\gamma}_{r,s} \longmapsto \lbrack\nabla^{\gamma}_{r,s},\ \ \Delta^{\gamma}_{r,s}$	$\text{adj}(N^{\gamma}_{r,s+1}) \neq \{\mathbf{nil}\}$
[ACALL-b]	$\Delta^{\gamma}_{r,s} \longmapsto \Delta^{\gamma}_{r,s}\ \ \Box$	
[ARET]	$\nabla^{\gamma}_{r,s},\ \lbrack\top^{\beta} \longmapsto \nabla^{\gamma}_{r,s+1}$	$\beta \in \text{adj}(N^{\gamma}_{r,s+1})$
[FCALL-a]	$\nabla^{\beta}_{f,0} \longmapsto \lbrack\nabla^{\beta}_{f,0},\ \ \bot$	$N^{\beta}_{f,0} = \mathbf{F}^{\beta}$
[FCALL-b]	$\Delta^{\gamma}_{r,s}\ \ \bot \longmapsto \Box$	
[FRET]	$\nabla^{\beta}_{f,0},\ \lbrack N^{\gamma}_{r,s+1} \longmapsto \nabla^{\beta}_{f,1}$	$N^{\beta}_{f,0} = \mathbf{F}^{\beta},\ \ \ \beta \in \text{adj}(N^{\gamma}_{r,s+1})$

Tabla 6.7: Reglas del esquema de compilación ascendente de TAG en EPDA

de los árboles elementales queda definido por el conjunto de reglas mostrado en la tabla 6.6 y los elementos inicial $\$_0$ y final $\overline{\overline{S}}$. §

6.6.3 Estrategia ascendente

Esquema de compilación 6.5 El esquema de compilación de una gramática de adjunción de árboles en un autómata a pila embebido que incorpora una estrategia ascendente para el recorrido de los árboles elementales queda definido por el conjunto de reglas mostrado en la tabla 6.7 y los elementos inicial $\$_0$ y final S. §

6.7 Esquemas de compilación de gramáticas lineales de índices

Para el análisis de gramáticas lineales de índices mediante autómatas a pila embebidos utilizaremos la semántica dada por Vijay-Shanker en [206] a las pilas del EPDA: una pila $\lbrack\alpha B$ se corresponde con el símbolo $B[\alpha]$ de una gramática lineal de índices. En consecuencia, para emular una derivación de una gramática lineal de índices en un EPDA es preciso ir modificando las pilas del autómata de acuerdo con los cambios en las pilas de índices indicados por la gramática. De ello se encargan las reglas de compilación descritas en la tabla 6.8.

La única novedad con respecto a las reglas utilizadas en el esquema de compilación 6.1 para gramáticas independientes del contexto radica en la introducción de la regla de compilación

Regla	Tarea
[INIT]	Inicia los cálculos a partir de la pila inicial.
[CALL]	Requiere el análisis de un determinado elemento gramatical $A[\ ]$ que no es un hijo dependiente, lo cual implica situar en la cima de la pila principal una nueva pila $\lbrack A$.
[SCALL]	Requiere el análisis de un determinado elemento gramatical que es un hijo dependiente, lo que implica tener que pasarle la pila de índices con los cambios correspondientes.
[SEL]	Selecciona una producción.
[PUB]	Determina que una producción de la gramática ha sido completamente analizada.
[RET]	Continúa el proceso de análisis después de que se haya completado una producción, lo cual implica eliminar una pila $\lbrack A$ de la cima de la pila principal.
[SCAN]	Reconoce los terminales que componen la cadena de entrada.

Tabla 6.8: Tipos de reglas de los esquemas de compilación de LIG en EPDA

[SCALL] utilizada para manejar las pilas de índices. Debido a las características propias de los autómatas a pila embebidos, las pilas de índices asociadas a los terminales se calculan de modo descendente, de tal forma que dada una producción $\boldsymbol{A} \rightarrow \Upsilon_1\ \boldsymbol{B}\ \Upsilon_2$, la pila de índices asociada a $\boldsymbol{B}$ se calcula a partir de la pila asociada a $\boldsymbol{A}$ en el momento de comenzar a analizar $\boldsymbol{B}$. Cuando se termine de analizar $\boldsymbol{B}$, su pila asociada será vacía. Quiere esto decir que todas las comprobaciones acerca de la buena formación de las pilas se chequean en al fase descendente del algoritmo y que durante la fase ascendente todas las pilas de índices que se propagan son pilas vacías.

Con respecto al tratamiento de los no-terminales de la gramática lineal de índices, los autómatas a pila embebidos no imponen ninguna restricción, circunstancia que aprovecharemos para definir un esquema de compilación para una estrategia genérica en la cual se ha parametrizado la información predicha en la fase descendente y la información propagada en la fase ascendente. Posteriormente definiremos los esquemas correspondientes a las estrategias descendente, Earley y ascendente.

Esquema de compilación 6.6 El esquema de compilación genérico de una gramática lineal de índices en un autómata a pila embebido queda definido por el conjunto de reglas mostrado en la tabla 6.9 y los elementos inicial $\$_0$ y final $\overleftarrow{S}$. Las reglas de compilación **[SCALL-2]** y **[SCALL-3]** podrían escribirse alternativamente de la forma:

$$\textbf{[SCALL-2]} \quad \nabla_{r,s} \longmapsto \lbrack \nabla_{r,s},\ \gamma'\ \overrightarrow{A_{r,s+1}} \qquad A_{r,0}[\circ\circ] \rightarrow \Upsilon_1 A_{r,s+1}[\circ\circ\gamma']\Upsilon_2$$

$$\textbf{[SCALL-3]} \quad \gamma\ \nabla_{r,s} \longmapsto \lbrack \nabla_{r,s},\ \overrightarrow{A_{r,s+1}} \qquad A_{r,0}[\circ\circ\gamma] \rightarrow \Upsilon_1 A_{r,s+1}[\circ\circ]\Upsilon_2$$

[INIT]	$\$_0 \longmapsto \$_0, [\nabla_{0,0}$	
[CALL]	$\nabla_{r,s} \longmapsto \nabla_{r,s},\ [\overrightarrow{A_{r,s+1}}$	$\boldsymbol{A_{r,0}} \to \Upsilon_1 A_{r,s+1}[\,]\Upsilon_2$
[SCALL-1]	$\nabla_{r,s} \longmapsto [\nabla_{r,s},\ \overrightarrow{A_{r,s+1}}$	$A_{r,0}[\circ\circ] \to \Upsilon_1 A_{r,s+1}[\circ\circ]\Upsilon_2$
[SCALL-2a]	$\nabla_{r,s} \longmapsto [\nabla_{r,s},\ \langle\gamma', r, s+1\rangle$	$A_{r,0}[\circ\circ] \to \Upsilon_1 A_{r,s+1}[\circ\circ\gamma']\Upsilon_2$
[SCALL-2b]	$\langle\gamma', r, s+1\rangle \longmapsto \langle\gamma', r, s+1\rangle\ \overrightarrow{A_{r,s+1}}$	
[SCALL-3a]	$\nabla_{r,s} \longmapsto [\nabla_{r,s},\ \nabla'_{r,s}$	$A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\circ\circ]\Upsilon_2$
[SCALL-3b]	$\langle\gamma, t, u\rangle\ \nabla'_{r,s} \longmapsto \overrightarrow{A_{r,s+1}}$	$A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\circ\circ]\Upsilon_2$
[SEL]	$\overrightarrow{A_{r,0}} \longmapsto \nabla_{r,0}$	$r \neq 0$
[PUB]	$\nabla_{r,n_r} \longmapsto \overleftarrow{A_{r,0}}$	
[RET]	$\nabla_{r,s},\ [\overleftarrow{A_{r,s+1}} \longmapsto \nabla_{r,s+1}$	$\boldsymbol{A_{r,0}} \to \Upsilon_1 \boldsymbol{A_{r,s+1}}\Upsilon_2$
[SCAN]	$\overrightarrow{A_{r,0}} \overset{a}{\longmapsto} \overleftarrow{A_{r,0}}$	$A_{r,0}[\,] \to a$

Tabla 6.9: Reglas del esquema de compilación genérico de LIG en EPDA

pero en tal caso las producciones involucradas no formarían parte de la familia de transiciones elementales de los EPDA. Esta circunstancia nos ha llevado a descomponer la regla de compilación **[SCALL-2]** en dos reglas **[SCALL-2a]** y **[SCALL-2b]** y a descomponer la regla **[SCALL-3]** en dos reglas **[SCALL-3a]** y **[SCALL-3b]**. Como un efecto colateral, las pilas del autómata tendrán la forma $[\alpha B$, donde B será un no-terminal de la gramática lineal de índices y α estará formado por una sucesión de triples $\langle\gamma, r, s\rangle$, donde γ es un índices mientras que r y s señalan una posición s en una producción r. La proyección del primer componente de dichos triples proporciona la pila de índices asociada a B. Los componentes r y s sólo se utilizan en la regla de compilación **[SCALL-2b]** mientras que son ignorados en **[SCALL-3b]**. §

6.7.1 Estrategia descendente

Esquema de compilación 6.7 El esquema de compilación con estrategia descendente de una gramática lineal de índices en un autómata a pila embebido queda definido por el conjunto de reglas de la tabla 6.10 y los elementos inicial $\$_0$ y final $\Box$. §

6.7.2 Estrategia Earley

Esquema de compilación 6.8 El esquema de compilación con estrategia Earley de una gramática lineal de índices en un autómata a pila embebido queda definido por el conjunto de reglas de la tabla 6.11 y los elementos inicial $\$_0$ y final $\overline{\overline{S}}$. §

[INIT]	$\$_0 \longmapsto \$_0, [\nabla_{0,0}$	
[CALL]	$\nabla_{r,s} \longmapsto \nabla_{r,s},\ [A_{r,s+1}$	$\boldsymbol{A_{r,0}} \to \Upsilon_1 A_{r,s+1}[\,]\Upsilon_2$
[SCALL-1]	$\nabla_{r,s} \longmapsto [\nabla_{r,s},\ \ A_{r,s+1}$	$A_{r,0}[\circ\circ] \to \Upsilon_1 A_{r,s+1}[\circ\circ]\Upsilon_2$
[SCALL-2a]	$\nabla_{r,s} \longmapsto [\nabla_{r,s},\ \ \langle\gamma', r, s+1\rangle$	$A_{r,0}[\circ\circ] \to \Upsilon_1 A_{r,s+1}[\circ\circ\gamma']\Upsilon_2$
[SCALL-2b]	$\langle\gamma', r, s+1\rangle \longmapsto \langle\gamma', r, s+1\rangle\ \ A_{r,s+1}$	
[SCALL-3a]	$\nabla_{r,s} \longmapsto [\nabla_{r,s},\ \ \nabla'_{r,s}$	$A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\circ\circ]\Upsilon_2$
[SCALL-3b]	$\langle\gamma, t, u\rangle\ \ \nabla'_{r,s} \longmapsto A_{r,s+1}$	$A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\circ\circ]\Upsilon_2$
[SEL]	$A_{r,0} \longmapsto \nabla_{r,0}$	$r \neq 0$
[PUB]	$\nabla_{r,n_r} \longmapsto \Box$	
[RET]	$\nabla_{r,s},\ [\Box \longmapsto \nabla_{r,s+1}$	$\boldsymbol{A_{r,0}} \to \Upsilon_1 \boldsymbol{A_{r,s+1}}\Upsilon_2$
[SCAN]	$A_{r,0} \overset{a}{\longmapsto} \Box$	$A_{r,0}[\,] \to a$

Tabla 6.10: Reglas del esquema de compilación descendente de LIG en EPDA

6.7.3 Estrategia ascendente

Esquema de compilación 6.9 El esquema de compilación con estrategia ascendente de una gramática lineal de índices en un autómata a pila embebido queda definido por el conjunto de reglas de la tabla 6.12 y los elementos inicial $\$_0$ y final S. §

6.8 Lenguajes de adjunción de árboles y EPDA

Vijay-Shanker estableció en [206] que la clase de los lenguajes aceptados por los EPDA era equivalente a la clase de los lenguajes de adjunción de árboles. Para ello se valió de una técnica que permite obtener una gramática de núcleo a partir de un EPDA y viceversa. En esta sección ofrecemos un modo distinto de establecer la equivalencia entre los lenguajes aceptados por los EPDA y los lenguajes de adjunción de árboles. Para ello haremos uso de los esquemas de compilación definidos previamente.

Teorema 6.5 *Los lenguajes de adjunción de árboles son un subconjunto de los lenguajes aceptados por la clase de los autómatas a pila embebidos.*

Demostración:

Por el esquema de compilación 6.2, a partir de cualquier TAG es posible construir un EPDA que acepta el lenguaje reconocido por dicha gramática. Análogamente, por el esquema de compilación 6.6, a partir de cualquier LIG es posible construir un EPDA que acepta el lenguaje reconocido por dicha gramática. □

Teorema 6.6 *La clase de los lenguajes aceptados por los EPDA es un subconjunto de los lenguajes de adjunción de árboles.*

[INIT]	$\$_0 \longmapsto \$_0, [\nabla_{0,0}$	
[CALL]	$\nabla_{r,s} \longmapsto \nabla_{r,s},\ [\overline{A_{r,s+1}}$	$\boldsymbol{A_{r,0}} \rightarrow \Upsilon_1 A_{r,s+1}[\,]\Upsilon_2$
[SCALL-1]	$\nabla_{r,s} \longmapsto [\nabla_{r,s},\ \ \overline{A_{r,s+1}}$	$A_{r,0}[\circ\circ] \rightarrow \Upsilon_1 A_{r,s+1}[\circ\circ]\Upsilon_2$
[SCALL-2a]	$\nabla_{r,s} \longmapsto [\nabla_{r,s},\ \ \langle\gamma', r, s+1\rangle$	$A_{r,0}[\circ\circ] \rightarrow \Upsilon_1 A_{r,s+1}[\circ\circ\gamma']\Upsilon_2$
[SCALL-2b]	$\langle\gamma', r, s+1\rangle \longmapsto \langle\gamma', r, s+1\rangle\ \ \overline{A_{r,s+1}}$	
[SCALL-3a]	$\nabla_{r,s} \longmapsto [\nabla_{r,s},\ \ \nabla'_{r,s}$	$A_{r,0}[\circ\circ\gamma] \rightarrow \Upsilon_1 A_{r,s+1}[\circ\circ]\Upsilon_2$
[SCALL-3b]	$\langle\gamma, t, u\rangle\ \ \nabla'_{r,s} \longmapsto \overline{A_{r,s+1}}$	$A_{r,0}[\circ\circ\gamma] \rightarrow \Upsilon_1 A_{r,s+1}[\circ\circ]\Upsilon_2$
[SEL]	$\overline{A_{r,0}} \longmapsto \nabla_{r,0}$	$r \neq 0$
[PUB]	$\nabla_{r,n_r} \longmapsto \overline{\overline{A_{r,0}}}$	
[RET]	$\nabla_{r,s},\ [\overline{\overline{A_{r,s+1}}} \longmapsto \nabla_{r,s+1}$	$\boldsymbol{A_{r,0}} \rightarrow \Upsilon_1 \boldsymbol{A_{r,s+1}}\Upsilon_2$
[SCAN]	$\overline{A_{r,0}} \overset{a}{\longmapsto} \overline{\overline{A_{r,0}}}$	$A_{r,0}[\,] \rightarrow a$

Tabla 6.11: Reglas del esquema de compilación Earley de LIG en EPDA

Demostración:

Mostraremos que para todo EPDA existe una gramática lineal de índices tal que el lenguaje reconocido por la gramática coincide con el lenguaje aceptado por el autómata.

Sea $\mathcal{A} = (V_T, V_S, \Theta, \$_0, \$_f)$ un autómata a pila embebido. Construiremos una gramática lineal de índices $\mathcal{L} = (V_T, V_N, V_I, S, P)$, donde $V_I = V_S$ y el conjunto V_N de no-terminales estará formado por pares $\langle A, B\rangle$ tales que $A, B \in V_S$. Para que $\mathcal{L}$ reconozca el lenguaje aceptado por $\mathcal{A}$ el conjunto de producciones en P ha de construirse a partir de las transiciones en Θ de la siguiente manera:

- Para toda transición $C \overset{a}{\longmapsto} F$ y para todo $E \in V_S$ creamos una producción
$$\langle C, E\rangle[\circ\circ] \rightarrow a\ \langle F, E\rangle[\circ\circ]$$
- Para toda transición $C \overset{a}{\longmapsto} CF$ y para todo $E \in V_S$ creamos una producción
$$\langle C, E\rangle[\circ\circ] \rightarrow a\ \langle F, E\rangle[\circ\circ C]$$
- Para toda transición $CF \overset{a}{\longmapsto} G$ y para todo $E \in V_S$ creamos una producción
$$\langle F, E\rangle[\circ\circ C] \rightarrow a\ \langle G, E\rangle[\circ\circ]$$
- Para todo par de transiciones $C, [F \overset{a}{\longmapsto} G$ y $C \overset{b}{\longmapsto} C, [F'$, y para todo $E \in V_S$ creamos una producción
$$\langle C, E\rangle[\circ\circ] \rightarrow b\ \langle F', F\rangle[\,]\ a\ \langle G, E\rangle[\circ\circ]$$
- Para todo par de transiciones $C, [F \overset{a}{\longmapsto} G$ y $C \overset{b}{\longmapsto} [C, F'$, y para todo $E \in V_S$ creamos una producción
$$\langle C, E\rangle[\circ\circ] \rightarrow b\ \langle F', F\rangle[\circ\circ]\ a\ \langle G, E\rangle[\,]$$
- Para todo $E \in V_S$ creamos una producción
$$\langle E, E\rangle[\,] \rightarrow \epsilon$$
- Para toda transición $\$_0 \overset{a}{\longmapsto} \$_0, [F$ o $\$_0 \overset{a}{\longmapsto} [\$_0, F$, donde $F \in V_S - \{\$_0\}$, creamos una producción
$$\langle \$_0, \$_0\rangle[\circ\circ] \rightarrow a\ \langle F, \$_f\rangle[\circ\circ]$$

[INIT]	$\$_0 \longmapsto \$_0, [\nabla_{0,0}$	
[CALL]	$\nabla_{r,s} \longmapsto \nabla_{r,s},\ [\Box$	$\boldsymbol{A_{r,0}} \to \Upsilon_1 A_{r,s+1}[\,]\Upsilon_2$
[SCALL-1]	$\nabla_{r,s} \longmapsto [\nabla_{r,s},\ \Box$	$A_{r,0}[\circ\circ] \to \Upsilon_1 A_{r,s+1}[\circ\circ]\Upsilon_2$
[SCALL-2a]	$\nabla_{r,s} \longmapsto [\nabla_{r,s},\ \langle\gamma', r, s+1\rangle$	$A_{r,0}[\circ\circ] \to \Upsilon_1 A_{r,s+1}[\circ\circ\gamma']\Upsilon_2$
[SCALL-2b]	$\langle\gamma', r, s+1\rangle \longmapsto \langle\gamma', r, s+1\rangle\ \Box$	
[SCALL-3a]	$\nabla_{r,s} \longmapsto [\nabla_{r,s},\ \nabla'_{r,s}$	$A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\circ\circ]\Upsilon_2$
[SCALL-3b]	$\langle\gamma, t, u\rangle\ \nabla'_{r,s} \longmapsto \Box$	$A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\circ\circ]\Upsilon_2$
[SEL]	$\Box \longmapsto \nabla_{r,0}$	$r \neq 0$
[PUB]	$\nabla_{r,n_r} \longmapsto A_{r,0}$	
[RET]	$\nabla_{r,s},\ [A_{r,s+1} \longmapsto \nabla_{r,s+1}$	$\boldsymbol{A_{r,0}} \to \Upsilon_1 \boldsymbol{A_{r,s+1}}\Upsilon_2$
[SCAN]	$\Box \overset{a}{\longmapsto} A_{r,0}$	$A_{r,0}[\,] \to a$

Tabla 6.12: Reglas del esquema de compilación ascendente de LIG en EPDA

- Para toda transición $B \overset{a}{\longmapsto} \$_f$ creamos una transición

$$\langle B, \$_f\rangle[\,] \to a$$

Con respecto al axioma de la gramática, tenemos que $S = \langle \$_0, \$_0\rangle$.

Es posible mostrar que $\langle C, E\rangle[\alpha] \overset{*}{\Rightarrow} w$ si y sólo si $([\alpha C, w) \overset{*}{\vdash} ([E, \epsilon)$, puesto que:

- Si una derivación $([\alpha C, w) \overset{*}{\vdash} ([E, \epsilon)$ es el resultado de aplicar la secuencia $t_1, \ldots, t_m$ de transiciones en Θ, entonces existe una secuencia $p_1, \ldots, p'_m$ de producciones en P tal que $\langle C, E\rangle[\alpha] \overset{*}{\Rightarrow} w$ resultado de aplicar $p_1, \ldots, p'_m$ reconoce w. La demostración se realiza por inducción en la longitud de la derivación del autómata. El caso base lo constituye la derivación $([E, \epsilon) \overset{0}{\vdash} ([E, \epsilon)$, para la que existe una transición $\langle E, E\rangle[\,] \to \epsilon$. Por hipótesis de inducción suponemos que la proposición se cumple para cualquier derivación del autómata de longitud m. En el paso de inducción verificamos que se cumple para cualquier posible derivación de longitud mayor que m:
  - Si $([\alpha C, aw) \vdash ([\alpha F, w) \overset{m}{\vdash} ([E, \epsilon)$, $\exists \langle C, E\rangle[\circ\circ] \to a\ \langle F, E\rangle[\circ\circ] \in P$, por hipótesis de inducción $\langle F, E\rangle[\alpha] \overset{*}{\Rightarrow} w$ y en consecuencia $\langle C, E\rangle[\alpha] \overset{*}{\Rightarrow} aw$.
  - Si $([\alpha C, aw) \vdash ([\alpha CF, w) \overset{m}{\vdash} ([E, \epsilon)$, $\exists \langle C, E\rangle[\circ\circ] \to a\ \langle F, E\rangle[\circ\circ C] \in P$, por hipótesis de inducción $\langle F, E\rangle[\alpha C] \overset{*}{\Rightarrow} w$ y en consecuencia $\langle C, E\rangle[\alpha] \overset{*}{\Rightarrow} aw$.
  - Si $([\alpha CF, aw) \vdash ([\alpha G, w) \overset{m}{\vdash} ([E, \epsilon)$, $\exists \langle F, E\rangle[\circ\circ C] \to a\ \langle G, E\rangle[\circ\circ] \in P$, por hipótesis de inducción $\langle G, E\rangle[\alpha] \overset{*}{\Rightarrow} w$ y en consecuencia $\langle F, E\rangle[\alpha C] \overset{*}{\Rightarrow} aw$.
  - Si $([\alpha C, bw_1aw_2) \vdash ([\alpha C[F', w_1aw_2) \overset{m_1}{\vdash} ([\alpha C[F, aw_2) \vdash ([\alpha G, w_2) \overset{m_2}{\vdash} ([E, \epsilon)$, $\exists \langle C, E\rangle[\circ\circ] \to b\ \langle F', F\rangle[\,]\ a\ \langle G, E\rangle[\circ\circ] \in P$, por hipótesis de inducción $\langle F', F\rangle[\,] \overset{*}{\Rightarrow} w_1$ y $\langle G, E\rangle[\alpha] \overset{*}{\Rightarrow} w_2$ y en consecuencia $\langle C, E\rangle[\alpha] \overset{*}{\Rightarrow} aw_1bw_2$.
  - Si $([\alpha C, bw_1aw_2) \vdash ([C[\alpha F', w_1aw_2) \overset{m_1}{\vdash} ([C[F, aw_2) \vdash ([G, w_2) \overset{m_2}{\vdash} ([E, \epsilon)$, $\exists \langle C, E\rangle[\circ\circ] \to b\ \langle F', F\rangle[\circ\circ]\ a\ \langle G, E\rangle[\,] \in P$, por hipótesis de inducción $\langle F', F\rangle[\alpha] \overset{*}{\Rightarrow} w_1$ y $\langle G, E\rangle[\,] \overset{*}{\Rightarrow} w_2$ y en consecuencia $\langle C, E\rangle[\alpha] \overset{*}{\Rightarrow} aw_1bw_2$.
- Si una derivación izquierda $\langle C, E\rangle[\alpha] \overset{*}{\Rightarrow} w$ reconoce la cadena w como resultado de aplicar la secuencia $p_1, \ldots, p_m$ de producciones en P, entonces existe una secuencia de

transiciones $t_1, \ldots, t'_m$ tal que la derivación $([\alpha C, w) \overset{*}{\vdash} ([E, \epsilon)$ es el resultado de aplicar la secuencia de transiciones $t_1, \ldots, t'_m$. La demostración se realiza por inducción en la longitud de la derivación de la gramática. El caso base lo constituye la derivación $\langle E, E\rangle[\ ] \Rightarrow \epsilon$, para la que existe una derivación $([E, \epsilon) \overset{0}{\vdash} ([E, \epsilon)$ en el autómata. Por hipótesis de inducción suponemos que la proposición se cumple para cualquier derivación del autómata de longitud m. En el paso de inducción verificamos que se cumple para cualquier posible derivación de longitud mayor que m:

- Si $\langle C, E\rangle[\alpha] \Rightarrow a\ \langle F, E\rangle[\alpha] \overset{m}{\Rightarrow} aw$, existe una transición $C \overset{a}{\longmapsto} F$, por hipótesis de inducción $([\alpha F, w) \overset{*}{\vdash} ([E, \epsilon)$ y en consecuencia $([\alpha C, aw) \overset{*}{\vdash} ([E, \epsilon)$.
- Si $\langle C, E\rangle[\alpha] \Rightarrow a\ \langle F, E\rangle[\alpha C] \overset{m}{\Rightarrow} aw$, existe una transición $C \overset{a}{\longmapsto} CF$, por hipótesis de inducción $([\alpha CF, w) \overset{*}{\vdash} ([E, \epsilon)$ y en consecuencia $([\alpha C, aw) \overset{*}{\vdash} ([E, \epsilon)$.
- Si $\langle F, E\rangle[\alpha C] \Rightarrow a\ \langle G, E\rangle[\alpha] \overset{m}{\Rightarrow} aw$, existe una transición $CF \overset{a}{\longmapsto} G$, por hipótesis de inducción $([\alpha G, w) \overset{*}{\vdash} ([E, \epsilon)$ y en consecuencia $([\alpha CF, aw) \overset{*}{\vdash} ([E, \epsilon)$.
- Si $\langle C, E\rangle[\alpha] \Rightarrow b\ \langle F', F\rangle[\ ]\ a\ \langle G, E\rangle[\alpha] \overset{m_1}{\Rightarrow} bw_1 a\ \langle G, E\rangle[\alpha] \overset{m_2}{\Rightarrow} aw_1 bw_2$, existe un par de transiciones $C \overset{b}{\longmapsto} C, [F'$ y $C, [F \overset{a}{\longmapsto} G$, por hipótesis de inducción $([F', w_1) \overset{*}{\vdash} ([F, \epsilon)$ y $([\alpha G, w_2) \overset{*}{\vdash} ([E, \epsilon)$ y en consecuencia $([\alpha C, aw_1 bw_2) \overset{*}{\vdash} ([E, \epsilon)$.
- Si $\langle C, E\rangle[\alpha] \Rightarrow b\ \langle F', F\rangle[\alpha]\ a\ \langle G, E\rangle[\ ] \overset{m_1}{\Rightarrow} bw_1 a\ \langle G, E\rangle[\ ] \overset{m_2}{\Rightarrow} aw_1 bw_2$, existe un par de transiciones $C \overset{b}{\longmapsto} [C, F'$ y $C, [F \overset{a}{\longmapsto} G$, por hipótesis de inducción $(\alpha F', w_1) \overset{*}{\vdash} ([F, \epsilon)$ y $([G, w_2) \overset{*}{\vdash} ([E, \epsilon)$ y en consecuencia $([\alpha C, aw_1 bw_2) \overset{*}{\vdash} ([E, \epsilon)$.

□

Ejemplo 6.4 El autómata a pila embebido $(V_T, V_S, \Theta, \$_0, \$_f)$, donde $V_T = \{a, b, c, d\}$ y $V_S = \{\$_0, \$_f, B, C, D, E\}$, del ejemplo 6.2 acepta el lenguaje $\{a^n b^n c^n d^n \mid n \geq 0\}$. A partir de dicho autómata construiremos una gramática lineal de índices $(V_T, V_S \times V_S, V_S, \langle -, \$_0\rangle, P)$. La tabla 6.13 muestra el conjunto de transiciones P, que ha sido obtenido a partir de las transiciones del autómata mostradas en la tabla 6.2. Para facilitar la lectura, hemos utilizado Γ para denotar cualquier posible elemento de V_S. La tabla 6.14 muestra la derivación de la cadena *aabbccdd* en esta gramática. La primera columna muestra la producción aplicada para obtener la forma sentencial de la segunda columna. ¶

Corolario 6.7 *La clase de los lenguajes aceptados por los autómatas a pila embebidos coincide con la clase de los lenguajes de adjunción de árboles.*

Demostración:

Inmediata a partir de los teoremas 6.5 y 6.5. □

6.9 Tabulación

La ejecución directa de un autómata a pila embebido puede tener complejidad de orden exponencial con respecto al tamaño de la cadena de entrada. Ello se debe a que en el caso de que

$$
\begin{array}{ll}
(a) & \langle \$_0, \$_0\rangle[\circ\circ] \to \langle D, \$_0\rangle[\circ\circ] \\
(c) & \langle B, \Gamma\rangle[\circ\circ] \to \langle D, \Gamma\rangle[\circ\circ B] \\
\\
(d) & \langle B, \Gamma\rangle[\circ\circ] \to \langle C, \Gamma\rangle[\circ\circ B] \\
(f) & \langle E, \Gamma\rangle[\circ\circ B] \to \langle C, \Gamma\rangle[\circ\circ] \\
\\
(g) & \langle C, \Gamma\rangle[\circ\circ] \to b \;\; \langle E, C\rangle[\circ\circ] \;\; c \;\; \langle C, \Gamma\rangle[\,] \\
\\
(h) & \langle D, \Gamma\rangle[\circ\circ] \to a \;\; \langle B, C\rangle[\circ\circ] \;\; d \;\; \langle D, \Gamma\rangle[\,] \\
(i) & \langle D, \Gamma\rangle[\circ\circ] \to a \;\; \langle B, D\rangle[\circ\circ] \;\; d \;\; \langle D, \Gamma\rangle[\,] \\
(j) & \langle D, \Gamma\rangle[\circ\circ] \to \langle \$_f, \Gamma\rangle[\circ\circ] \\
\\
(k) & \langle \$_0, \$_0\rangle \to \epsilon \\
(l) & \langle C, C\rangle \to \epsilon \\
(m) & \langle D, D\rangle \to \epsilon \\
(n) & \langle D, \$_f\rangle \to \epsilon
\end{array}
$$

Tabla 6.13: Producciones de la LIG derivada del EPDA que acepta $\{a^n b^n c^n d^n\}$

$$
\begin{array}{llcl}
(a) & \langle \$_0, \$_0\rangle[\,] & \Rightarrow & \langle \$_0, \$_0\rangle[\,] \;\; \langle D, \$_f\rangle[\,] \\
(k) & & \Rightarrow & \langle D, \$_f\rangle[\,] \\
(i) & & \Rightarrow & a \;\; \langle B, D\rangle[\,] \;\; d \;\; \langle D, \$_f\rangle[\,] \\
(c) & & \Rightarrow & a \;\; \langle D, D\rangle[B] \;\; d \;\; \langle D, \$_f\rangle[\,] \\
(h) & & \Rightarrow & aa \;\; \langle B, C\rangle[B] \;\; \langle D, D\rangle[\,] \;\; dd \;\; \langle D, \$_f\rangle[\,] \\
(d) & & \Rightarrow & aa \;\; \langle C, C\rangle[BB] \;\; \langle D, D\rangle[\,] \;\; dd \;\; \langle D, \$_f\rangle[\,] \\
(g) & & \Rightarrow & aab \;\; \langle E, C\rangle[BB] \;\; c \;\; \langle C, C\rangle[\,] \;\; \langle D, D\rangle[\,] \;\; dd \;\; \langle D, \$_f\rangle[\,] \\
(f) & & \Rightarrow & aab \;\; \langle C, C\rangle[B] \;\; c \;\; \langle C, C\rangle[\,] \;\; \langle D, D\rangle[\,] \;\; dd \;\; \langle d, \$_f\rangle[\,] \\
(g) & & \Rightarrow & aabb \;\; \langle E, C\rangle[B] \;\; cc \;\; \langle C, C\rangle[\,] \;\; \langle C, C\rangle[\,] \;\; \langle D, D\rangle[\,] \;\; dd \;\; \langle D, \$_f\rangle[\,] \\
(f) & & \Rightarrow & aabb \;\; \langle C, C\rangle[\,] \;\; cc \;\; \langle C, C\rangle[\,] \;\; \langle C, C\rangle[\,] \;\; \langle D, D\rangle[\,] \;\; dd \;\; \langle D, \$_f\rangle[\,] \\
(l) & & \Rightarrow & aabbcc \;\; \langle C, C\rangle[\,] \;\; \langle C, C\rangle[\,] \;\; \langle D, D\rangle[\,] \;\; dd \;\; \langle D, \$_f\rangle[\,] \\
(l) & & \Rightarrow & aabbcc \;\; \langle C, C\rangle[\,] \;\; \langle D, D\rangle[\,] \;\; dd \;\; \langle D, \$_f\rangle[\,] \\
(l) & & \Rightarrow & aabbcc \;\; \langle D, D\rangle[\,] \;\; dd \;\; \langle D, \$_f\rangle[\,] \\
(m) & & \Rightarrow & aabbccdd \;\; \langle D, \$_f\rangle[\,] \\
(n) & & \Rightarrow & aabbccdd
\end{array}
$$

Tabla 6.14: Derivación de la cadena *aabbccdd*

Transición	**Compilación de LIG**	**Compilación de TAG**
$C \overset{a}{\longmapsto} F$	**[SEL][PUB][SCAN]**	**[SEL][PUB][SCAN]**
$C \longmapsto C,\ [F$	**[INIT][CALL]**	**[INIT][CALL]**
$C \longmapsto [C,\ F$	**[SCALL-1]**	**[SCALL]**
$C,\ [F \longmapsto G$	**[RET]**	**[RET][SRET][ARET][FRET]**
$C \longmapsto [C,\ X\ F$	**[SCALL-2]**	**[ACALL]**
$X\ C \longmapsto [C,\ F$	**[SCALL-3]**	**[FCALL]**

Tabla 6.15: Tipos de transiciones de los EPDA

varias transiciones sean aplicables en un configuración dada, el contenido de la pila del autómata deberá replicarse y cada transición deberá aplicarse sobre una copia diferente. Para conseguir una complejidad polinomial deberemos evitar replicar la pila del autómata. Un modo de conseguirlo consiste en diseñar una técnica de tabulación que permita trabajar con representaciones condensadas de las configuraciones en lugar de con las configuraciones completas. Dichas representaciones condensadas se denominan ítems y se almacenan en una tabla, por lo que pueden ser reutilizadas y compartidas.

El diseño de una técnica de tabulación general para EPDA se presenta como una tarea complicada. Sin embargo, es posible diseñar una técnica de tabulación para el subconjunto de los EPDA que utilizan los siguientes tipos de transiciones:

- SWAP
- WRAP-A
- WRAP-B
- UNWRAP
- $C \longmapsto [C,\ X\ F$
- $X\ C \longmapsto [C,\ F$

Las transiciones $C \longmapsto [C,\ XF$ pueden verse como la actuación combinada de una transición WRAP-B y una transición PUSH. Las transiciones $X\ C \longmapsto [C,\ F$ pueden verse como la actuación combinada de una transición WRAP-B y una transición POP. En consecuencia, toda operación de apilamiento, que se realiza conjuntamente con una transición WRAP-B, deberá estar ligada con una transición WRAP-B en la que se realiza la extracción del elemento apilado. Es precisamente esta relación entre apilamientos y extracción de la pila la que nos permitirá diseñar una técnica de tabulación para EPDA.

El conjunto de transiciones seleccionado es suficiente para definir los esquemas de compilación para TAG y LIG de las secciones precedentes. La tabla 6.15 muestra los tipos de transiciones y su relación con las reglas de dichos esquemas de compilación. En esta tabla hemos considerado que las transiciones de tipo SWAP son las únicas que consumen terminales de la cadena de entrada. Este hecho no resta generalidad ya que, para cualquier transición, el efecto de la lectura de la la cadena de entrada se puede conseguir mediante la aplicación consecutiva de una transición WRAP-A, una transición SWAP y una transición UNWRAP.

El primer paso en la definición de los ítems consiste en definir los distintos tipos de derivaciones que se pueden dar. En concreto, consideraremos los tres tipos siguientes:

Derivaciones de llamada. Son aquellas derivaciones en las que una pila es trasmitida mediante transiciones de tipo WRAP-B, por lo que presentan la forma

$$\begin{aligned}(\Upsilon \llbracket \alpha A, a_{h+1} \dots a_n) &\overset{*}{\vdash} (\Upsilon \llbracket A \;\; \Upsilon_1 \llbracket \alpha X B, a_{i+1} \dots a_n) \\ &\overset{*}{\vdash} (\Upsilon \llbracket A \;\; \Upsilon_1 \llbracket \alpha X C, a_{j+1} \dots a_n)\end{aligned}$$

donde $A, B, C, X \in V_S$, $\alpha \in V_S^*$, $\Upsilon, \Upsilon_1 \in (\llbracket V_S^*)^*$ y no existe un par $(\llbracket \alpha X F, f) \neq (\llbracket \alpha X B, i)$ tal que

$$\begin{aligned}(\Upsilon \llbracket \alpha A, a_{h+1} \dots a_n) &\overset{*}{\vdash} (\Upsilon \llbracket A \;\; \Upsilon_1 \llbracket \alpha X F, a_{f+1} \dots a_n) \\ &\overset{*}{\vdash} (\Upsilon \llbracket A \;\; \Upsilon_1 \llbracket \alpha X B, a_{i+1} \dots a_n) \\ &\overset{*}{\vdash} (\Upsilon \llbracket A \;\; \Upsilon_1 \llbracket \alpha X C, a_{j+1} \dots a_n)\end{aligned}$$

Las dos ocurrencias de α se refieren a la misma pila en el sentido de que dicha pila es trasmitida sin cambios a través de la derivación. En este tipo de derivaciones se cumple que para cualquier $\Upsilon' \in (\llbracket V_S^*)^*$ y $\alpha' \in V_S^*$

$$\begin{aligned}(\Upsilon' \llbracket \alpha' A, a_{h+1} \dots a_n) &\overset{*}{\vdash} (\Upsilon' \llbracket A \;\; \Upsilon_1 \llbracket \alpha' X B, a_{i+1} \dots a_n) \\ &\overset{*}{\vdash} (\Upsilon' \llbracket A \;\; \Upsilon_1 \llbracket \alpha' X C, a_{j+1} \dots a_n)\end{aligned}$$

tal y como se indica en la figura 6.7. La independencia de la derivación con respecto a Υ y α permite que sean representadas por *ítems de llamada* de la forma

$$[A, h \mid B, i, X, C, j, X \mid -, -, -, -]$$

Derivaciones de retorno Son derivaciones resultado de aplicar transiciones UNWRAP, por lo que presentan la forma

$$\begin{aligned}(\Upsilon \llbracket \alpha A, a_{h+1} \dots a_n) &\overset{*}{\vdash} (\Upsilon \llbracket A \;\; \Upsilon_1 \llbracket \alpha X B, a_{i+1} \dots a_n) \\ &\overset{*}{\vdash} (\Upsilon \llbracket A \;\; \Upsilon_1 \llbracket B \;\; \Upsilon_2 \llbracket \alpha D, a_{p+1} \dots a_n) \\ &\overset{*}{\vdash} (\Upsilon \llbracket A \;\; \Upsilon_1 \llbracket B \;\; \Upsilon_2 \llbracket E, a_{q+1} \dots a_n) \\ &\overset{*}{\vdash} (\Upsilon \llbracket A \;\; \Upsilon_1 \llbracket C, a_{j+1} \dots a_n)\end{aligned}$$

donde $A, B, C, D, E, X \in V_S$, $\alpha \in V_S^*$, $\Upsilon, \Upsilon_1, \Upsilon_2 \in (\llbracket V_S^*)^*$, α es transmitida sin cambios a lo largo de la derivación y no existen $(\llbracket \alpha X F, f) \neq (\llbracket \alpha X B, i)$ ni $(\llbracket \alpha G, g) \neq (\llbracket \alpha D, p)$ tal que

$$\begin{aligned}(\Upsilon \llbracket \alpha A, a_{h+1} \dots a_n) &\overset{*}{\vdash} (\Upsilon \llbracket A \;\; \Upsilon_1 \llbracket \alpha X F, a_{f+1} \dots a_n) \\ &\overset{*}{\vdash} (\Upsilon \llbracket A \;\; \Upsilon_1 \llbracket \alpha X B, a_{i+1} \dots a_n) \\ &\overset{*}{\vdash} (\Upsilon \llbracket A \;\; \Upsilon_1 \llbracket B \;\; \Upsilon_2 \llbracket \alpha G, a_{g+1} \dots a_n) \\ &\overset{*}{\vdash} (\Upsilon \llbracket A \;\; \Upsilon_1 \llbracket B \;\; \Upsilon_2 \llbracket \alpha D, a_{p+1} \dots a_n) \\ &\overset{*}{\vdash} (\Upsilon \llbracket A \;\; \Upsilon_1 \llbracket B \;\; \Upsilon_2 \llbracket E, a_{q+1} \dots a_n) \\ &\overset{*}{\vdash} (\Upsilon \llbracket A \;\; \Upsilon_1 \llbracket C, a_{j+1} \dots a_n)\end{aligned}$$

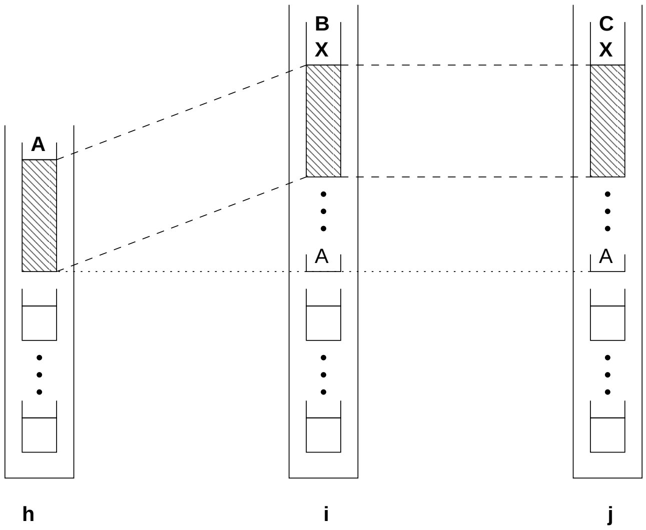

Figura 6.7: Derivaciones de llamada en EPDA

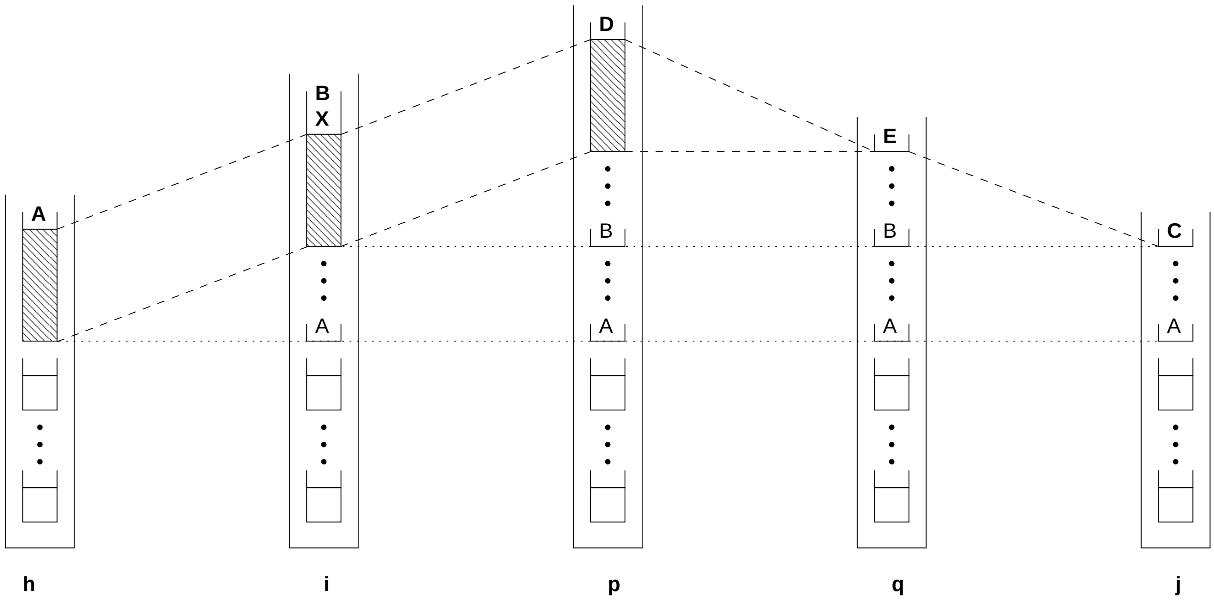

Figura 6.8: Derivaciones de retorno en EPDA

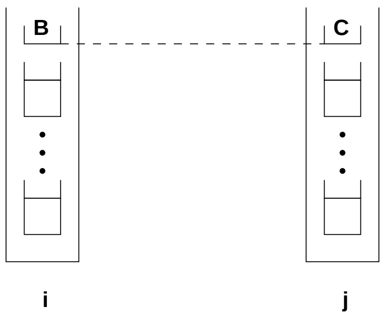

Figura 6.9: Derivaciones de puntos especiales en EPDA

En este tipo de derivaciones, para cualquier $\Upsilon' \in ([V_S^*)^*$ y $\alpha' \in V_S^*$ tal que existe una derivación $([\alpha' D, a_{p+1} \dots a_n) \overset{*}{\vdash} ([E, a_{q+1} \dots a_n)$, se cumple

$$\begin{aligned}(\Upsilon' \; [\alpha' A, a_{h+1} \dots a_n) \; & \overset{*}{\vdash} (\Upsilon \; [A \;\; \Upsilon_1 \; [\alpha' X B, a_{i+1} \dots a_n) \\ & \overset{*}{\vdash} (\Upsilon' \; [A \;\; \Upsilon_1 \; [B \;\; \Upsilon_2 \; [\alpha' D, a_{p+1} \dots a_n) \\ & \overset{*}{\vdash} (\Upsilon' \; [A \;\; \Upsilon_1 \; [B \;\; \Upsilon_2 \; [E, a_{q+1} \dots a_n) \\ & \overset{*}{\vdash} (\Upsilon' \; [A \;\; \Upsilon_1 \; [C, a_{j+1} \dots a_n)\end{aligned}$$

tal y como se indica en la figura 6.8. La independencia con respecto a Υ permite que este tipo de derivaciones sean representadas por *ítems de retorno* de la forma

$$[A, h \mid B, i, X, C, j, - \mid D, p, E, q]$$

en los que la parte (D, p, E, q) permite asegurar la relación entre $[\alpha D$ y $[\alpha A$.

Derivaciones de puntos especiales. Son aquellas derivaciones que sitúan una pila con un solo elemento en la cima del la pila principal, por lo que tienen la forma.

$$(\Upsilon \; [B, a_{i+1} \dots a_n) \overset{*}{\vdash} (\Upsilon \; [C, a_{j+1} \dots a_n)$$

donde $B, C \in V_S$ y $\Upsilon \in ([V_S^*)^*$ y no existe $([F, f) \neq ([B, i)$ tal que

$$(\Upsilon \; [F, a_{f+1} \dots a_n) \;\; \overset{*}{\vdash} (\Upsilon \; [B, a_{i+1} \dots a_n) \;\; \overset{*}{\vdash} (\Upsilon \; [C, a_{j+1} \dots a_n)$$

Para cualquier $\Upsilon' \in ([V_S^*)^*$ se cumple que

$$(\Upsilon' \; [B, a_{i+1} \dots a_n) \overset{*}{\vdash} (\Upsilon' \; [C, a_{j+1} \dots a_n)$$

tal y como se indica en la figura 6.9, por lo que pueden ser representadas de modo condensado por *ítems de puntos especiales* de la forma

$$[-, - \mid B, i, -, C, j, - \mid -, -, -, -]$$

Los diferentes ítems se combinan entre sí a partir del ítem inicial

$$[-, - \mid \$_0, 0, -, \$_0, 0, - \mid -, -, -, -]$$

mediante las reglas de combinación de las tablas 6.16 y 6.17. La aceptación de la cadena de entrada $a_1 \dots a_n$ por parte del autómata se indica mediante la presencia de ítems finales de la forma

$$[-, - \mid F, 0, -, \$_f, n, - \mid -, -, -, -]$$

tal que existe una transición $\$_0 \longmapsto [\$_0, F$ o una transición $\$_0 \longmapsto \$_0, [F$.

Teorema 6.8 *La manipulación de configuraciones mediante la aplicación de transiciones en los autómatas a pila embebidos es equivalente a la manipulación de ítems mediante las reglas de combinación de las tablas 6.16 y 6.17.*

$$\frac{[A,h \mid B,i,X,C,j,X \mid -,-,-,-]}{[A,h \mid B,i,X,F,k,X \mid -,-,-,-]} \quad C \overset{a}{\longmapsto} F,\ \ k=j \text{ si } a=\epsilon,\ \ k=j+1 \text{ si } a \in V_T$$

$$\frac{[A,h \mid B,i,X,C,j,- \mid D,p,E,q]}{[A,h \mid B,i,X,F,k,- \mid D,p,E,q]} \quad C \overset{a}{\longmapsto} F,\ \ k=j \text{ si } a=\epsilon,\ \ k=j+1 \text{ si } a \in V_T$$

$$\frac{[A,h \mid B,i,X,C,j,X \mid -,-,-,-]}{[-,- \mid F,j,-,F,j,- \mid -,-,-,-]} \quad C \longmapsto C, [F$$

$$\frac{[A,h \mid B,i,X,C,j,- \mid D,p,E,q]}{[-,- \mid F,j,-,F,j,- \mid -,-,-,-]} \quad C[\circ\circ] \longmapsto C[\circ\circ]\ F[\,]$$

$$\frac{[A,h \mid B,i,X,C,j,X \mid -,-,-,-]}{[A,h \mid F,j,X,F,j,X \mid -,-,-,-]} \quad C \longmapsto [C, F$$

$$\frac{[A,h \mid B,i,X,C,j,- \mid D,p,E,q]}{[-,- \mid F,j,-,F,j,- \mid -,-,-,-]} \quad C \longmapsto [C, F$$

$$\frac{[A,h \mid B,i,X,C,j,X \mid -,-,-,-]}{[C,j \mid F,j,X',F,j,X' \mid -,-,-,-]} \quad C \longmapsto [C, X'F$$

$$\frac{[A,h \mid B,i,X,C,j,- \mid D,p,E,q]}{[C,j \mid F,j,X',F,j,X' \mid -,-,-,-]} \quad C \longmapsto [C, X'F$$

$$\frac{\begin{array}{c}[A,h \mid B,i,X,C,j,X \mid -,-,-,-] \\ [M,m \mid N,t,X',A,h,X' \mid -,-,-,-]\end{array}}{[M,m \mid F,j,X',F,j,X' \mid -,-,-,-]} \quad XC \longmapsto [C, F$$

$$\frac{\begin{array}{c}[A,h \mid B,i,X,C,j,X \mid -,-,-,-] \\ [M,m \mid N,t,X',A,h,- \mid D,p,E,q]\end{array}}{[-,- \mid F,j,-,F,j,- \mid -,-,-,-]} \quad XC \longmapsto [C, F$$

Tabla 6.16: Combinación de ítems en EPDA (fase de llamada)

$$\frac{\begin{array}{c}[-,- \mid F',j,-,F,k,- \mid -,-,-,-] \\ [A,h \mid B,i,X,C,j,X \mid -,-,-,-]\end{array}}{[A,h \mid B,i,X,G,k,X \mid -,-,-,-]} \quad \begin{array}{l} C \longmapsto C,[F' \\ C,[F \longmapsto G \end{array}$$

$$\frac{\begin{array}{c}[-,- \mid F',j,-,F,k,- \mid -,-,-,-] \\ [A,h \mid B,i,X,C,j,- \mid D,p,E,q]\end{array}}{[A,h \mid B,i,X,G,k,- \mid D,p,E,q]} \quad \begin{array}{l} C \longmapsto C,[F' \\ C,[F \longmapsto G \end{array}$$

$$\frac{\begin{array}{c}[A,h \mid F',j,X,F,k,- \mid D,p,E,q] \\ [A,h \mid B,i,X,C,j,X \mid -,-,-,-]\end{array}}{[A,h \mid B,i,X,G,k,- \mid D,p,E,q]} \quad \begin{array}{l} C \longmapsto [C,F' \\ C,[F \longmapsto G \end{array}$$

$$\frac{\begin{array}{c}[-,- \mid F',j,-,F,k,- \mid -,-,-,-] \\ [A,h \mid B,i,X,C,j,- \mid D,p,E,q]\end{array}}{[A,h \mid B,i,X,G,k,- \mid D,p,E,q]} \quad \begin{array}{l} C \longmapsto [C,F' \\ C,[F \longmapsto G \end{array}$$

$$\frac{\begin{array}{c}[C,j \mid F',j,X',F,k,- \mid D,p,E,q] \\ [A,h \mid B,i,X,C,j,X \mid -,-,-,-] \\ [A,h \mid D,p,X,E,q,- \mid O,u,P,v]\end{array}}{[A,h \mid B,i,X,G,k,- \mid O,u,P,v]} \quad \begin{array}{l} C \longmapsto [C,X'F' \\ C,[F \longmapsto G \end{array}$$

$$\frac{\begin{array}{c}[C,j \mid F',j,X',F,k,- \mid O,u,P,v] \\ [A,h \mid B,i,X,C,j,- \mid D,p,E,q] \\ [-,- \mid O,u,-,P,v,- \mid -,-,-,-]\end{array}}{[A,h \mid B,i,X,G,k,- \mid D,p,E,q]} \quad \begin{array}{l} C \longmapsto [C,X'F' \\ C,[F \longmapsto G \end{array}$$

$$\frac{\begin{array}{c}[M,m \mid F',j,X',F,k,- \mid D,p,E,q] \\ [A,h \mid B,i,X,C,j,X \mid -,-,-,-] \\ [M,m \mid N,t,X',A,h,X' \mid -,-,-,-]\end{array}}{[A,h \mid B,i,X,G,k,- \mid F',j,F,k]} \quad \begin{array}{l} XC \longmapsto [C,F' \\ C,[F \longmapsto G \end{array}$$

$$\frac{\begin{array}{c}[-,- \mid F',j,-,F,k,- \mid -,-,-,-] \\ [A,h \mid B,i,X,C,j,X \mid -,-,-,-] \\ [M,m \mid N,t,X',A,h,- \mid D,p,E,q]\end{array}}{[A,h \mid B,i,X,G,k,- \mid F',j,F,k]} \quad \begin{array}{l} XC \longmapsto [C,F' \\ C,[F \longmapsto G \end{array}$$

Tabla 6.17: Combinación de ítems en EPDA (fase de retorno)

Demostración:

Debemos demostrar que para toda derivación existe un ítem que la representa y que para todo ítem existe una derivación a la que representa, teniendo en cuenta las diferentes formas en las que las derivaciones y los ítems pueden haber sido producidos. Con el fin de abreviar, realizaremos ambas demostraciones en paralelo. Para ello será necesario hacer una lista exhaustiva con todos los tipos de derivaciones que se pueden dar, mostrando para cada una de estas derivaciones la regla de combinación de ítems correspondiente. Dicha lista es la que se muestra a continuación.

- Derivaciones que son el resultado de aplicar una transición $C \overset{a}{\longmapsto} F$
  - a una derivación de llamada:

$$\begin{array}{rl} (\Upsilon \ [\alpha A, a_{h+1} \dots a_n) & \overset{*}{\vdash} (\Upsilon \ [A \ \Upsilon_1 \ [\alpha X B, a_{i+1} \dots a_n) \\ & \overset{*}{\vdash} (\Upsilon \ [A \ \Upsilon_1 \ [\alpha X C, a_{j+1} \dots a_n) \\ & \vdash (\Upsilon \ [A \ \Upsilon_1 \ [\alpha X F, a_{k+1} \dots a_n) \end{array}$$

$$\frac{[A, h \mid B, i, X, C, j, X \mid -, -, -, -]}{[A, h \mid B, i, X, F, k, X \mid -, -, -, -]} \quad C \overset{a}{\longmapsto} F, \ \ k = j \text{ si } a = \epsilon, \ \ k = j+1 \text{ si } a \in V_T$$

  - a una derivación de retorno:

$$\begin{array}{rl} (\Upsilon \ [\alpha A, a_{h+1} \dots a_n) & \overset{*}{\vdash} (\Upsilon \ [A \ \Upsilon_1 \ [\alpha X B, a_{i+1} \dots a_n) \\ & \overset{*}{\vdash} (\Upsilon \ [A \ \Upsilon_1 \ [B \ \Upsilon_2 \ [\alpha D, a_{p+1} \dots a_n) \\ & \overset{*}{\vdash} (\Upsilon \ [A \ \Upsilon_1 \ [B \ \Upsilon_2 \ [E, a_{q+1} \dots a_n) \\ & \overset{*}{\vdash} (\Upsilon \ [A \ \Upsilon_1 \ [C, a_{j+1} \dots a_n) \\ & \vdash (\Upsilon \ [A \ \Upsilon_1 \ [F, a_{k+1} \dots a_n) \end{array}$$

$$\frac{[A, h \mid B, i, X, C, j, - \mid D, p, E, q]}{[A, h \mid B, i, X, F, k, - \mid D, p, E, q]} \quad C \overset{a}{\longmapsto} F, \ \ k = j \text{ si } a = \epsilon, \ \ k = j+1 \text{ si } a \in V_T$$

  - a una derivación de puntos especiales:

$$\begin{array}{rl} (\Upsilon \ [B, a_{i+1} \dots a_n) & \overset{*}{\vdash} (\Upsilon \ [C, a_{j+1} \dots a_n) \\ & \vdash (\Upsilon \ [F, a_{k+1} \dots a_n) \end{array}$$

$$\frac{[-, - \mid B, i, -, C, j, - \mid -, -, -, -]}{[-, - \mid B, i, -, F, k, - \mid -, -, -, -]} \quad C \overset{a}{\longmapsto} F, \ \ k = j \text{ si } a = \epsilon, \ \ k = j+1 \text{ si } a \in V_T$$

- Derivaciones que son el resultado de aplicar una transición $C \longmapsto C, [F$
  - a una derivación de llamada:

$$\begin{array}{rl} (\Upsilon \ [\alpha A, a_{h+1} \dots a_n) & \overset{*}{\vdash} (\Upsilon \ [A \ \Upsilon_1 \ [\alpha X B, a_{i+1} \dots a_n) \\ & \overset{*}{\vdash} (\Upsilon \ [A \ \Upsilon_1 \ [\alpha X C, a_{j+1} \dots a_n) \\ & \vdash (\Upsilon \ [A \ \Upsilon_1 \ [\alpha X C \ F, a_{j+1} \dots a_n) \end{array}$$

$$\frac{[A, h \mid B, i, X, C, j, X \mid -, -, -, -]}{[-, - \mid F, j, -, F, j, - \mid -, -, -, -]} \quad C \longmapsto C, [F$$

  - a una derivación de retorno:

$$\begin{array}{rl} (\Upsilon \ [A[\delta], a_{h+1} \dots a_n) & \overset{*}{\vdash} (\Upsilon \ [A \ \Upsilon_1 \ [\alpha X B, a_{i+1} \dots a_n) \\ & \overset{*}{\vdash} (\Upsilon \ [A \ \Upsilon_1 \ [B \ \Upsilon_2 \ [\alpha D, a_{p+1} \dots a_n) \\ & \overset{*}{\vdash} (\Upsilon \ [A \ \Upsilon_1 \ [B \ \Upsilon_2 \ [E, a_{q+1} \dots a_n) \\ & \overset{*}{\vdash} (\Upsilon \ [A \ \Upsilon_1 \ [C, a_{j+1} \dots a_n) \\ & \vdash (\Upsilon \ [A \ \Upsilon_1 \ [C \ [F, a_{j+1} \dots a_n) \end{array}$$

$$\frac{[A, h \mid B, i, X, C, j, - \mid D, p, E, q]}{[-, - \mid F, j, -, F, j, - \mid -, -, -, -]} \quad C \longmapsto C, [F$$

- a una derivación de puntos especiales:

$$\begin{array}{ll} (\Upsilon\ [B, a_{i+1} \dots a_n) & \overset{*}{\vdash} (\Upsilon\ [C, a_{j+1} \dots a_n) \\ & \vdash (\Upsilon\ [C\ [F, a_{j+1} \dots a_n) \end{array}$$

$$\frac{[-,- \mid B,i,-,C,j,- \mid -,-,-,-]}{[-,- \mid F,j,-,F,j,- \mid -,-,-,-]}\ C \longmapsto C, [F$$

- Derivaciones que son el resultado de aplicar una transición $C \longmapsto [C, F$
  - a una derivación de llamada:

$$\begin{array}{ll} (\Upsilon\ [\alpha A, a_{h+1} \dots a_n) & \overset{*}{\vdash} (\Upsilon\ [A \\ ;\Upsilon_1\ [\alpha X B, a_{i+1} \dots a_n) & \\ & \overset{*}{\vdash} (\Upsilon\ [A\ \Upsilon_1\ [\alpha X C, a_{j+1} \dots a_n) \\ & \vdash (\Upsilon\ [A\ \Upsilon_1\ [C[\,]\ [\alpha X F, a_{j+1} \dots a_n) \end{array}$$

$$\frac{[A,h \mid B,i,X,C,j,X \mid -,-,-,-]}{[A,h \mid F,j,X,F,j,X \mid -,-,-,-]}\ C \longmapsto [C, F$$

  - a una derivación de retorno:

$$\begin{array}{ll} (\Upsilon\ [\alpha A, a_{h+1} \dots a_n) & \overset{*}{\vdash} (\Upsilon\ [A\ \Upsilon_1\ [\alpha X B, a_{i+1} \dots a_n) \\ & \overset{*}{\vdash} (\Upsilon\ [A\ \Upsilon_1\ [B\ \Upsilon_2\ [\alpha D, a_{p+1} \dots a_n) \\ & \overset{*}{\vdash} (\Upsilon\ [A\ \Upsilon_1\ [B\ \Upsilon_2\ [E, a_{q+1} \dots a_n) \\ & \overset{*}{\vdash} (\Upsilon\ [A\ \Upsilon_1\ [C, a_{j+1} \dots a_n) \\ & \vdash (\Upsilon\ [A\ \Upsilon_1\ [C\ [F, a_{j+1} \dots a_n) \end{array}$$

$$\frac{[A,h \mid B,i,X,C,j,- \mid D,p,E,q]}{[-,- \mid F,j,-,F,j,- \mid -,-,-,-]}\ C \longmapsto [C, F$$

  - a una derivación de puntos especiales:

$$\begin{array}{ll} (\Upsilon\ [B, a_{i+1} \dots a_n) & \overset{*}{\vdash} (\Upsilon\ [C, a_{j+1} \dots a_n) \\ & \vdash (\Upsilon\ [C\ [F, a_{j+1} \dots a_n) \end{array}$$

$$\frac{[-,- \mid B,i,-,C,j,- \mid -,-,-,-]}{[-,- \mid F,j,-,F,j,- \mid -,-,-,-]}\ C \longmapsto [C, F$$

- Derivaciones que son el resultado de aplicar una transición $C \longmapsto [C, X'F$
  - a una derivación de llamada:

$$\begin{array}{ll} (\Upsilon\ [\alpha A, a_{h+1} \dots a_n) & \overset{*}{\vdash} (\Upsilon\ [A\ \Upsilon_1\ [\alpha X B, a_{i+1} \dots a_n) \\ & \overset{*}{\vdash} (\Upsilon\ [A\ \Upsilon_1\ [\alpha X C, a_{j+1} \dots a_n) \\ & \vdash (\Upsilon\ [A\ \Upsilon_1\ [C\ [\alpha X X' F, a_{j+1} \dots a_n) \end{array}$$

$$\frac{[A,h \mid B,i,X,C,j,X \mid -,-,-,-]}{[C,j \mid F,j,X',F,j,X' \mid -,-,-,-]}\ C \longmapsto [C, X'F$$

  - a una derivación de retorno:

$$\begin{array}{ll} (\Upsilon\ [\alpha A, a_{h+1} \dots a_n) & \overset{*}{\vdash} (\Upsilon\ [A\ \Upsilon_1\ [\alpha X B, a_{i+1} \dots a_n) \\ & \overset{*}{\vdash} (\Upsilon\ [A\ \Upsilon_1\ [B\ \Upsilon_2\ [\alpha D, a_{p+1} \dots a_n) \\ & \overset{*}{\vdash} (\Upsilon\ [A\ \Upsilon_1\ [B\ \Upsilon_2\ [E, a_{q+1} \dots a_n) \\ & \overset{*}{\vdash} (\Upsilon\ [A\ \Upsilon_1\ [C, a_{j+1} \dots a_n) \\ & \vdash (\Upsilon\ [A\ \Upsilon_1\ [C\ [X'F, a_{j+1} \dots a_n) \end{array}$$

$$\frac{[A,h \mid B,i,X,C,j,- \mid D,p,E,q]}{[C,j \mid F,j,X',F,j,X' \mid -,-,-,-]}\ C \longmapsto [C, X'F$$

– a una derivación de puntos especiales:

$$\begin{array}{ll}(\Upsilon\ [B, a_{i+1}\dots a_n) & \overset{*}{\vdash} (\Upsilon\ [C, a_{j+1}\dots a_n) \\ & \vdash (\Upsilon\ [C\ [X'F, a_{j+1}\dots a_n)\end{array}$$

$$\frac{[-,-\mid B,i,-,C,j,-\mid -,-,-,-]}{[C,j\mid F,j,X',F,j,X'\mid -,-,-,-]}\ C\longmapsto [C,X'F$$

- Derivaciones que son el resultado de aplicar una transición $XC \longmapsto [C, F$ a una derivación de llamada, con los tres casos siguientes:
  - la derivación de llamada es a su vez obtenida a partir de una derivación de llamada:

$$\begin{array}{ll}(\Upsilon\ [\alpha M, a_{m+1}\dots a_n) & \overset{*}{\vdash} (\Upsilon\ [M\ \Upsilon_1\ [\alpha X'N, a_{t+1}\dots a_n) \\ & \overset{*}{\vdash} (\Upsilon\ [M\ \Upsilon_1\ [\alpha X'A, a_{h+1}\dots a_n) \\ & \overset{*}{\vdash} (\Upsilon\ [M\ \Upsilon_1\ [A\ \Upsilon_1\ [\alpha X'XB, a_{i+1}\dots a_n) \\ & \overset{*}{\vdash} (\Upsilon\ [M\ \Upsilon_1\ [A\ \Upsilon_1\ [\alpha X'XC, a_{j+1}\dots a_n) \\ & \vdash (\Upsilon\ [M\ \Upsilon_1\ [A\ \Upsilon_1\ [C\ [\alpha X'F, a_{j+1}\dots a_n)\end{array}$$

$$\frac{\begin{array}{c}[A,h\mid B,i,X,C,j,X\mid -,-,-,-]\\ [M,m\mid N,t,X',A,h,X'\mid -,-,-,-]\end{array}}{[M,m\mid F,j,X',F,j,X'\mid -,-,-,-]}\ XC\longmapsto [C,F$$

  - la derivación de llamada es a su vez obtenida a partir de una derivación de retorno:

$$\begin{array}{ll}(\Upsilon\ [\alpha M, a_{m+1}\dots a_n) & \overset{*}{\vdash} (\Upsilon\ [M\ \Upsilon_1\ [\alpha X'N, a_{t+1}\dots a_n) \\ & \overset{*}{\vdash} (\Upsilon\ [M\ \Upsilon_1\ [N\ \Upsilon_2\ [\alpha D, a_{p+1}\dots a_n) \\ & \overset{*}{\vdash} (\Upsilon\ [M\ \Upsilon_1\ [N\ \Upsilon_2\ [E, a_{q+1}\dots a_n) \\ & \overset{*}{\vdash} (\Upsilon\ [M\ \Upsilon_1\ [A, a_{h+1}\dots a_n) \\ & \overset{*}{\vdash} (\Upsilon\ [M\ \Upsilon_1\ [A\ [XB, a_{i+1}\dots a_n) \\ & \overset{*}{\vdash} (\Upsilon\ [M\ \Upsilon_1\ [A\ [XC, a_{j+1}\dots a_n) \\ & \vdash (\Upsilon\ [M\ \Upsilon_1\ [A\ [C\ [F, a_{j+1}\dots a_n)\end{array}$$

$$\frac{\begin{array}{c}[A,h\mid B,i,X,C,j,X\mid -,-,-,-]\\ [M,m\mid N,t,X',A,h,-\mid D,p,E,q]\end{array}}{[-,-\mid F,j,-,F,j,-\mid -,-,-,-]}\ XC\longmapsto [C,F$$

  - la derivación de llamada es a su vez obtenida a partir de una derivación de puntos especiales:

$$\begin{array}{ll}(\Upsilon\ [N, a_{t+1}\dots a_n) & \overset{*}{\vdash} (\Upsilon\ [A, a_{h+1}\dots a_n) \\ & \overset{*}{\vdash} (\Upsilon\ [A\ [XB, a_{i+1}\dots a_n) \\ & \overset{*}{\vdash} (\Upsilon\ [A\ [XC, a_{j+1}\dots a_n) \\ & \vdash (\Upsilon\ [A\ [C\ [F, a_{j+1}\dots a_n)\end{array}$$

$$\frac{\begin{array}{c}[A,h\mid B,i,X,C,j,X\mid -,-,-,-]\\ [-,-\mid N,t,-,A,h,-\mid -,-,-,-]\end{array}}{[-,-\mid F,j,-,F,j,-\mid -,-,-,-]}\ XC\longmapsto [C,F$$

- Derivaciones que son el resultado de aplicar una transición $C, [F \longmapsto G$ a una derivación obtenida tras aplicar una transición $C \longmapsto C, [F'$
  - a una derivación de llamada:

$$\begin{array}{ll}(\Upsilon\ [\alpha A, a_{h+1}\dots a_n) & \overset{*}{\vdash} (\Upsilon\ [A\ \Upsilon_1\ [\alpha XB, a_{i+1}\dots a_n) \\ & \overset{*}{\vdash} (\Upsilon\ [A\ \Upsilon_1\ [\alpha XC, a_{j+1}\dots a_n) \\ & \vdash (\Upsilon\ [A\ \Upsilon_1\ [\alpha XC\ [F', a_{j+1}\dots a_n) \\ & \overset{*}{\vdash} (\Upsilon\ [A\ \Upsilon_1\ [\alpha XC\ [F, a_{k+1}\dots a_n) \\ & \vdash (\Upsilon\ [A\ \Upsilon_1\ [\alpha XG, a_{k+1}\dots a_n)\end{array}$$

$$\frac{\begin{array}{c}[-,- \mid F',j,-,F,k,- \mid -,-,-,-] \\ [A,h \mid B,i,X,C,j,X \mid -,-,-,-]\end{array}}{[A,h \mid B,i,X,G,k,X \mid -,-,-,-]} \quad \begin{array}{l} C \longmapsto C,[F' \\ C,[F \longmapsto G \end{array}$$

– a una derivación de retorno:

$$\begin{array}{rl}(\Upsilon\ [\alpha A, a_{h+1}\dots a_n) & \overset{*}{\vdash} (\Upsilon\ [A\ \Upsilon_1\ [\alpha X B, a_{i+1}\dots a_n) \\ & \overset{*}{\vdash} (\Upsilon\ [A\ \Upsilon_1\ [B\ \Upsilon_2\ [\alpha D, a_{p+1}\dots a_n) \\ & \overset{*}{\vdash} (\Upsilon\ [A\ \Upsilon_1\ [B\ \Upsilon_2\ [E, a_{q+1}\dots a_n) \\ & \overset{*}{\vdash} (\Upsilon\ [A\ \Upsilon_1\ [C, a_{j+1}\dots a_n) \\ & \vdash (\Upsilon\ [A\ \Upsilon_1\ [C\ [F', a_{j+1}\dots a_n) \\ & \overset{*}{\vdash} (\Upsilon\ [A\ \Upsilon_1\ [C\ [F, a_{k+1}\dots a_n) \\ & \vdash (\Upsilon\ [A\ \Upsilon_1\ [G, a_{k+1}\dots a_n)\end{array}$$

$$\frac{\begin{array}{c}[-,- \mid F',j,-,F,k,- \mid -,-,-,-] \\ [A,h \mid B,i,X,C,j,- \mid D,p,E,q]\end{array}}{[A,h \mid B,i,X,G,k,- \mid D,p,E,q]} \quad \begin{array}{l} C \longmapsto C,[F' \\ C,[F \longmapsto G \end{array}$$

– a una derivación de puntos especiales:

$$\begin{array}{rl}(\Upsilon\ [B, a_{i+1}\dots a_n) & \overset{*}{\vdash} (\Upsilon\ [C, a_{j+1}\dots a_n) \\ & \vdash (\Upsilon\ [C\ [F', a_{j+1}\dots a_n) \\ & \overset{*}{\vdash} (\Upsilon\ [C\ [F, a_{k+1}\dots a_n) \\ & \vdash (\Upsilon\ [G, a_{k+1}\dots a_n)\end{array}$$

$$\frac{\begin{array}{c}[-,- \mid F',j,-,F,k,- \mid -,-,-,-] \\ [-,- \mid B,i,X,C,j,- \mid -,-,-,-]\end{array}}{[-,- \mid B,i,X,G,k,- \mid -,-,-,-]} \quad \begin{array}{l} C \longmapsto C,[F' \\ C,[F \longmapsto G \end{array}$$

- Derivaciones que son el resultado de aplicar una transición $C,[F \longmapsto G$ a una derivación obtenida tras aplicar una transición $C \longmapsto [C,F'$
  – a una derivación de llamada:

$$\begin{array}{rl}(\Upsilon\ [\alpha A, a_{h+1}\dots a_n) & \overset{*}{\vdash} (\Upsilon\ [A\ \Upsilon_1\ [\alpha X B, a_{i+1}\dots a_n) \\ & \overset{*}{\vdash} (\Upsilon\ [A\ \Upsilon_1\ [\alpha X C, a_{j+1}\dots a_n) \\ & \vdash (\Upsilon\ [A\ \Upsilon_1\ [C\ [\alpha X F', a_{j+1}\dots a_n) \\ & \overset{*}{\vdash} (\Upsilon\ [A\ \Upsilon_1\ [C\ [F'\ \Upsilon_2\ [\alpha D, a_{p+1}\dots a_n) \\ & \overset{*}{\vdash} (\Upsilon\ [A\ \Upsilon_1\ [C\ [F'\ \Upsilon_2\ [E, a_{q+1}\dots a_n) \\ & \overset{*}{\vdash} (\Upsilon\ [A\ \Upsilon_1\ [C\ [F, a_{k+1}\dots a_n) \\ & \vdash (\Upsilon\ [A\ \Upsilon_1\ [G, a_{k+1}\dots a_n)\end{array}$$

$$\frac{\begin{array}{c}[A,h \mid F',j,X,F,k,- \mid D,p,E,q] \\ [A,h \mid B,i,X,C,j,X \mid -,-,-,-]\end{array}}{[A,h \mid B,i,X,G,k,- \mid D,p,E,q]} \quad \begin{array}{l} C \longmapsto [C,F' \\ C,[F \longmapsto G \end{array}$$

  – a una derivación de retorno:

$$\begin{array}{rl}(\Upsilon\ [\alpha A, a_{h+1}\dots a_n) & \overset{*}{\vdash} (\Upsilon\ [A\ \Upsilon_1\ [\alpha X B, a_{i+1}\dots a_n) \\ & \overset{*}{\vdash} (\Upsilon\ [A\ \Upsilon_1\ [B\ \Upsilon_2\ [\alpha D, a_{p+1}\dots a_n) \\ & \overset{*}{\vdash} (\Upsilon\ [A\ \Upsilon_1\ [C\ \Upsilon_2\ [E, a_{q+1}\dots a_n) \\ & \overset{*}{\vdash} (\Upsilon\ [A\ \Upsilon_1\ [C, a_{j+1}\dots a_n) \\ & \vdash (\Upsilon\ [A\ \Upsilon_1\ [C\ [F', a_{j+1}\dots a_n) \\ & \overset{*}{\vdash} (\Upsilon\ [A\ \Upsilon_1\ [C\ [F, a_{k+1}\dots a_n) \\ & \vdash (\Upsilon\ [A\ \Upsilon_1\ [G, a_{k+1}\dots a_n)\end{array}$$

$$\frac{\begin{array}{c}[-,- \mid F',j,-,F,k,- \mid -,-,-,-]\\ [A,h \mid B,i,X,C,j,- \mid D,p,E,q]\end{array}}{[A,h \mid B,i,X,G,k,- \mid D,p,E,q]} \quad \begin{array}{l} C \longmapsto [C,F' \\ C,[F \longmapsto G\end{array}$$

– a una derivación de puntos especiales:

$$\begin{array}{ll}(\Upsilon\ [B,a_{i+1}\dots a_n) & \stackrel{*}{\vdash} (\Upsilon\ [C,a_{j+1}\dots a_n)\\ & \vdash (\Upsilon\ [C\ [F',a_{j+1}\dots a_n)\\ & \stackrel{*}{\vdash} (\Upsilon\ [C\ [F,a_{k+1}\dots a_n)\\ & \vdash (\Upsilon\ [G,a_{k+1}\dots a_n)\end{array}$$

$$\frac{\begin{array}{c}[-,- \mid F',j,-,F,k,- \mid -,-,-,-]\\ [-,- \mid B,i,-,C,j,- \mid -,-,-,-]\end{array}}{[-,- \mid B,i,-,G,k,- \mid -,-,-,-]} \quad \begin{array}{l} C \longmapsto [C,F' \\ C,[F \longmapsto G\end{array}$$

- Derivaciones que son el resultado de aplicar una transición $C,[F \longmapsto G$ a una derivación obtenida tras aplicar una transición $C \longmapsto [C,X'F'$

– a una derivación de llamada:

$$\begin{array}{ll}(\Upsilon\ [\alpha A,a_{h+1}\dots a_n) & \stackrel{*}{\vdash} (\Upsilon\ [A\ \Upsilon_1\ [\alpha XB,a_{i+1}\dots a_n)\\ & \stackrel{*}{\vdash} (\Upsilon\ [A\ \Upsilon_1\ [\alpha XC,a_{j+1}\dots a_n)\\ & \vdash (\Upsilon\ [A\ \Upsilon_1\ [C\ [\alpha XX'F',a_{j+1}\dots a_n)\\ & \stackrel{*}{\vdash} (\Upsilon\ [A\ \Upsilon_1\ [C\ [F'\ \Upsilon_2\ [\alpha XD,a_{p+1}\dots a_n)\\ & \stackrel{*}{\vdash} (\Upsilon\ [A\ \Upsilon_1\ [C\ [F'\ \Upsilon_2\ [D\ \Upsilon_3\ [\alpha O,a_{u+1}\dots a_n)\\ & \stackrel{*}{\vdash} (\Upsilon\ [A\ \Upsilon_1\ [C\ [F'\ \Upsilon_2\ [D\ \Upsilon_3\ [P,a_{v+1}\dots a_n)\\ & \stackrel{*}{\vdash} (\Upsilon\ [A\ \Upsilon_1\ [C\ [F'\ \Upsilon_2\ [E,a_{q+1}\dots a_n)\\ & \stackrel{*}{\vdash} (\Upsilon\ [A\ \Upsilon_1\ [C\ [F,a_{k+1}\dots a_n)\\ & \vdash (\Upsilon\ [A\ \Upsilon_1\ [G,a_{k+1}\dots a_n)\end{array}$$

$$\frac{\begin{array}{c}[C,j \mid F',j,X',F,k,- \mid D,p,E,q]\\ [A,h \mid B,i,X,C,j,X \mid -,-,-,-]\\ [A,h \mid D,p,X,E,q,- \mid O,u,P,v]\end{array}}{[A,h \mid B,i,X,G,k,- \mid O,u,P,v]} \quad \begin{array}{l} C \longmapsto [C,X'F' \\ C,[F \longmapsto G\end{array}$$

– a una derivación de retorno:

$$\begin{array}{ll}(\Upsilon\ [\alpha A,a_{h+1}\dots a_n) & \stackrel{*}{\vdash} (\Upsilon\ [A\ \Upsilon_1\ [\alpha XB,a_{i+1}\dots a_n)\\ & \stackrel{*}{\vdash} (\Upsilon\ [A\ \Upsilon_1\ [B\ \Upsilon_2\ [\alpha D,a_{p+1}\dots a_n)\\ & \stackrel{*}{\vdash} (\Upsilon\ [A\ \Upsilon_1\ [C\ \Upsilon_2\ [E,a_{q+1}\dots a_n)\\ & \stackrel{*}{\vdash} (\Upsilon\ [A\ \Upsilon_1\ [C,a_{j+1}\dots a_n)\\ & \vdash (\Upsilon\ [A\ \Upsilon_1\ [C\ [X'F',a_{j+1}\dots a_n)\\ & \stackrel{*}{\vdash} (\Upsilon\ [A\ \Upsilon_1\ [C\ [F'\ \Upsilon_3\ [O,a_{u+1}\dots a_n)\\ & \stackrel{*}{\vdash} (\Upsilon\ [A\ \Upsilon_1\ [C\ [F'\ \Upsilon_3\ [P,a_{v+1}\dots a_n)\\ & \stackrel{*}{\vdash} (\Upsilon\ [A\ \Upsilon_1\ [C\ [F,a_{k+1}\dots a_n)\\ & \vdash (\Upsilon\ [A\ \Upsilon_1\ [G,a_{k+1}\dots a_n)\end{array}$$

$$\frac{\begin{array}{c}[C,j \mid F',j,X',F,k,- \mid O,u,P,v]\\ [A,h \mid B,i,X,C,j,- \mid D,p,E,q]\\ [-,- \mid O,u,-,P,v,- \mid -,-,-,-]\end{array}}{[A,h \mid B,i,X,G,k,- \mid D,p,E,q]} \quad \begin{array}{l} C \longmapsto [C,X'F' \\ C,[F \longmapsto G\end{array}$$

– a una derivación de puntos especiales:

$$\begin{array}{ll}(\Upsilon\ [B, a_{i+1} \dots a_n) & \overset{*}{\vdash} (\Upsilon\ [C, a_{j+1} \dots a_n) \\ & \vdash (\Upsilon\ [C\ [X'F', a_{j+1} \dots a_n) \\ & \overset{*}{\vdash} (\Upsilon\ [C\ [F'\ \Upsilon_1\ [O, a_{u+1} \dots a_n) \\ & \overset{*}{\vdash} (\Upsilon\ [C\ [F'\ \Upsilon_1\ [P, a_{v+1} \dots a_n) \\ & \overset{*}{\vdash} (\Upsilon\ [C\ [F, a_{k+1} \dots a_n) \\ & \vdash (\Upsilon\ [G, a_{k+1} \dots a_n)\end{array}$$

$$\cfrac{\begin{array}{c}[C, j \mid F', j, X', F, k, - \mid O, u, P, v] \\ [-, - \mid B, i, -, C, j, - \mid -, -, -, -] \\ [-, - \mid O, u, -, P, v, - \mid -, -, -, -]\end{array}}{[-, - \mid B, i, -, G, k, - \mid -, -, -, -]} \quad \begin{array}{l}C \longmapsto [C, X'F' \\ C, [F \longmapsto G\end{array}$$

- Derivaciones que son el resultado de aplicar una transición $C, [F \longmapsto G$ a una derivación obtenida tras aplicar una transición $XC \longmapsto [C, F'$ a una derivación de llamada, con los tres casos siguientes:
  – la derivación de llamada es a su vez derivada de una derivación de llamada:

$$\begin{array}{ll}(\Upsilon\ [\alpha M, a_{m+1} \dots a_n) & \overset{*}{\vdash} (\Upsilon\ [M\ \Upsilon_1\ \alpha X'N, a_{t+1} \dots a_n) \\ & \overset{*}{\vdash} (\Upsilon\ [M\ \Upsilon_1\ [\alpha X'A, a_{h+1} \dots a_n) \\ & \overset{*}{\vdash} (\Upsilon\ [M\ \Upsilon_1\ [A\ \Upsilon_2\ [\alpha X'XB, a_{i+1} \dots a_n) \\ & \overset{*}{\vdash} (\Upsilon\ [M\ \Upsilon_1\ [A\ \Upsilon_2\ [\alpha X'XC, a_{j+1} \dots a_n) \\ & \vdash (\Upsilon\ [M\ \Upsilon_1\ [A\ \Upsilon_2\ [C\ [\alpha X'F', a_{j+1} \dots a_n) \\ & \overset{*}{\vdash} (\Upsilon\ [M\ \Upsilon_1\ [A\ \Upsilon_2\ [C\ [F'\ \Upsilon_3\ [D[\delta], a_{p+1} \dots a_n) \\ & \overset{*}{\vdash} (\Upsilon\ [M\ \Upsilon_1\ [A\ \Upsilon_2\ [C\ [F'\ \Upsilon_3\ [E, a_{q+1} \dots a_n) \\ & \overset{*}{\vdash} (\Upsilon\ [M\ \Upsilon_1\ [A\ \Upsilon_2\ [C\ [F, a_{k+1} \dots a_n) \\ & \vdash (\Upsilon\ [M\ \Upsilon_1\ [A\ \Upsilon_2\ [G, a_{k+1} \dots a_n)\end{array}$$

$$\cfrac{\begin{array}{c}[M, m \mid F', j, X', F, k, - \mid D, p, E, q] \\ [A, h \mid B, i, X, C, j, X \mid -, -, -, -] \\ [M, m \mid N, t, X', A, h, X' \mid -, -, -, -]\end{array}}{[A, h \mid B, i, X, G, k, - \mid F', j, F, k]} \quad \begin{array}{l}XC \longmapsto [C, F' \\ C, [F \longmapsto G\end{array}$$

  – la derivación de llamada es a su vez derivada de una derivación de retorno:

$$\begin{array}{ll}(\Upsilon\ [\alpha M, a_{m+1} \dots a_n) & \overset{*}{\vdash} (\Upsilon\ [M\ \Upsilon_1\ [\alpha X'N, a_{t+1} \dots a_n) \\ & \overset{*}{\vdash} (\Upsilon\ [M\ \Upsilon_1\ [N\ \Upsilon_2\ [\alpha D, a_{p+1} \dots a_n) \\ & \overset{*}{\vdash} (\Upsilon\ [M\ \Upsilon_1\ [N\ \Upsilon_2\ [E, a_{q+1} \dots a_n) \\ & \overset{*}{\vdash} (\Upsilon\ [M\ \Upsilon_1\ [A, a_{h+1} \dots a_n) \\ & \overset{*}{\vdash} (\Upsilon\ [M\ \Upsilon_1\ [A\ \Upsilon_3\ [XB, a_{i+1} \dots a_n) \\ & \overset{*}{\vdash} (\Upsilon\ [M\ \Upsilon_1\ [A\ \Upsilon_3\ [XC, a_{j+1} \dots a_n) \\ & \vdash (\Upsilon\ [M\ \Upsilon_1\ [A\ \Upsilon_3\ [XC\ [F', a_{j+1} \dots a_n) \\ & \overset{*}{\vdash} (\Upsilon\ [M\ \Upsilon_1\ [A\ \Upsilon_3\ [XC\ [F, a_{k+1} \dots a_n) \\ & \vdash (\Upsilon\ [M\ \Upsilon_1\ [A\ \Upsilon_3\ [G, a_{k+1} \dots a_n)\end{array}$$

$$\cfrac{\begin{array}{c}[-, - \mid F', j, -, F, k, - \mid -, -, -, -] \\ [A, h \mid B, i, X, C, j, X \mid -, -, -, -] \\ [M, m \mid N, t, X', A, h, - \mid D, p, E, q]\end{array}}{[A, h \mid B, i, X, G, k, - \mid F', j, F, k]} \quad \begin{array}{l}XC \longmapsto [C, F' \\ C, [F \longmapsto G\end{array}$$

– la derivación de llamada es a su vez derivada de una derivación de puntos especiales:

$$\begin{array}{ll}
(\lbrack\alpha X'N, a_{t+1}\dots a_n) & \overset{*}{\vdash} (\lbrack A, a_{h+1}\dots a_n) \\
& \overset{*}{\vdash} (\lbrack A\ \Upsilon_3\ \lbrack XB, a_{i+1}\dots a_n) \\
& \overset{*}{\vdash} (\lbrack A\ \Upsilon_3\ \lbrack XC, a_{j+1}\dots a_n) \\
& \vdash (\lbrack A\ \Upsilon_3\ \lbrack C\ \lbrack F', a_{j+1}\dots a_n) \\
& \overset{*}{\vdash} (\lbrack A\ \Upsilon_3\ \lbrack C\ \lbrack F, a_{k+1}\dots a_n) \\
& \vdash (\lbrack A\ \Upsilon_3\ \lbrack G, a_{k+1}\dots a_n)
\end{array}$$

$$\frac{\begin{array}{c}[-,- \mid F',j,-,F,k,- \mid -,-,-,-] \\ [A,h \mid B,i,X,C,j,X \mid -,-,-,-] \\ [-,- \mid N,t,X',A,h,- \mid -,-,-,-]\end{array}}{[A,h \mid B,i,X,G,k,- \mid F',j,F,k]} \quad \begin{array}{l} XC \longmapsto \lbrack C,F' \\ C,\lbrack F \longmapsto G\end{array}$$

□

La complejidad espacial de la técnica de tabulación con respecto a la longitud n de la cadena de entrada es $\mathcal{O}(n^5)$, puesto que cada ítem almacena 5 posiciones de la cadena de entrada. La complejidad temporal es $\mathcal{O}(n^7)$. Dicha complejidad viene dada por la siguiente regla:

$$\frac{\begin{array}{c}[C,j \mid F',j,X',F,k,- \mid D,p,E,q] \\ [A,h \mid B,i,X,C,j,X \mid -,-,-,-] \\ [A,h \mid D,p,X,E,q,- \mid O,u,P,v]\end{array}}{[A,h \mid B,i,X,G,k,- \mid O,u,P,v]} \quad \begin{array}{l} C \longmapsto \lbrack C,X'F' \\ C,\lbrack F \longmapsto G\end{array}$$

Podemos reducir la complejidad temporal aplicando la técnica propuesta en [53, 125], según la cual la regla anterior debe ser dividida en dos reglas de menor complejidad, de tal modo que el consecuente de la primera sea un pseudo-ítem intermedio que proporcione a la segunda regla la información necesaria para que la aplicación combinada de ambas reglas sea equivalente a la regla original. En el caso que nos ocupa, las dos reglas obtenidas son

$$\frac{\begin{array}{c}[C,j \mid F',j,X',F,k,- \mid D,p,E,q] \\ [A,h \mid D,p,X,E,q,- \mid O,u,P,v]\end{array}}{[[F',j,X',F,k,- \mid O,u,P,v]]} \quad \begin{array}{l} C \longmapsto \lbrack C,X'F' \\ C,\lbrack F \longmapsto G\end{array}$$

$$\frac{\begin{array}{c}[[F',j,X',F,k,- \mid O,u,P,v]] \\ [A,h \mid B,i,X,C,j,X \mid -,-,-,-] \\ [A,h \mid D,p,X,E,q,- \mid O,u,P,v]\end{array}}{[A,h \mid B,i,X,G,k,- \mid O,u,P,v]} \quad \begin{array}{l} C \longmapsto \lbrack C,X'F' \\ C,\lbrack F \longmapsto G\end{array}$$

donde $[[F',j,X',F,k,- \mid O,u,P,v]]$ es el pseudo-ítem intermedio que relaciona las dos reglas. La primera regla ignora la posición h, que es posteriormente recuperada del segundo y tercer ítem que intervienen en la segunda regla, que junto con el pseudo-ítem son suficientes para garantizar la existencia del ítem $[C,j \mid F',j,X',F,k,- \mid D,p,E,q]$ por la definición de derivaciones de llamada y retorno. La primera regla presenta una complejidad temporal $\mathcal{O}(n^6)$ (la posición h no interviene) y la segunda presenta también una complejidad $\mathcal{O}(n^6)$ (las posiciones p y q no intervienen) por lo que hemos logrado rebajar la complejidad temporal final de la técnica de tabulación a $\mathcal{O}(n^6)$.

Capítulo 7

Autómatas a pila embebidos ascendentes

En este capítulo se presentan los autómatas a pila embebidos ascendentes o BEPDA, que constituye el modelo dual de los autómatas a pila embebidos estudiados en el capítulo precedente. Las aportaciones más importantes realizadas en este capítulo son: la definición formal consistente de los BEPDA con estados, la definición de los BEPDA sin estados, la demostración de que estos autómatas aceptan la clase de los lenguajes de adjunción de árboles, la definición de esquemas de compilación para gramáticas de adjunción de árboles y gramáticas lineales de índices y el diseño de técnicas de tabulación que permiten la ejecución de los BEPDA en tiempo polinomial. Este capítulo está basado en [17].

7.1 Introducción

Los autómatas a pila embebidos ascendentes (*Bottom-up Embedded Push-Down Automata*, BEPDA) fueron descritos por Schabes en [168] y por Schabes y Vijay-Shanker en [176] con el fin de proporcionar un modelo de autómata sobre el cual poder ejecutar un algoritmo de análisis sintáctico de tipo LR para gramáticas de adjunción de árboles. Rambow trata de definir formalmente las propiedades de este modelo de autómata en [152], aunque no lo consigue totalmente puesto que su trabajo presenta inconsistencias entre la definición de las configuraciones y la definición de las transiciones permitidas.

Los autómatas a pila embebidos ascendentes constituyen el modelo dual de los autómatas a pila embebidos, pues mientras estos últimos permiten simular derivaciones de una gramática de adjunción de árboles en las cuales las adjunciones se reconocen de modo descendente, los BEPDA sólo pueden tratar derivaciones en las cuales las adjunciones se reconocen de modo ascendente. Esta dualidad es una consecuencia del juego de transiciones que se permite utilizar en cada uno de los modelos de autómata citados.

7.2 Definición con estados

Al igual que los autómatas a pila embebidos, los autómatas a pila embebidos ascendentes trabajan con una pila, que denominamos *principal*, constituida a su vez por pilas. La pila principal, la cadena de entrada y el control finito constituyen los tres elementos básicos de un BEPDA. La *configuración* en un momento dado viene determinada por el estado del control finito en que se encuentra el autómata, el contenido de la pila y la parte de la cadena de entrada que resta por leer, como se muestra en las figuras 7.1 y 7.3. Un conjunto de *transiciones* permite cambiar de configuración. Existen dos tipos diferentes de transiciones:

1. Transiciones que consultan:

   - el estado q del control finito;
   - el siguiente símbolo terminal a de la cadena de entrada.

   Como resultado, el autómata puede cambiar al estado q', avanzar una posición en la cadena de entrada y crear una nueva pila $[Z'$ y situarla en la cima de la pila principal. En la figura 7.2 se muestra el resultado de aplicar una transición

   $$(q', [Z) \in \delta(q, a, \epsilon, \epsilon, \epsilon)$$

   a un autómata a pila embebido ascendente con la configuración de la figura 7.1.

   El dual de este tipo de transición en EPDA con estados lo constituiría el borrado de las pilas vacías, con la salvedad de que en los EPDA con estados no es necesario especificar transiciones para realizar dicha acción, pues se lleva a cabo implícitamente siempre que una pila queda vacía.

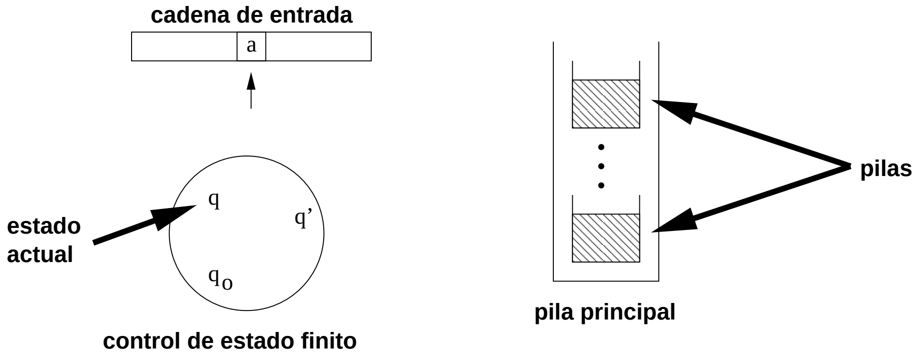

Figura 7.1: Una configuración de un BEPDA

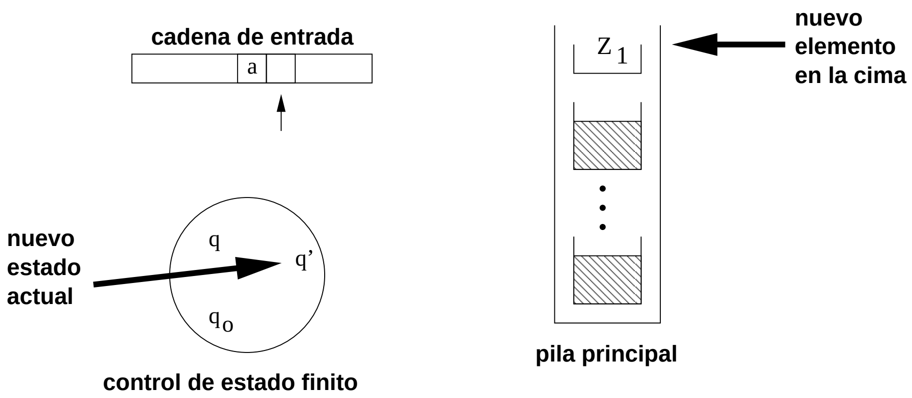

Figura 7.2: Configuración de un BEPDA tras aplicar una transición del primer tipo

2. Transiciones que consultan:

   - el estado q del control finito;
   - el siguiente símbolo terminal a de la cadena de entrada;

- el contenido de i pilas $[\alpha_j$, con $1 \leq j \leq i$, de tamaño acotado situadas en la cima de la pila principal;
- m elementos situados en la cima de la pila $[\alpha Z_m \ldots Z_1$ situada bajo las i anteriores;
- el contenido de las $k-i$ pilas $[\alpha_j$, con $i+1 \leq j \leq k$, de tamaño acotado situadas bajo la pila anterior.

Como resultado, el autómata puede:

- cambiar al estado q';
- avanzar una posición en la cadena de entrada;
- eliminar las i pilas de tamaño acotado de la cima;
- reemplazar los m elementos de la siguiente pila por el elemento X;
- eliminar las $k-i$ pilas de tamaño acotado situadas bajo dicha pila.

La figura 7.4 muestra el resultado de aplicar una transición

$$(q', Z) \in \delta(q,\ a,\ [\alpha_k \ldots [\alpha_{i+1},\ Z_m \ldots Z_1,\ [\alpha_i \ldots [\alpha_1)$$

a un autómata a pila embebido ascendente con la configuración de la figura 7.3.

Este tipo de transiciones se corresponde con el dual de las transiciones de un EPDA.

Definimos formalmente un autómata a pila embebido ascendente como una tupla $(Q, V_T, V_S, \delta, q_0, Q_F, \$_f)$ donde:

- Q es un conjunto finito de estados.
- V_T es un conjunto finito de símbolos terminales.
- V_S es un conjunto finito de símbolos de pila.
- q_0 es el estado inicial.
- $Q_F \subseteq Q$ es el conjunto de estados finales.
- $\$_f \in V_S$ es el símbolo final de la pila.
- δ es una relación de $Q \times V_T \cup \{\epsilon\} \times ([V_S^+)^* \times V_S^* \times ([V_S^+)^*$ en subconjuntos finitos de $Q \times V_S \cup \{[V_S\}$, donde $[\notin V_S$ es un símbolo utilizado para separar las diferentes pilas que componen la pila principal.

La *configuración* de un autómata a pila embebido ascendente en un momento dado viene definida por el triple (q, Υ, w), donde $q \in Q$ indica el estado en el que se encuentra, $\Upsilon \in ([V_S^+)^*$ el contenido de la pila principal y $w \in V_T^*$ la parte de la cadena de entrada que resta por leer. Es importante indicar que todas las pilas almacenadas en la pila principal deben contener al menos un elemento. La pila principal sólo estará vacía en la configuración inicial (q_0, ϵ, w).

El paso de una configuración (q, Υ, aw) a otra configuración (q', Υ', w) viene determinado por la aplicación de una transición y se denota mediante $\vdash$. Tenemos los dos casos siguientes:

1. Dada una configuración (q, Υ, aw) y una transición

$$(q', [Z) \in \delta(q, a, \epsilon, \epsilon, \epsilon)$$

tenemos que

$$(q, \Upsilon, aw) \vdash (q', \Upsilon[Z, w)$$

puesto que el autómata a pila embebido ascendente pasará a la nueva configuración $(q', \Upsilon[Z, w)$.

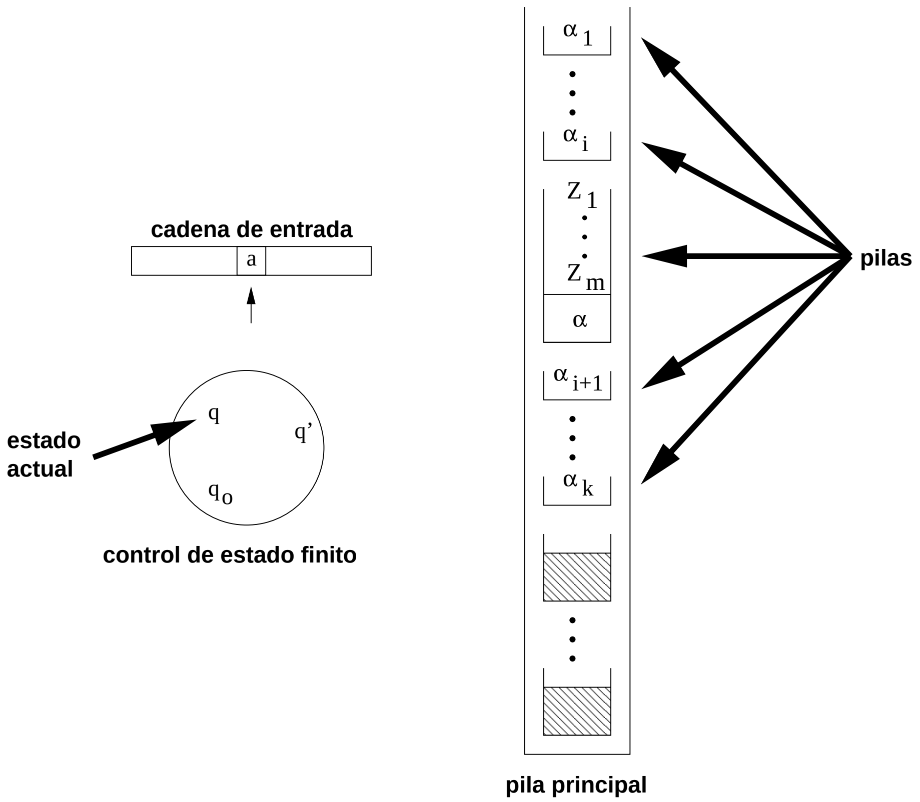

Figura 7.3: Otra configuración de un BEPDA

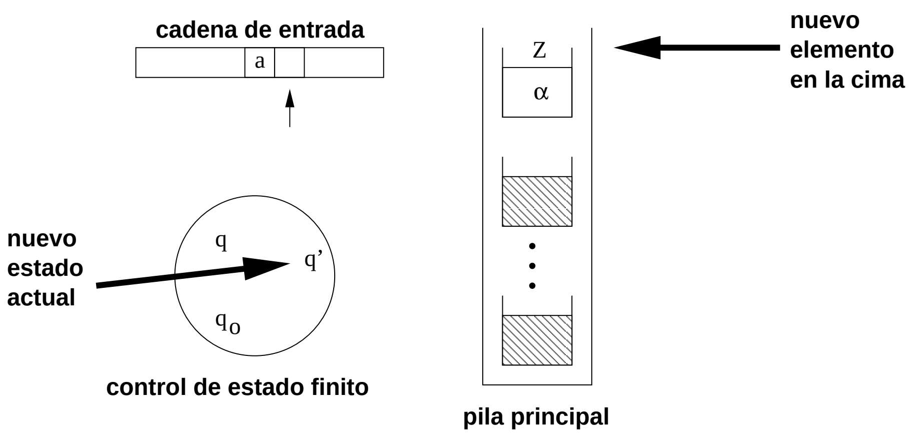

Figura 7.4: Configuración de un BEPDA tras aplicar una transición del segundo tipo

2. Dada una configuración $(q, \Upsilon[\alpha_k \ldots [\alpha_{i+1}[\alpha Z_m \ldots Z_1[\alpha_i \ldots [\alpha_1, aw)$ y una transición

$$(q', Z) \in \delta(q,\ a,\ [\alpha_k \ldots [\alpha_{i+1},\ Z_m \ldots Z_1,\ [\alpha_i \ldots [\alpha_1)$$

tenemos que

$$(q, \Upsilon[\alpha_k \ldots [\alpha_{i+1}[\alpha Z_m \ldots Z_1[\alpha_i \ldots [\alpha_1, aw) \vdash (q', \Upsilon[\alpha Z, w)$$

puesto que el autómata a pila embebido ascendente pasará a la nueva configuración $(q', \Upsilon[\alpha Z, w)$.

Denotamos por $\overset{*}{\vdash}$ el cierre reflexivo y transitivo de $\vdash$.

El *lenguaje aceptado por estado final* por un autómata a pila embebido ascendente viene determinado por el conjunto de cadenas $w \in V_T^*$ tal que $(q_0, \epsilon, w) \overset{*}{\vdash} (p, \Upsilon, \epsilon)$, donde $p \in Q_F$ y $\Upsilon \in ([V_S^+)^*$.

El *lenguaje aceptado por pila vacía* por un autómata a pila embebido viene determinado por el conjunto de cadenas $w \in V_T^*$ tal que $(q_0, \epsilon, w) \overset{*}{\vdash} (p, [\$_f, \epsilon)$ para cualquier $q \in Q$. Observamos una vez más la dualidad existente entre EPDA y BEPDA, pues mientras el primero parte de una pila $[\$_0$ para finalizar con una pila vacía, el segundo parte de una pila vacía para finalizar con una pila $[\$_f$.

Teorema 7.1 *Dado un autómata a pila embebido ascendente $\mathcal{A}$ que acepta el lenguaje L por estado final, existe un autómata a pila embebido ascendente $\mathcal{A}'$ que acepta L por pila vacía.*

Demostración:

Sea $\mathcal{A} = (Q, V_T, V_S, \delta, q_0, Q_F, \$_f)$ el BEPDA que acepta el lenguaje L por estado final. El BEPDA $\mathcal{A}' = (Q', V_T, V_S, \delta', q_0, Q_F, \$_f)$ acepta L por pila vacía, con $Q' = Q \cup \{q'\}$ tal que $q' \notin Q$, y con δ' conteniendo las transiciones en δ junto con las siguientes transiciones encargadas de vaciar al pila cuando se alcanza un estado final:

$$(q', \$_f) \in \delta'(q_f, \epsilon, \epsilon, Z, \epsilon)$$

$$(q', \$_f) \in \delta'(q', \epsilon, \epsilon, Z\,\$_f, \epsilon)$$

$$(q', \$_f) \in \delta'(q', \epsilon, \epsilon, Z, [\$_f)$$

para todo $q_f \in Q_F$ y $Z \in V_S$. □

Teorema 7.2 *Dado un autómata a pila embebido ascendente $\mathcal{A}$ que acepta el lenguaje L por pila vacía, existe un autómata a pila embebido ascendente $\mathcal{A}'$ que acepta L por estado final.*

Demostración:

Sea $\mathcal{A} = (Q, V_T, V_S, \delta, q_0, Q_F, \$_f)$ el BEPDA que acepta el lenguaje L por pila vacía. El BEPDA $\mathcal{A}' = (Q, V_T, V_S', \delta, q_0, Q_F, \$_f)$ acepta L por estado final, con $V_S' = V_S \cup \{\$_0\}$ tal que $\$_0 \notin V_S$ y con δ' conteniendo las transiciones de δ junto con las siguientes transiciones encargadas de situar una pila $[\$_0$ al comienzo de al operación de $\mathcal{A}'$, y de detectar la pila $[\$_0[\$_f$, que correspondería a la pila $[\$_f$ de la configuración final de $\mathcal{A}$, para poder transitar a un estado final.

$$(q_0, [\$_0) \in \delta'(q_0, \epsilon, \epsilon, \epsilon, \epsilon)$$

$$(q_f, \$_f) \in \delta'(q, \epsilon, \epsilon, \$_0, [\$_f)$$

para todo $q \in Q$. □

Teorema 7.3 *Dado un BEPDA $\mathcal{A}$, existe un BEPDA $\mathcal{A}'$ tal que el lenguaje aceptado por $\mathcal{A}$ coincide con el lenguaje aceptado por $\mathcal{A}'$ y las transiciones en $\mathcal{A}'$ están normalizadas, esto es, tienen alguna de las dos formas siguientes:*

$$(q', [Z) \in \delta(q, a, \epsilon, \epsilon, \epsilon)$$

$$(q', Z) \in \delta(q,\ a,\ [Z'_k \ldots [Z'_{i+1},\ Z_m \ldots Z_1,\ [Z'_i \ldots [Z'_1)$$

donde $q, q' \in Q$, $a \in V_T \cup \{\epsilon\}$, $Z, Z_1, \ldots, Z_m \in V_S$, $Z'_1, \ldots, Z'_k \in V_S$, $0 \leq i \leq k$ y $0 \leq m \leq 2$.

Demostración:

Las transiciones de $\mathcal{A}'$ se construyen a partir de las transiciones de $\mathcal{A}$. Las transiciones

$$(q', [Z) \in \delta(q, a, \epsilon, \epsilon, \epsilon)$$

permanecen sin cambios. Con respecto a las transiciones

$$(q', Z) \in \delta(q, a, [Z'_{k,l_k} \ldots Z'_{k,1} [\ldots [Z'_{i+1,l_{i+1}} \ldots Z'_{i+1,1}, Z_p \ldots Z_1, [Z'_{i,l_i} \ldots Z'_{i,1} [\ldots [Z'_{1,l_1} \ldots Z'_{1,1})$$

precisaremos introducir un nuevo estado q'' que permita distinguir las configuraciones derivadas del proceso de normalización del resto de las transiciones. A continuación se muestra el nuevo conjunto de transiciones normalizadas que sustituyen a las transiciones de este último tipo:

- Una transición de inicio:

$$(q'', Z'_{1,1}) \in \delta(q, \epsilon, \epsilon, Z'_{1,1}, \epsilon) \qquad \text{si } [Z'_{i,l_i} \ldots Z'_{i,1} [\ldots [Z'_{1,l_1} \ldots Z'_{1,1} \neq \epsilon$$

- Un conjunto de transiciones para el vaciado de cada una de las pilas en $[Z'_{i,l_i} \ldots Z'_{i,1} [\ldots [Z'_{1,l_1} \ldots Z'_{1,1}$:

$$(q'', Z'_{r,s+1}) \in \delta(q'', \epsilon, \epsilon, Z'_{r,s+1}\, Z'_{r,s}, \epsilon) \qquad 1 \leq r \leq i,\ 1 \leq s < l_r$$
$$(q'', Z'_{r+1,1}) \in \delta(q'', \epsilon, \epsilon, Z'_{r+1,1}, [Z'_{r,l_r}, \epsilon) \quad 1 \leq r < i$$

  En el caso de $i = 0$, esto es $[Z'_{i,l_i} \ldots Z'_{i,1} [\ldots [Z'_{1,l_1} \ldots Z'_{1,1} = \epsilon$, en lugar de las transiciones anteriores tendremos una sola transición

$$(q'', [Z'_{0,0}) \in \delta(q, \epsilon, \epsilon, \epsilon)$$

- Un conjunto de transiciones para pasar a la pila que tiene $Z_p \ldots Z_1$ en su cima. En el caso de $p = 0$ estas transiciones también apilan Z:

$$(q'', Z_1) \in \delta(q'', \epsilon, \epsilon, Z_1, [Z'_{i,l_i}, \epsilon) \quad \text{si } p \geq 1$$
$$(q'', Z) \in \delta(q'', \epsilon, \epsilon, Z', [Z'_{i,l_i}, \epsilon) \quad \forall Z' \in V_S \text{ si } p = 0$$

- Un conjunto de transiciones para la sustitución de $Z_p \ldots Z_1$ por Z cuando $p \geq 1$:

$$(q'', Z_{s+1}) \in \delta(q'', \epsilon, \epsilon, Z_{s+1}\, Z_s, \epsilon) \quad 1 \leq s < p$$
$$(q'', Z) \in \delta(q'', \epsilon, \epsilon, Z_p, \epsilon)$$

- Un conjunto de transiciones que vacían las pilas en $[Z'_{i,l_i} \dots Z'_{i,1} [\dots [Z'_{1,l_1} \dots Z'_{1,1}$, dejándolas únicamente con el último elemento:

$$
\begin{array}{ll}
(q'', Z'_{r,1}) \in \delta(q''', \epsilon, \epsilon, Z'_{r,1}, \epsilon) & \forall q''' \in Q \\
(q'', Z'_{r,s+1}) \in \delta(q'', \epsilon, \epsilon, Z'_{r,s+1}\, Z'_{r,s}, \epsilon) & i+1 \leq r \leq k,\ 1 \leq s < l_r
\end{array}
$$

- Un conjunto de transiciones para ir eliminando las pilas $[Z'_{i,l_i} [\dots [Z'_{1,l_1}$ resultantes de la aplicación del anterior conjunto de transiciones:

$$(q'', Z) \in \delta(q'', \epsilon, Z'_{r,l_r}, Z, \epsilon) \quad i+1 \leq r \leq k$$

□

Ejemplo 7.1 El autómata embebido a pila ascendente definido por la tupla $(\{q_0, q_1, q_2, q_3\}, \{a, b, c, d\}, \{B, C, D\}, \delta, q_0, \emptyset, \$_f)$, donde δ contiene las transiciones mostradas en la parte izquierda de la tabla 7.1, acepta el lenguaje $\{a^n b^n c^n d^n \mid n \geq 0\}$ por pila vacía. En la parte derecha de la tabla 7.1 se muestra la secuencia de configuraciones que sigue el autómata para analizar correctamente la cadena de entrada *aabbccdd*. La primera columna muestra la transición aplicada, la segunda el estado, la tercera el contenido de la pila y la cuarta la parte que resta por leer de la cadena de entrada. Obsérvese como la secuencia de pilas coincide, en orden inverso, con la secuencia de pilas de las configuraciones del autómata a pila embebido de la tabla 6.1. ¶

7.3 Autómatas a pila embebidos ascendentes sin estados

Al igual que en el caso de los autómatas a pila y de los autómatas a pila embebidos, el control finito es un elemento prescindible de los autómatas a pila embebidos ascendentes puesto que el estado correspondiente a una configuración puede ser incluido en el elemento de la cima de la pila. Como resultado obtenemos una definición alternativa, que juzgamos más simple y homogénea, según la cual un autómata a pila embebido es una tupla $(V_T, V_S, \Theta, \$_0, \$_f)$ donde:

- V_T es un conjunto finito de símbolos terminales.
- V_S es un conjunto finito de símbolos de pila.
- $\$_0 \in V_S$ es el símbolo inicial de pila.
- $\$_f \in V_S$ es el símbolo final de pila.
- Θ es un conjunto de transiciones, cada una de las cuales pertenece a uno de los siguientes tipos, donde $C, F, G \in V_S$, $\Upsilon \in ([V_S^*)^*$, $\alpha \in V_S^*$ y $a \in V_T \cup \{\epsilon\}$:

  SWAP: Transiciones de la forma $C \overset{a}{\longmapsto} F$ que reemplazan el elemento C de la cima de la pila por el elemento F mientras se lee a de la cadena de entrada. El resultado de aplicar una transición de este tipo a una pila $\Upsilon [\alpha C$ es una pila $\Upsilon [\alpha F$.

  PUSH: Transiciones de la forma $C \overset{a}{\longmapsto} CF$ que apilan un nuevo elemento F mientras se lee a de la cadena de entrada. El resultado de aplicar una transición de este tipo a una pila $\Upsilon [\alpha C$ es una pila $\Upsilon [\alpha CF$.

(a)	$(q_0, [D) \in \delta(q_0, a, \epsilon, \epsilon, \epsilon)$
(b)	$(q_1, [C) \in \delta(q_0, b, \epsilon, \epsilon, \epsilon)$
(c)	$(q_1, [C) \in \delta(q_1, b, \epsilon, \epsilon, \epsilon)$
(d)	$(q_2, B) \in \delta(q_1, c, \epsilon, C, \epsilon)$
(e)	$(q_2, B) \in \delta(q_2, c, [C, \epsilon, \epsilon)$
(f)	$(q_3, B) \in \delta(q_2, d, [D, BB, \epsilon)$
(g)	$(q_3, B) \in \delta(q_3, d, [D, BB, \epsilon)$
(h)	$(q_3, \$_f) \in \delta(q_3, d, [D, B, \epsilon)$
(i)	$(q_0, [\$_f) \in \delta(q_0, a, \epsilon, \epsilon, \epsilon)$

	q_0		$aabbccdd$
(a)	q_0	$[D$	$abbccdd$
(a)	q_0	$[D[D$	$bbccdd$
(b)	q_1	$[D[D[C$	$bccdd$
(c)	q_1	$[D[D[C[C$	$ccdd$
(d)	q_2	$[D[D[C[B$	cdd
(e)	q_2	$[D[D[BB$	dd
(f)	q_3	$[D[B$	d
(h)	q_3	$[\$_f$	

Tabla 7.1: Transiciones del BEPDA que acepta $\{a^n b^n c^n d^n \mid n > 0\}$ (izquierda) y configuraciones de dicho autómata durante el análisis de $aabbccdd$ (derecha)

POP: Transiciones de la forma $CF \overset{a}{\longmapsto} G$ que eliminan los dos elementos C y F de la cima de la pila y los sustituyen por G mientras se lee a de la cadena de entrada. El resultado de aplicar una transición de este tipo a una pila $\Upsilon[\alpha CF$ es una pila $\Upsilon[\alpha G$.

UNWRAP-A: Transiciones de la forma $C, [F \overset{a}{\longmapsto} G$ que eliminan la pila $[F$ situada en la cima de la pila principal, mientras la cima C de la pila situada inmediatamente debajo es reemplazada por G. El resultado de aplicar una transición de este tipo a una pila $\Upsilon[\alpha C[F$ es una pila $\Upsilon[\alpha G$.

UNWRAP-B: Transiciones de la forma $[C, F \overset{a}{\longmapsto} G$ que reemplazan el elemento F situado en la cima de la pila ubicada en la cima de la pila principal por G, mientras que eliminan la pila $[C$ situada bajo dicha pila. El resultado de aplicar una transición de este tipo a una pila $\Upsilon[C[\alpha F$ es una pila $\Upsilon[\alpha G$.

WRAP: Transiciones de la forma $C \overset{a}{\longmapsto} C, [F$ que sitúan una nueva pila $[F$ en la cima de la pila principal. El resultado de aplicar una transición de este tipo a una pila $\Upsilon[\alpha C$ es una pila $\Upsilon[\alpha C[F$.

Los tres primeros tipos coinciden con las transiciones del mismo nombre presentes en los autómatas a pila embebidos. Las transiciones de tipo UNWRAP-A y UNWRAP-B permiten eliminar una pila situado encima y debajo, respectivamente, de la pila que ocupará la cima de la pila principal una vez aplicada la transición, como se puede observar en la figura 7.5. Las transiciones de tipo WRAP permiten crear una nueva pila en la cima de la pila principal, como muestra la figura 7.6.

Comparando la figura 7.5 con la 6.4 se observa que UNWRAP-A y UNWRAP-B son las transiciones duales de WRAP-A y WRAP-B, respectivamente. Comparando la figura 7.6

con la 6.5 se observa que WRAP es la transición dual de UNWRAP.

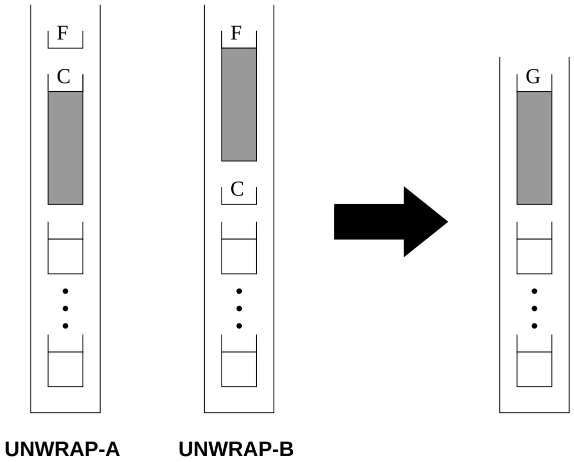

Figura 7.5: Transiciones UNWRAP-A y UNWRAP-B

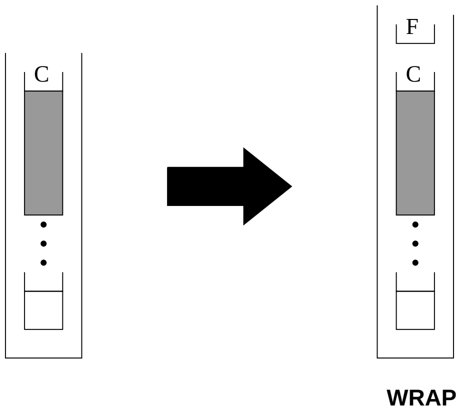

Figura 7.6: Transición WRAP

La *configuración* en un momento dado de un autómata a pila embebido ascendente sin estados viene determinada por el par (Υ, w), donde Υ es el contenido de la pila y w la parte de la cadena de entrada que resta por leer. Una configuración (Υ, aw) deriva una configuración (Υ', w), hecho que denotamos mediante $(\Upsilon, aw) \vdash (\Upsilon', w)$, si y sólo si existe una transición que aplicada a Υ devuelve Υ' y consume a de la cadena de entrada. En caso de ser necesario identificar una derivación d concreta, utilizaremos la notación $\overset{*}{\vdash}_d$. Denotamos por $\overset{*}{\vdash}$ el cierre reflexivo y transitivo de $\vdash$.

Decimos que una cadena de entrada w es aceptada por un autómata a pila embebido si $([\$_0, w) \overset{*}{\vdash} ([\$_0 [\$_f, \epsilon)$. El *lenguaje aceptado* por un autómata a pila embebido viene determinado por el conjunto de cadenas $w \in V_T^*$ tal que $([\$_0, w) \overset{*}{\vdash} ([\$_0 [\$_f, \epsilon)$.

En lo que resta, cuando nos refiramos a BEPDA sin especificar si se trata de la versión con estados o sin estados, debe entenderse que nos estamos refiriendo a la versión sin estados que acabamos de definir.

(a)	$\$_0 \stackrel{a}{\longmapsto} \$_0, [D$
(b)	$D \stackrel{a}{\longmapsto} D, [D$
(c)	$D \longmapsto D, [C$
(d)	$C \stackrel{b}{\longmapsto} C, [C$
(e)	$C \longmapsto B$
(f)	$B \longmapsto BE$
(g)	$[C, E \stackrel{c}{\longmapsto} C$
(h)	$BC \longmapsto B$
(i)	$[D, B \stackrel{d}{\longmapsto} D$
(j)	$BD \longmapsto B$
(k)	$D \longmapsto \$_f$

	$[\$_0$	$aabbccdd$
(a)	$[\$_0[D$	$aabbccdd$
(b)	$[\$_0[D[D$	$abbccdd$
(c)	$[\$_0[D[D[C$	$bbccdd$
(d)	$[\$_0[D[D[C[C$	$bccdd$
(d)	$[\$_0[D[D[C[C[C$	$ccdd$
(e)	$[\$_0[D[D[C[C[B$	$ccdd$
(f)	$[\$_0[D[D[C[C[BE$	$ccdd$
(g)	$[\$_0[D[D[C[BC$	cdd
(e)	$[\$_0[D[D[C[BB$	cdd
(f)	$[\$_0[D[D[C[BBE$	cdd
(g)	$[\$_0[D[D[BBC$	dd
(h)	$[\$_0[D[D[BB$	dd
(i)	$[\$_0[D[BD$	d
(j)	$[\$_0[D[B$	d
(i)	$[\$_0[D$	
(k)	$[\$_0[\$_f$	

Tabla 7.2: Transiciones del BEPDA sin estados que acepta $\{a^n b^n c^n d^n\}$ (izquierda) y configuraciones de dicho autómata durante el análisis de *aabbccdd* (derecha)

Ejemplo 7.2 El autómata a pila embebido ascendente sin estados definido por la tupla $(\{a, b, c, d\}, \{B, C, D, E, F, \$_0, \$_f\}, \Theta, \$_0, \$_f)$, donde Θ contiene las transiciones mostradas en la tabla 7.2, acepta el lenguaje $\{a^n b^n c^n d^n \mid n \geq 0\}$. En la misma tabla se muestra la secuencia de configuraciones que sigue el autómata para analizar correctamente la cadena de entrada *aabbccdd*. La primera columna muestra la transición aplicada, la segunda el contenido de la pila y la tercera la parte que resta por leer de la cadena de entrada. Obsérvese como la secuencia de pilas coincide, en orden inverso, con la secuencia de pilas de las configuraciones del autómata a pila embebido sin estados de la tabla 6.2. ¶

Teorema 7.4 *Mediante la utilización de un conjunto de transiciones SWAP, PUSH, POP, UNWRAP-A, UNWRAP-B y WRAP es posible emular transiciones complejas del tipo*

$$[F_k \dots [F_{i+1}, DC_1 \dots C_m, [F_i \dots [F_1 \stackrel{a}{\longmapsto} \ DB$$

tal que su aplicación da lugar a un paso de derivación

$$\Upsilon[F_k \dots [F_{i+1}[\alpha DC_1 \dots C_m[F_i \dots [F_1 \ \vdash \ \Upsilon[\alpha DB$$

donde

- $a \in V_T \cup \{\epsilon\}$.

- $0 \leq m \leq 2$.
- $B, C_1, \ldots, C_2, F_1, \ldots, F_k, \in V_S$.
- *Si* $m = 0$ *entonces* $D \in V_S$.
- *Si* $m > 0$ *entonces* $D = \epsilon$.

Demostración:

Realizaremos una prueba constructiva creando un procedimiento que permita emular el tipo de transiciones propuesto en el teorema 7.4 a partir de las transiciones de los autómatas a pila embebidos ascendentes sin estados. Para ello precisamos nuevos símbolos de pila ∇_j y X', donde $i + 1 \leq j \leq k$ y $X \in V_S$. La emulación se realizará en tres fases: la primera se encargará de eliminar las pilas $[F_i \ldots [F_1$, la segunda de obtener la pila $[\alpha D B$ y la tercera se encargará de eliminar las pilas $[F_k \ldots [F_{i+1}$.

Fase 1 Partimos de una configuración

$$(\Upsilon [F_k \ldots [F_{i+1} [\alpha D C_1 \ldots C_m, [F_i \ldots [F_1, aw)$$

donde $D \in V_S$ si $m = 0$ y $D = \epsilon$ en otro caso. Comenzamos la emulación mediante la creación de una transición que sustituye F_1 por ∇_1

$$F_1 \longmapsto \nabla_1$$

Para eliminar las $i - 1$ pilas de la cima crearemos las siguientes $i - 1$ transiciones

$$[F_{j+1}, \nabla_j \longmapsto \nabla_{j+1}$$

donde $1 \leq j \leq i - 1$.
En el caso de que $i = 0$, esto es $[F_i \ldots [F_1 = \epsilon$, esta fase constaría únicamente de la transición

$$X \longmapsto X, [\nabla_0$$

donde $X = D$ si $m = 0$, $X = C_1$ si $m = 1$ y $X = C_2$ si $m = 2$.
Como resultado de la aplicación de esta fase obtendremos una configuración

$$(\Upsilon [F_k \ldots [F_{i+1} [\alpha D C_1 \ldots C_m [\nabla_i, aw)$$

Fase 2 Las transiciones de esta fase dependen del valor de m, por lo que tenemos tres posibilidades:

- Si $m = 2$ se trata de reemplazar $C_1 C_2$ por B:

$$C_2 [\nabla_i \longmapsto \nabla_i'$$

$$C_1 \nabla_i' \longmapsto B'$$

- Si $m = 1$ nos encontramos ante el cambio de C_1 por B:

$$C_1 [\nabla_i \longmapsto \nabla_i'$$

$$\nabla_i' \longmapsto B'$$

- Si $m = 0$ se trata de apilar B:

$$D [\nabla_i \longmapsto D'$$

$$D' \longmapsto D' B'$$

Después de esta fase obtenemos una configuración

$$(\Upsilon\lfloor F_k \ldots \lfloor F_{i+1} \lfloor \alpha D' B' aw)$$

donde $D' = \epsilon$ si $m > 0$.

Fase 3 Para iniciar esta fase precisamos de una transición

$$\lfloor F_{i+1}, B' \longmapsto \nabla_{i+1}$$

Para crear las las restantes $k - i - 1$ pilas unitarias utilizaremos $k - i - 1$ transiciones de la forma

$$\lfloor F_{j+1}, \nabla_j \longmapsto \nabla_{j+1}$$

donde $i + 1 \leq j < k$, para finalizar con la transición

$$\nabla_k \overset{a}{\longmapsto} B$$

En el caso de que $k = i$, esto es, $\lfloor F_k \ldots \lfloor F_{i+1} = \epsilon$, esta fase constaría únicamente de la transición

$$B' \longmapsto B$$

Tras esta fase obtendremos la configuración

$$(\Upsilon\lfloor \alpha D' B, w)$$

donde $D' = \epsilon$ si $m > 0$. En el caso de que $m = 0$, para emular completamente la transición deseada deberemos crear una transición

$$D'X \longmapsto Y$$

por cada transición $DX \longmapsto Y$ presente en el BEPDA.

□

Ejemplo 7.3 La transición $\lfloor E \lfloor F, BC, \lfloor G \lfloor H \overset{a}{\longmapsto} B$ se emula mediante el conjunto de transiciones mostrado en la tabla 7.3. La misma tabla muestra el resultado de aplicar las transiciones de la tabla 7.3 a una configuración $(\Upsilon\lfloor \alpha B, aw)$. La primera columna muestra la transición aplicada, la segunda muestra el contenido de la pila del BEPDA y la tercera la parte de la cadena de entrada que falta por leer. ¶

7.4 Equivalencia entre autómatas a pila embebidos sin estados y con estados

Para establecer la equivalencia entre las versiones con estados y sin estados de los autómatas a pila embebidos ascendentes, haremos uso del teorema 7.3, que posibilita la utilización de transiciones complejas en los BEPDA con estados, y del teorema 7.4, que establece una forma normal para las transiciones de los BEPDA sin estados.

(a)	$H \longmapsto \nabla_1$
(b)	$[G, \nabla_1 \longmapsto \nabla_2$
(c)	$C, [\nabla_2 \longmapsto \nabla'_2$
(d)	$B\nabla'_2 \longmapsto B'$
(e)	$[F, B' \longmapsto \nabla_3$
(f)	$[E, \nabla_3 \longmapsto \nabla_4$
(g)	$\nabla_4 \overset{a}{\longmapsto} B$

	$\Upsilon[E[F[\alpha BC[G[H$	aw
(a)	$\Upsilon[E[F[\alpha BC[G[\nabla_1$	aw
(b)	$\Upsilon[E[F[\alpha BC[\nabla_2$	aw
(c)	$\Upsilon[E[F[\alpha B\nabla'_2$	aw
(d)	$\Upsilon[E[F[\alpha B'$	aw
(e)	$\Upsilon[E[\alpha\nabla_3$	aw
(f)	$\Upsilon\alpha\nabla_4$	aw
(g)	$\Upsilon\alpha B$	w

Tabla 7.3: Normalización de una transición compleja en un BEPDA (izquierda) y emulación de la misma (derecha)

Teorema 7.5 *Para todo BEPDA sin estados $\mathcal{A}$, existe un BEPDA con estados $\mathcal{A}'$ tal que el lenguaje aceptado por $\mathcal{A}$ es igual al lenguaje aceptado por $\mathcal{A}'$.*

Demostración:

Sea $\mathcal{A} = (V_T, V_S, \Theta, \$_0, \$_f)$ un BEPDA sin estados. El BEPDA con estados $\mathcal{A}' = (Q, V_T, V_S, \delta, q_0, Q_F, \$_f)$ acepta el mismo lenguaje (por pila vacía) que $\mathcal{A}$ si las transiciones en δ se obtienen mediante una traducción adecuada de las transiciones en Θ. Consideremos los posibles casos:

SWAP: Una transición $C \overset{a}{\longmapsto} F$ se traduce por una transición $(q, F) \in \delta(q, a, \epsilon, C, \epsilon)$

PUSH: Una transición $C \overset{a}{\longmapsto} CF$ se traduce por una transición $(q', C) \in \delta(q, a, \epsilon, C, \epsilon)$ más una transición $(q, F) \in \delta(q', \epsilon, \epsilon, \epsilon, \epsilon)$, donde q' es un estado que sólo se utiliza en estas dos transiciones.

POP: Una transición $CF \overset{a}{\longmapsto} G$ se traduce por una transición $(q, G) \in \delta(q, a, \epsilon, CF, \epsilon)$

UNWRAP-A: Una transición $C, [F \overset{a}{\longmapsto} G$ se traduce por una transición $(q, G) \in \delta(q, a, \epsilon, C, [F)$.

UNWRAP-B: Una transición $[C, F \overset{a}{\longmapsto} G$ se traduce por una transición $(q, G) \in \delta(q, a, [C, F, \epsilon)$.

WRAP: Una transición $C \overset{a}{\longmapsto} C, [F$ se traduce por una transición $(q'', C) \in \delta(q, a, \epsilon, C, \epsilon)$ más una transición $(q, [F) \in \delta(q'', a, \epsilon, \epsilon, \epsilon)$, donde q'' es un estado que sólo se utiliza en estas dos transiciones.

Adicionalmente, deberemos considerar la transición

$$(q', \$_f) \in \delta(q, \epsilon, \epsilon, \$_0, [\$_f)$$

que obtiene una pila $[\$_f$ cuando se alcanza la configuración $(q, [\$_0[\$_f, \epsilon)$, equivalente a la configuración final del BEPDA sin estados.

El conjunto Q estará formado por el estado q y todos los estados q' y q'' utilizados en la traducción de transiciones POP y WRAP. □

Teorema 7.6 *Para todo BEPDA con estados $\mathcal{A}$, existe un BEPDA sin estados $\mathcal{A}'$ tal que el lenguaje aceptado por $\mathcal{A}$ es igual al lenguaje aceptado por $\mathcal{A}'$.*

Demostración:

Dado un un BEPDA con estados $\mathcal{A} = (Q, V_T, V_S, \delta, q_0, Q_F, \$_f)$ construiremos un BEPDA sin estados $\mathcal{A}' = (V_T, V_S', \Theta, \$_0', \$_f')$ que acepte el mismo lenguaje es aceptado por $\mathcal{A}$ por pila vacía. El conjunto V_S' estará formado por pares $\langle Z, q \rangle$, donde $Z \in V_S \cup \{-\}$ y $q \in Q \cup \{-\}$, y por los elementos inicial $\$_0' = \langle -, q_0 \rangle$ y final $\$_f' = \langle \$_f, - \rangle$.

Las transiciones en Θ tendrán el formato de las transiciones descritas en el teorema 7.4 y serán el resultado de traducir las transiciones en δ. Supondremos que las transiciones de $\mathcal{A}$ están en la forma normal definida en el teorema 7.3. Consideremos cada uno de los posibles casos:

- Una transición
$$(q', [Z) \in \delta(q, a, \epsilon, \epsilon, \epsilon)$$
se traduce por una transición
$$\langle B, q \rangle \longmapsto \langle B, q \rangle, [\langle Z, q' \rangle$$
para todo $B \in V_S \cup \{-\}$.
- Una transición
$$(q', Z) \in \delta(q, a, [Z_k' \ldots [Z_{i+1}', Z_1, [Z_i' \ldots [Z_1')$$
se traduce por transiciones
$$[\langle F_k, q_k \rangle \ldots [\langle F_{i+1}, q_{i+1} \rangle, \langle Z_1, q'' \rangle, [\langle F_i, q_i \rangle \ldots [\langle F_1, q_1 \rangle \overset{a}{\longmapsto} \langle Z, q' \rangle$$
para todo $q_1, \ldots, q_k, q'' \in Q$.
- Una transición
$$(q', Z) \in \delta(q, a, [Z_k' \ldots [Z_{i+1}', \epsilon, [Z_i' \ldots [Z_1')$$
se traduce por transiciones
$$[\langle F_k, q_k \rangle \ldots [\langle F_{i+1}, q_{i+1} \rangle, \langle D, q'' \rangle, [\langle F_i, q_i \rangle \ldots [\langle F_1, q_1 \rangle \overset{a}{\longmapsto} \langle D, q'' \rangle \langle Z, q' \rangle$$
para todo $D \in V_S$ y $q_1, \ldots, q_k, q'' \in Q$. Es importante resaltar que esta traducción es correcta porque en toda configuración válida de un BEPDA con estados cada pila debe contener al menos un elemento, lo cual garantiza la existencia de D.
- Una transición
$$(q', Z) \in \delta(q, a, [Z_k' \ldots [Z_{i+1}', Z_2 Z_1, [Z_i' \ldots [Z_1')$$
se traduce por transiciones
$$[\langle F_k, q_k \rangle \ldots [\langle F_{i+1}, q_{i+1} \rangle, \langle Z_2, q''' \rangle \langle Z_1, q'' \rangle, [\langle F_i, q_i \rangle \ldots [\langle F_1, q_1 \rangle \overset{a}{\longmapsto} \langle Z, q' \rangle$$
para todo $q_1, \ldots, q_k, q'', q''' \in Q$.

La transición inicial
$$\langle -, - \rangle \longmapsto \langle -, - \rangle [\langle -, q_0 \rangle$$
configura la pila para que puedan comenzar a ser aplicadas las transiciones definidas anteriormente.

La configuración final de $\mathcal{A}'$ es $(\langle -, - \rangle [\langle \$_f, q \rangle, \epsilon)$, que se corresponde con la configuración final $(q, [\$_f, \epsilon)$ de $\mathcal{A}$. □

Corolario 7.7 *Los BEPDA sin estados son equivalentes a los BEPDA con estados.*

Demostración:

La veracidad del enunciado se deriva directamente de los teoremas 7.5 y 7.6. □

7.5 Esquemas de compilación de gramáticas independientes del contexto

De forma análoga al caso de los EPDA (sección 6.5), tomamos una configuración $(\$_0 B_1 B_2 \ldots B_n, w)$ de un autómata a pila como equivalente a una configuración $([\$_0 [B_1 [B_1 \ldots [B_1, w)$ de un BEPDA, con lo cual las pilas unitarias pasan a representar el papel de los símbolos de pila. Con respecto a las transiciones, tenemos que:

- Las transiciones SWAP $C \longmapsto F$ permanecen sin cambios.
- Las transiciones PUSH $C \longmapsto CF$ del autómata a pila son reemplazadas por transiciones WRAP $C \longmapsto C, [F$ en el autómata a pila embebido ascendente.
- Las transiciones POP $CF \longmapsto G$ del autómata a pila son reemplazadas por transiciones UNWRAP-A $C, [F \longmapsto G$ en el autómata a pila embebido ascendente.

Un autómata a pila embebido ascendente cuyas configuraciones sólo contengan pilas unitarias y que sólo utilice transiciones SWAP, WRAP y UNWRAP-A, será equivalente a un autómata a pila cuyas configuraciones se obtendrán a partir de las configuraciones del EPDA eliminando los símbolos $[$. El conjunto de transiciones contendrá las transiciones SWAP y el resultado de convertir las transiciones WRAP en transiciones PUSH y las transiciones UNWRAP-A en transiciones POP.

Puesto que las transiciones WRAP de los BEPDA coinciden con las transiciones WRAP-A de los EPDA y que las transiciones UNWRAP-A de los BEPDA coinciden con las transiciones UNWRAP de los EPDA, el resultado de traducir un autómata a pila a un EPDA coincide con el resultado de traducir dicho autómata a pila a un BEPDA. En consecuencia, el esquema de compilación genérico 7.1 de gramáticas independientes del contexto en autómatas a pila embebidos ascendentes coincide con el esquema de compilación genérico 6.1 de gramáticas independientes del contexto en autómatas a pila embebidos.

Esquema de compilación 7.1 El esquema de compilación genérico de una gramática independiente del contexto en un autómata a pila embebido ascendente queda definido por el siguiente conjunto de reglas y los elementos inicial $\$_0$ y final $\overleftarrow{S}$.

$$
\begin{array}{lll}
\textbf{[INIT]} & \$_0 \longmapsto \$_0 \ [\, \nabla_{0,0} & \\
\textbf{[CALL]} & \nabla_{r,s} \longmapsto \nabla_{r,s} \ [\, \overrightarrow{A_{r,s+1}} & \\
\textbf{[SEL]} & \overrightarrow{A_{r,0}} \longmapsto \nabla_{r,0} & r \neq 0 \\
\textbf{[PUB]} & \nabla_{r,n_r} \longmapsto \overleftarrow{A_{r,0}} & \\
\textbf{[RET]} & \nabla_{r,s} \ [\, \overleftarrow{A_{r,s+1}} \longmapsto \nabla_{r,s+1} & \\
\textbf{[SCAN]} & \overrightarrow{A_{r,0}} \overset{a}{\longmapsto} \overleftarrow{A_{r,0}} & A_{r,0} \to a
\end{array}
$$

§

7.6 Esquemas de compilación de gramáticas de adjunción de árboles

Los autómatas a pila embebidos ascendentes sólo pueden ser utilizados para definir esquemas de compilación de gramáticas de adjunción de árboles en los que las las derivaciones se reconocen de modo ascendente, esto es, el reconocimiento de una adjunción se inicia en el nodo pie y se termina en el nodo raíz de un árbol auxiliar. En consecuencia, un esquema de compilación de una TAG en un BEPDA debe realizar las tareas siguientes:

- Recorrer los nodos de los árboles elementales.
- En el caso de la adjunción del árbol auxiliar β en un nodo N^γ:
  - Una vez recorrido el subárbol de γ que cuelga del nodo N^γ, suspender el recorrido de γ y comenzar el recorrido de β a partir de su nodo pie.
  - Al alcanzar el nodo raíz de β, suspender el recorrido de este último para continuar el recorrido del superárbol de γ a partir del nodo N^γ.

El recorrido de un árbol elemental es equivalente al recorrido del conjunto de producciones independientes del contexto que lo componen, por lo que podemos utilizar para esta tarea las reglas de compilación **[CALL]**, **[SEL]**, **[PUB]**y **[RET]** definidas en el esquema de compilación 6.1. Al igual que en el capítulo 3, consideraremos una producción adicional $\top^\beta \rightarrow \mathbf{R}^\alpha$ para cada árbol inicial α y dos producciones adicionales para cada árbol auxiliar β: $\top^\beta \rightarrow \mathbf{R}^\beta$ y $\mathbf{F}^\beta \rightarrow \bot^\beta$, donde $\mathbf{R}^\beta$ y $\mathbf{F}^\beta$ se refieren a los nodos raíz y pie de β, respectivamente.

Para poder tratar las adjunciones deberemos ser capaces de trasmitir, desde el nodo pie a la raíz de un árbol auxiliar, el nodo sobre el cual se ha adjuntado dicho árbol auxiliar. En el caso de que se realicen otras adjunciones en nodos de la espina, deberemos ir apilando los nodos de adjunción en los cuales se aplican. Utilizaremos las propias pilas del BEPDA para almacenar las pilas de adjunciones pendientes. Para ello, dotaremos de la siguiente semántica a las pilas del BEPDA: dada una pila $[\alpha C$, el elemento C de la cima nos informa del punto en el que se encuentra el recorrido de un árbol elemental mientras que la parte restante α contiene la pila de adjunciones pendientes en dicho punto. La figura 6.6 muestra de modo intuitivo las diferentes reglas de compilación, que en el caso de un BEPDA realizan las funciones siguientes:

- La regla de compilación **[SRET]** será la encargada de crear las transiciones que propaguen la pila de adjunciones pendientes a través de la espina de los árboles auxiliares, desde el pie hasta la raíz.
- La regla **[SCALL]** es una regla **[CALL]** que trata con elementos de la espina de un árbol auxiliar. Se define con el fin de formar un par **[SCALL]**–**[SRET]** análogo al par **[CALL]**–**[RET]**.
- Al llegar al nodo de adjunción N^γ, deberemos apilar dicho nodo en la pila de adjunciones pendientes y pasar al pie del árbol auxiliar β. La regla de compilación **[FRET]** se encargará de crear las transiciones adecuadas para tal fin.
- Al llegar al nodo raíz de β, deberemos eliminar el nodo N^γ de la pila de adjunciones pendientes y continuar el recorrido en el superárbol de γ a partir de dicho nodo. De crear las transiciones necesarias para este fin se encargará la regla de compilación **[FRET]**.
- La regla **[FCALL]** es un tipo especial de regla **[CALL]** que permite pasar del pie del árbol auxiliar β a un nodo de un árbol elemental que pueda servir como nodo de adjunción.

[INIT]	$\$_0 \longmapsto \$_0 \, [\nabla^{\alpha}_{0,0}$	$\alpha \in \boldsymbol{I}$
[CALL]	$\nabla^{\gamma}_{r,s} \longmapsto \nabla^{\gamma}_{r,s}, \, [\overrightarrow{N^{\gamma}_{r,s+1}}$	$N^{\gamma}_{r,s+1} \notin \text{espina}(\gamma), \;\; \mathbf{nil} \in \text{adj}(N_{r,s+1})$
[SCALL]	$\nabla^{\beta}_{r,s} \longmapsto \nabla^{\beta}_{r,s} \, [\overrightarrow{N^{\beta}_{r,s+1}}$	$N^{\beta}_{r,s+1} \in \text{espina}(\beta), \;\; \mathbf{nil} \in \text{adj}(N^{\beta}_{r,s+1})$
[SEL]	$\overrightarrow{N_{r,0}}^{\gamma} \longmapsto \nabla^{\gamma}_{r,0}$	
[PUB]	$\nabla^{\gamma}_{r,n_r} \longmapsto \overleftarrow{N^{\gamma}_{r,0}}$	
[RET]	$\nabla^{\gamma}_{r,s}, \, [\overleftarrow{N^{\gamma}_{r,s+1}} \longmapsto \nabla^{\gamma}_{r,s+1}$	$N^{\gamma}_{r,s+1} \notin \text{espina}(\gamma), \;\; \mathbf{nil} \in \text{adj}(N_{r,s+1})$
[SRET]	$[\nabla^{\beta}_{r,s}, \; \overleftarrow{N^{\beta}_{r,s+1}} \longmapsto \nabla^{\beta}_{r,s+1}$	$N^{\beta}_{r,s+1} \in \text{espina}(\beta), \;\; \mathbf{nil} \in \text{adj}(N^{\beta}_{r,s+1})$
[SCAN]	$\overrightarrow{N^{\gamma}_{r,0}} \overset{a}{\longmapsto} \overleftarrow{N^{\gamma}_{r,0}}$	$N^{\gamma}_{r,0} \to a$
[ACALL]	$\nabla^{\gamma}_{r,s} \longmapsto \nabla^{\gamma}_{r,s}, \, [\overrightarrow{\top^{\beta}}$	$\beta \in \text{adj}(N^{\gamma}_{r,s+1})$
[ARET-a]	$[\nabla^{\gamma}_{r,s}, \; \overleftarrow{\top^{\beta}} \longmapsto \top$	$\beta \in \text{adj}(N^{\gamma}_{r,s+1})$
[ARET-b]	$\Delta^{\gamma}_{r,s} \;\; \top \longmapsto \nabla^{\gamma}_{r,s+1}$	
[FCALL]	$\nabla^{\beta}_{f,0} \longmapsto \nabla^{\beta}_{f,0}, \, [\overrightarrow{N^{\gamma}_{r,s+1}}$	$N^{\beta}_{f,0} = \mathbf{F}^{\beta}, \;\; \beta \in \text{adj}(N^{\gamma}_{r,s+1})$
[FRET-a]	$[\nabla^{\beta}_{f,0}, \; \overleftarrow{N^{\gamma}_{r,s+1}} \longmapsto \Delta^{\gamma}_{r,s}$	$N^{\beta}_{f,0} = \mathbf{F}^{\beta}, \;\; \beta \in \text{adj}(N^{\gamma}_{r,s+1})$
[FRET-b]	$\Delta^{\gamma}_{r,s} \longmapsto \Delta^{\gamma}_{r,s} \; \nabla^{\beta}_{f,1}$	$N^{\beta}_{f,0} = \mathbf{F}^{\beta}, \;\; \beta \in \text{adj}(N^{\gamma}_{r,s+1})$

Tabla 7.4: Reglas del esquema de compilación genérico de TAG en BEPDA

- La regla **[ACALL]** es un tipo especial de regla **[CALL]** que permite pasar de un nodo de un árbol elemental a la raíz de un árbol auxiliar que pueda ser adjuntado en dicho nodo.

A continuación definimos un esquema de compilación genérico, derivado del esquema de compilación 6.1, en el cual se ha parametrizado la información concerniente al recorrido de los árboles elementales.

Esquema de compilación 7.2 El esquema de compilación genérico de una gramática de adjunción de árboles en un autómata a pila embebido ascendente queda definido por el conjunto de reglas mostrado en la figura 6.4 y los elementos inicial $\$_0$ y final $\overleftarrow{S}$. Es interesante señalar que las pilas de adjunciones pendientes no almacenan directamente los nodos $N^{\gamma}_{r,s+1}$ en los que se realizaron las adjunciones sino el elemento $\nabla^{\gamma}_{r,s}$ que indica el punto en el que se lanzó la adjunción, almacenado en las pilas bajo la forma $\Delta^{\gamma}_{r,s}$ para evitar confusiones. Podemos ver un símbolo Δ como un símbolo ∇ en espera de la finalización de una adjunción.

Las reglas de compilación **[ARET]** y **[FRET]** podrían escribirse alternativamente:

$$\textbf{[ARET]} \quad \left[\nabla^{\gamma}_{r,s},\ \Delta^{\gamma}_{r,s}\ \overleftarrow{\top^{\beta}} \longmapsto \nabla^{\gamma}_{r,s+1} \qquad \beta \in \text{adj}(N^{\gamma}_{r,s+1})\right.$$

$$\textbf{[FRET]} \quad \left[\nabla^{\beta_1}_{f_1,0},\ \overleftarrow{N^{\gamma}_{r,s+1}} \longmapsto \Delta^{\gamma}_{r,s}\ \nabla^{\beta_2}_{f_2,1} \quad N^{\beta_1}_{f_1,0} = \mathbf{F}^{\beta_1},\quad N^{\beta_2}_{f_2,0} = \mathbf{F}^{\beta_2},\quad \beta_1, \beta_2 \in \text{adj}(N^{\gamma}_{r,s+1})\right.$$

pero como las transiciones resultantes no son transiciones elementales, hemos tenido que descomponer dichas reglas de compilación en dos pares de reglas: **[ARET-a]** más **[ARET-b]** y **[FRET-a]** más **[FRET-b]**. §

El esquema de compilación genérico establece que las adjunciones se tienen que reconocer de modo ascendente, puesto que al pasar al pie de un árbol auxiliar se apila el nodo de adjunción en la pila de adjunciones pendientes y al llegar al nodo raíz dicho nodo se saca de la pila de adjunciones pendientes. Es interesante remarcar que la regla de compilación **[RET]** utiliza una transición UNWRAP-A con el significado de "elimina una pila unitaria que indica un nodo del árbol γ que no pertenece a la espina y por tanto debe tener una una pila vacía de adjunciones pendientes", mientras que la regla de compilación **[SRET]** utiliza una transición UNWRAP-B a la que dota del significado "elimina una pila que indica el reconocimiento de un nodo perteneciente a la espina de un árbol auxiliar β pero preserva la lista de adjunciones pendientes". En la figura 7.5 se observa cómo la parte sombreada (en nuestro caso, la pila de adjunciones pendientes) permanece en su posición original en la operación UNWRAP-A mientras que en la operación UNWRAP-B la parte sombreada es pasada a la nueva pila en la cima.

El esquema de compilación genérico no establece ninguna restricción sobre la estrategia utilizada para recorrer de los árboles elementales. A continuación se definen, de acuerdo con la tabla 5.2, los esquemas de compilación correspondientes a tres estrategias particulares aplicadas al recorrido de los árboles elementales: ascendente, Earley y descendente.

7.6.1 Estrategia descendente

Esquema de compilación 7.3 El esquema de compilación de una gramática de adjunción de árboles en un autómata a pila embebido ascendente que incorpora una estrategia descendente para el recorrido de los árboles elementales queda definido por el conjunto de reglas mostrado en la tabla 7.5 y los elementos inicial $\$_0$ y final $\Box$. §

7.6.2 Estrategia Earley

Esquema de compilación 7.4 El esquema de compilación de una gramática de adjunción de árboles en un autómata a pila embebido ascendente que incorpora una estrategia descendente para el recorrido de los árboles elementales queda definido por el conjunto de reglas mostrado en la tabla 7.6 y los elementos inicial $\$_0$ y final $\overline{\overline{S}}$. §

7.6.3 Estrategia ascendente

Esquema de compilación 7.5 El esquema de compilación de una gramática de adjunción de árboles en un autómata a pila embebido ascendente que incorpora una estrategia ascendente

[INIT]	$\$_0 \longmapsto \$_0 \lfloor \nabla^{\alpha}_{0,0}$	$\alpha \in \boldsymbol{I}$
[CALL]	$\nabla^{\gamma}_{r,s} \longmapsto \nabla^{\gamma}_{r,s},\ \lfloor N^{\gamma}_{r,s+1}$	$N^{\gamma}_{r,s+1} \notin \text{espina}(\gamma),\ \ \mathbf{nil} \in \text{adj}(N_{r,s+1})$
[SCALL]	$\nabla^{\beta}_{r,s} \longmapsto \nabla^{\beta}_{r,s} \lfloor N^{\beta}_{r,s+1}$	$N^{\beta}_{r,s+1} \in \text{espina}(\beta),\ \ \mathbf{nil} \in \text{adj}(N^{\beta}_{r,s+1})$
[SEL]	$N^{\gamma}_{r,0} \longmapsto \nabla^{\gamma}_{r,0}$	
[PUB]	$\nabla^{\gamma}_{r,n_r} \longmapsto \Box$	
[RET]	$\nabla^{\gamma}_{r,s},\ \lfloor \Box \longmapsto \nabla^{\gamma}_{r,s+1}$	$N^{\gamma}_{r,s+1} \notin \text{espina}(\gamma),\ \ \mathbf{nil} \in \text{adj}(N_{r,s+1})$
[SRET]	$\lfloor \nabla^{\beta}_{r,s},\ \Box \longmapsto \nabla^{\beta}_{r,s+1}$	$N^{\beta}_{r,s+1} \in \text{espina}(\beta),\ \ \mathbf{nil} \in \text{adj}(N^{\beta}_{r,s+1})$
[SCAN]	$N^{\gamma}_{r,0} \overset{a}{\longmapsto} \Box$	$N^{\gamma}_{r,0} \to a$
[ACALL]	$\nabla^{\gamma}_{r,s} \longmapsto \nabla^{\gamma}_{r,s},\ \lfloor \top^{\beta}$	$\beta \in \text{adj}(N^{\gamma}_{r,s+1})$
[ARET-a]	$\lfloor \nabla^{\gamma}_{r,s},\ \Box \longmapsto \top$	$\text{adj}(N^{\gamma}_{r,s+1}) \neq \{\mathbf{nil}\}$
[ARET-b]	$\Delta^{\gamma}_{r,s}\ \ \top \longmapsto \nabla^{\gamma}_{r,s+1}$	$\text{adj}(N^{\gamma}_{r,s+1}) \neq \{\mathbf{nil}\}$
[FCALL]	$\nabla^{\beta}_{f,0} \longmapsto \nabla^{\beta}_{f,0},\ \lfloor N^{\gamma}_{r,s+1}$	$N^{\beta}_{f,0} = \mathbf{F}^{\beta},\ \ \beta \in \text{adj}(N^{\gamma}_{r,s+1})$
[FRET-a]	$\lfloor \nabla^{\beta}_{f,0},\ \Box \longmapsto \Delta^{\gamma}_{r,s}$	$N^{\beta}_{f,0} = \mathbf{F}^{\beta},\ \ \beta \in \text{adj}(N^{\gamma}_{r,s+1})$
[FRET-b]	$\Delta^{\gamma}_{r,s} \longmapsto \Delta^{\gamma}_{r,s}\ \nabla^{\beta}_{f,1}$	$N^{\beta}_{f,0} = \mathbf{F}^{\beta},\ \ \beta \in \text{adj}(N^{\gamma}_{r,s+1})$

Tabla 7.5: Reglas del esquema de compilación descendente de TAG en BEPDA

[INIT]	$\$_0 \longmapsto \$_0 \lfloor \nabla^{\alpha}_{0,0}$	$\alpha \in \boldsymbol{I}$
[CALL]	$\nabla^{\gamma}_{r,s} \longmapsto \nabla^{\gamma}_{r,s},\ \lfloor \overline{N^{\gamma}_{r,s+1}}$	$N^{\gamma}_{r,s+1} \notin \text{espina}(\gamma),\ \ \mathbf{nil} \in \text{adj}(N_{r,s+1})$
[SCALL]	$\nabla^{\beta}_{r,s} \longmapsto \nabla^{\beta}_{r,s} \lfloor \overline{N^{\beta}_{r,s+1}}$	$N^{\beta}_{r,s+1} \in \text{espina}(\beta),\ \ \mathbf{nil} \in \text{adj}(N^{\beta}_{r,s+1})$
[SEL]	$\overline{N_{r,0}}^{\gamma} \longmapsto \nabla^{\gamma}_{r,0}$	
[PUB]	$\nabla^{\gamma}_{r,n_r} \longmapsto \overline{\overline{N^{\gamma}_{r,0}}}$	
[RET]	$\nabla^{\gamma}_{r,s},\ \lfloor \overline{\overline{N^{\gamma}_{r,s+1}}} \longmapsto \nabla^{\gamma}_{r,s+1}$	$N^{\gamma}_{r,s+1} \notin \text{espina}(\gamma),\ \ \mathbf{nil} \in \text{adj}(N_{r,s+1})$
[SRET]	$\lfloor \nabla^{\beta}_{r,s},\ \overline{\overline{N^{\beta}_{r,s+1}}} \longmapsto \nabla^{\beta}_{r,s+1}$	$N^{\beta}_{r,s+1} \in \text{espina}(\beta),\ \ \mathbf{nil} \in \text{adj}(N^{\beta}_{r,s+1})$
[SCAN]	$\overline{N^{\gamma}_{r,0}} \overset{a}{\longmapsto} \overline{\overline{N^{\gamma}_{r,0}}}$	$N^{\gamma}_{r,0} \to a$
[ACALL]	$\nabla^{\gamma}_{r,s} \longmapsto \nabla^{\gamma}_{r,s},\ \lfloor \overline{\top^{\beta}}$	$\beta \in \text{adj}(N^{\gamma}_{r,s+1})$
[ARET-a]	$\lfloor \nabla^{\gamma}_{r,s},\ \overline{\overline{\top^{\beta}}} \longmapsto \top$	$\text{adj}(N^{\gamma}_{r,s+1}) \neq \{\mathbf{nil}\}$
[ARET-b]	$\Delta^{\gamma}_{r,s}\ \ \top \longmapsto \nabla^{\gamma}_{r,s+1}$	$\text{adj}(N^{\gamma}_{r,s+1}) \neq \{\mathbf{nil}\}$
[FCALL]	$\nabla^{\beta}_{f,0} \longmapsto \nabla^{\beta}_{f,0},\ \lfloor \overline{N^{\gamma}_{r,s+1}}$	$N^{\beta}_{f,0} = \mathbf{F}^{\beta},\ \ \beta \in \text{adj}(N^{\gamma}_{r,s+1})$
[FRET-a]	$\lfloor \nabla^{\beta}_{f,0},\ \overline{\overline{N^{\gamma}_{r,s+1}}} \longmapsto \Delta^{\gamma}_{r,s}$	$N^{\beta}_{f,0} = \mathbf{F}^{\beta},\ \ \beta \in \text{adj}(N^{\gamma}_{r,s+1})$
[FRET-b]	$\Delta^{\gamma}_{r,s} \longmapsto \Delta^{\gamma}_{r,s}\ \nabla^{\beta}_{f,1}$	$N^{\beta}_{f,0} = \mathbf{F}^{\beta},\ \ \beta \in \text{adj}(N^{\gamma}_{r,s+1})$

Tabla 7.6: Reglas del esquema de compilación Earley de TAG en BEPDA

[INIT]	$\$_0 \longmapsto \$_0 \ [\nabla^{\alpha}_{0,0}$	$\alpha \in \boldsymbol{I}$
[CALL]	$\nabla^{\gamma}_{r,s} \longmapsto \nabla^{\gamma}_{r,s}, \ [\square$	$N^{\gamma}_{r,s+1} \not\in \text{espina}(\gamma), \ \ \mathbf{nil} \in \text{adj}(N_{r,s+1})$
[SCALL]	$\nabla^{\beta}_{r,s} \longmapsto \nabla^{\beta}_{r,s} \ [\square$	$N^{\beta}_{r,s+1} \in \text{espina}(\beta), \ \ \mathbf{nil} \in \text{adj}(N^{\beta}_{r,s+1})$
[SEL]	$\square \longmapsto \nabla^{\gamma}_{r,0}$	
[PUB]	$\nabla^{\gamma}_{r,n_r} \longmapsto N^{\gamma}_{r,0}$	
[RET]	$\nabla^{\gamma}_{r,s}, \ [N^{\gamma}_{r,s+1} \longmapsto \nabla^{\gamma}_{r,s+1}$	$N^{\gamma}_{r,s+1} \not\in \text{espina}(\gamma), \ \ \mathbf{nil} \in \text{adj}(N_{r,s+1})$
[SRET]	$[\nabla^{\beta}_{r,s}, \ N^{\beta}_{r,s+1} \longmapsto \nabla^{\beta}_{r,s+1}$	$N^{\beta}_{r,s+1} \in \text{espina}(\beta), \ \ \mathbf{nil} \in \text{adj}(N^{\beta}_{r,s+1})$
[SCAN]	$\square \overset{a}{\longmapsto} N^{\gamma}_{r,0}$	$N^{\gamma}_{r,0} \to a$
[ACALL]	$\nabla^{\gamma}_{r,s} \longmapsto \nabla^{\gamma}_{r,s}, \ [\square$	$\beta \in \text{adj}(N^{\gamma}_{r,s+1})$
[ARET-a]	$[\nabla^{\gamma}_{r,s}, \ \top^{\beta} \longmapsto \top$	$\text{adj}(N^{\gamma}_{r,s+1}) \neq \{\mathbf{nil}\}$
[ARET-b]	$\Delta^{\gamma}_{r,s} \ \ \top \longmapsto \nabla^{\gamma}_{r,s+1}$	$\text{adj}(N^{\gamma}_{r,s+1}) \neq \{\mathbf{nil}\}$
[FCALL]	$\nabla^{\beta}_{f,0} \longmapsto \nabla^{\beta}_{f,0}, \ [\square$	$N^{\beta}_{f,0} = \mathbf{F}^{\beta}, \ \ \beta \in \text{adj}(N^{\gamma}_{r,s+1})$
[FRET-a]	$[\nabla^{\beta}_{f,0}, \ N^{\gamma}_{r,s+1} \longmapsto \Delta^{\gamma}_{r,s}$	$N^{\beta}_{f,0} = \mathbf{F}^{\beta}, \ \ \beta \in \text{adj}(N^{\gamma}_{r,s+1})$
[FRET-b]	$\Delta^{\gamma}_{r,s} \longmapsto \Delta^{\gamma}_{r,s} \ \nabla^{\beta}_{f,1}$	$N^{\beta}_{f,0} = \mathbf{F}^{\beta}, \ \ \beta \in \text{adj}(N^{\gamma}_{r,s+1})$

Tabla 7.7: Reglas del esquema de compilación ascendente de TAG en BEPDA

para el recorrido de los árboles elementales queda definido por el conjunto de reglas mostrado en la tabla 7.7 y los elementos inicial $\$_0$ y final S. §

7.7 Esquemas de compilación de gramáticas lineales de índices

Para el análisis de gramáticas lineales de índices mediante autómatas a pila embebidos utilizaremos la misma semántica utilizada en el caso de los EPDA: una pila $[\alpha B$ se corresponde con el símbolo $B[\alpha]$ de una gramática lineal de índices. En consecuencia, para emular una derivación de una gramática lineal de índices en un BEPDA, es preciso ir modificando las pilas del autómata de acuerdo con los cambios en las pilas de índices indicados por la gramática. De ello se encargan las reglas de compilación descritas en la tabla 7.8.

La novedad con respecto al esquema de compilación 7.1 para gramáticas independientes del contexto estriba en la presencia de las reglas de compilación **[SRET]**, que son las encargadas de transmitir las pilas de índices durante la fase ascendente del algoritmo de análisis. Una vez más, sale a relucir la dualidad existente entre los EPDA y los BEPDA, pues mientras en los primeros sólo es posible trasmitir las pilas de índices durante la fase predictiva o descendente, en los segundos las forma de las transiciones sólo permite que sean transmitidas en la fase ascendente.

Con respecto al tratamiento de los no-terminales, la forma de las transiciones de los BEPDA no imponen ninguna restricción sobre su tratamiento. Este hecho nos permite, de forma análoga al caso de los EPDA, definir un esquema de compilación genérico que será posteriormente adaptado para crear los esquemas de compilación correspondientes a las estrategias descendente, Earley y ascendente aplicadas a los no-terminales de la gramática lineal de índices.

Regla	Tarea
[INIT]	Inicia los cálculos a partir de la pila inicial.
[CALL]	Requiere el análisis de un determinado elemento gramatical $B[\,]$, lo cual implica situar en la cima de la pila principal una nueva pila $[B$.
[SEL]	Selecciona una producción.
[PUB]	Determina que una producción de la gramática ha sido completamente analizada.
[RET]	Continúa el proceso de análisis después de que se haya completado una producción que tiene el no-terminal B en su lado izquierdo, lo cual implica eliminar una pila $[B$ de la cima de la pila principal.
[SRET]	Continúa el proceso de análisis después de que se haya reconocido un elemento gramatical que es un hijo dependiente, lo que implica tener que pasarle la pila de índices con los cambios correspondientes.
[SCAN]	Reconoce los terminales que componen la cadena de entrada.

Tabla 7.8: Tipos de reglas de los esquemas de compilación de LIG en BEPDA

Esquema de compilación 7.6 El esquema de compilación genérico de una gramática lineal de índices en un autómata a pila embebido ascendente queda definido por el conjunto de reglas mostrado en la tabla 7.9 y los elementos inicial $\$_0$ y final $\overleftarrow{S}$.
Las reglas de compilación **[SRET-2]** y **[SRET-3]** podrían escribirse alternativamente de la forma:

$$\textbf{[SRET-2]} \quad [\nabla_{r,s}, \;\; \gamma' \; \overrightarrow{A_{r,s+1}} \longmapsto \nabla_{r,s+1} \qquad A_{r,0}[\circ\circ] \to \Upsilon_1 A_{r,s+1}[\circ\circ\gamma']\Upsilon_2$$

$$\textbf{[SRET-3]} \quad [\nabla_{r,s}, \;\; \overrightarrow{A_{r,s+1}} \longmapsto \gamma \;\; \nabla_{r,s+1} \qquad A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\circ\circ]\Upsilon_2$$

pero en tal caso las producciones involucradas no forman parte de la familia de transiciones elementales de los BEPDA. Esta circunstancia nos ha llevado a descomponer la regla de compilación **[SRET-2]** en dos reglas **[SRET-2a]** y **[SRET-2b]** y a descomponer la regla **[SRET-3]** en dos reglas **[SRET-3a]** y **[SRET-3b]**. Al igual que ocurrió en el caso de los EPDA, esta descomposición de las reglas de compilación conlleva como efecto colateral un cambio en las pilas del autómata, que pasarán a tener la forma $[\alpha B$, donde B será un no-terminal de la gramática lineal de índices y α estará formado por una sucesión de triples $\langle\gamma, r, s\rangle$, donde γ es un índices mientras que r y s señalan una posición s en una producción r. La proyección del primer componente de dicho triples proporciona la pila de índices asociada a B. Los componentes r y s sólo se utilizan en la regla de compilación **[SRET-3b]** mientras que son ignorados en **[SRET-2b]**. §

[INIT]	$\$_0 \longmapsto \$_0, [\nabla_{0,0}$	
[CALL]	$\nabla_{r,s} \longmapsto \nabla_{r,s},\ [\overrightarrow{A_{r,s+1}}$	$\boldsymbol{A_{r,0}} \to \Upsilon_1 \boldsymbol{A_{r,s+1}} \Upsilon_2$
[SEL]	$\overrightarrow{A_{r,0}} \longmapsto \nabla_{r,0}$	$r \neq 0$
[PUB]	$\nabla_{r,n_r} \longmapsto \overleftarrow{A_{r,0}}$	
[RET]	$\nabla_{r,s},\ [\overleftarrow{A_{r,s+1}} \longmapsto \nabla_{r,s+1}$	$\boldsymbol{A_{r,0}} \to \Upsilon_1 A_{r,s+1}[\,]\Upsilon_2$
[SRET-1]	$[\nabla_{r,s},\ \overleftarrow{A_{r,s+1}} \longmapsto \nabla_{r,s+1}$	$A_{r,0}[\circ\circ] \to \Upsilon_1 A_{r,s+1}[\circ\circ]\Upsilon_2$
[SRET-2a]	$[\nabla_{r,s},\ \overleftarrow{A_{r,s+1}} \longmapsto \nabla'_{r,s+1}$	$A_{r,0}[\circ\circ] \to \Upsilon_1 A_{r,s+1}[\circ\circ\gamma']\Upsilon_2$
[SRET-2b]	$\langle\gamma', t, u\rangle\ \nabla'_{r,s+1} \longmapsto \nabla_{r,s+1}$	$A_{r,0}[\circ\circ] \to \Upsilon_1 A_{r,s+1}[\circ\circ\gamma']\Upsilon_2$
[SRET-3a]	$[\nabla_{r,s},\ \overleftarrow{A_{r,s+1}} \longmapsto \langle\gamma, r, s+1\rangle$	$A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\circ\circ]\Upsilon_2$
[SRET-3b]	$\langle\gamma, r, s+1\rangle \longmapsto \langle\gamma, r, s+1\rangle\ \nabla_{r,s+1}$	$A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\circ\circ]\Upsilon_2$
[SCAN]	$\overrightarrow{A_{r,0}} \overset{a}{\longmapsto} \overleftarrow{A_{r,0}}$	$A_{r,0}[\,] \to a$

Tabla 7.9: Reglas del esquema de compilación genérico de LIG en BEPDA

7.7.1 Estrategia descendente

Esquema de compilación 7.7 El esquema de compilación con estrategia descendente de una gramática lineal de índices en un autómata a pila embebido ascendente queda definido por el conjunto de reglas de la tabla 6.10 y los elementos inicial $\$_0$ y final $\Box$. §

7.7.2 Estrategia Earley

Esquema de compilación 7.8 El esquema de compilación con estrategia Earley de una gramática lineal de índices en un autómata a pila embebido ascendente queda definido por el conjunto de reglas de la tabla 6.11 y los elementos inicial $\$_0$ y final $\overline{\overline{S}}$. §

7.7.3 Estrategia ascendente

Esquema de compilación 7.9 El esquema de compilación con estrategia ascendente de una gramática lineal de índices en un autómata a pila embebido ascendente queda definido por el conjunto de reglas de la tabla 6.12 y los elementos inicial $\$_0$ y final S. §

[INIT]	$\$_0 \longmapsto \$_0, [\nabla_{0,0}$	
[CALL]	$\nabla_{r,s} \longmapsto \nabla_{r,s},\ [A_{r,s+1}$	$\boldsymbol{A_{r,0}} \to \Upsilon_1 \boldsymbol{A_{r,s+1}} \Upsilon_2$
[SEL]	$A_{r,0} \longmapsto \nabla_{r,0}$	$r \neq 0$
[PUB]	$\nabla_{r,n_r} \longmapsto \Box$	
[RET]	$\nabla_{r,s},\ [\Box \longmapsto \nabla_{r,s+1}$	$\boldsymbol{A_{r,0}} \to \Upsilon_1 A_{r,s+1}[\,]\Upsilon_2$
[SRET-1]	$[\nabla_{r,s},\ \Box \longmapsto \nabla_{r,s+1}$	$A_{r,0}[\circ\circ] \to \Upsilon_1 A_{r,s+1}[\circ\circ]\Upsilon_2$
[SRET-2a]	$[\nabla_{r,s},\ \Box \longmapsto \nabla'_{r,s+1}$	$A_{r,0}[\circ\circ] \to \Upsilon_1 A_{r,s+1}[\circ\circ\gamma']\Upsilon_2$
[SRET-2b]	$\langle\gamma', t, u\rangle\ \nabla'_{r,s+1} \longmapsto \nabla_{r,s+1}$	$A_{r,0}[\circ\circ] \to \Upsilon_1 A_{r,s+1}[\circ\circ\gamma']\Upsilon_2$
[SRET-3a]	$[\nabla_{r,s},\ \Box \longmapsto \langle\gamma, r, s+1\rangle$	$A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\circ\circ]\Upsilon_2$
[SRET-3b]	$\langle\gamma, r, s+1\rangle \longmapsto \langle\gamma, r, s+1\rangle\ \nabla_{r,s+1}$	$A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\circ\circ]\Upsilon_2$
[SCAN]	$A_{r,0} \overset{a}{\longmapsto} \Box$	$A_{r,0}[\,] \to a$

Tabla 7.10: Esquema de compilación descendente de LIG en BEPDA

[INIT]	$\$_0 \longmapsto \$_0, [\nabla_{0,0}$	
[CALL]	$\nabla_{r,s} \longmapsto \nabla_{r,s},\ [\overline{A_{r,s+1}}$	$\boldsymbol{A_{r,0}} \to \Upsilon_1 \boldsymbol{A_{r,s+1}} \Upsilon_2$
[SEL]	$\overline{A_{r,0}} \longmapsto \nabla_{r,0}$	$r \neq 0$
[PUB]	$\nabla_{r,n_r} \longmapsto \overline{\overline{A_{r,0}}}$	
[RET]	$\nabla_{r,s},\ [\overline{\overline{A_{r,s+1}}} \longmapsto \nabla_{r,s+1}$	$\boldsymbol{A_{r,0}} \to \Upsilon_1 A_{r,s+1}[\,]\Upsilon_2$
[SRET-1]	$[\nabla_{r,s},\ \overline{\overline{A_{r,s+1}}} \longmapsto \nabla_{r,s+1}$	$A_{r,0}[\circ\circ] \to \Upsilon_1 A_{r,s+1}[\circ\circ]\Upsilon_2$
[SRET-2a]	$[\nabla_{r,s},\ \overline{\overline{A_{r,s+1}}} \longmapsto \nabla'_{r,s+1}$	$A_{r,0}[\circ\circ] \to \Upsilon_1 A_{r,s+1}[\circ\circ\gamma']\Upsilon_2$
[SRET-2b]	$\langle\gamma', t, u\rangle\ \nabla'_{r,s+1} \longmapsto \nabla_{r,s+1}$	$A_{r,0}[\circ\circ] \to \Upsilon_1 A_{r,s+1}[\circ\circ\gamma']\Upsilon_2$
[SRET-3a]	$[\nabla_{r,s},\ \overline{\overline{A_{r,s+1}}} \longmapsto \langle\gamma, r, s+1\rangle$	$A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\circ\circ]\Upsilon_2$
[SRET-3b]	$\langle\gamma, r, s+1\rangle \longmapsto \langle\gamma, r, s+1\rangle\ \nabla_{r,s+1}$	$A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\circ\circ]\Upsilon_2$
[SCAN]	$\overline{A_{r,0}} \overset{a}{\longmapsto} \overline{\overline{A_{r,0}}}$	$A_{r,0}[\,] \to a$

Tabla 7.11: Reglas del esquema de compilación Earley de LIG en BEPDA

[INIT]	$\$_0 \longmapsto \$_0, [\nabla_{0,0}$	
[CALL]	$\nabla_{r,s} \longmapsto \nabla_{r,s},\ [\Box$	$\boldsymbol{A_{r,0}} \rightarrow \Upsilon_1 \boldsymbol{A_{r,s+1}} \Upsilon_2$
[SEL]	$\Box \longmapsto \nabla_{r,0}$	$r \neq 0$
[PUB]	$\nabla_{r,n_r} \longmapsto A_{r,0}$	
[RET]	$\nabla_{r,s},\ [A_{r,s+1} \longmapsto \nabla_{r,s+1}$	$\boldsymbol{A_{r,0}} \rightarrow \Upsilon_1 A_{r,s+1}[\,]\Upsilon_2$
[SRET-1]	$[\nabla_{r,s},\ A_{r,s+1} \longmapsto \nabla_{r,s+1}$	$A_{r,0}[\circ\circ] \rightarrow \Upsilon_1 A_{r,s+1}[\circ\circ]\Upsilon_2$
[SRET-2a]	$[\nabla_{r,s},\ A_{r,s+1} \longmapsto \nabla'_{r,s+1}$	$A_{r,0}[\circ\circ] \rightarrow \Upsilon_1 A_{r,s+1}[\circ\circ\gamma']\Upsilon_2$
[SRET-2b]	$\langle \gamma', t, u\rangle\ \nabla'_{r,s+1} \longmapsto \nabla_{r,s+1}$	$A_{r,0}[\circ\circ] \rightarrow \Upsilon_1 A_{r,s+1}[\circ\circ\gamma']\Upsilon_2$
[SRET-3a]	$[\nabla_{r,s},\ A_{r,s+1} \longmapsto \langle \gamma, r, s+1\rangle$	$A_{r,0}[\circ\circ\gamma] \rightarrow \Upsilon_1 A_{r,s+1}[\circ\circ]\Upsilon_2$
[SRET-3b]	$\langle \gamma, r, s+1\rangle \longmapsto \langle \gamma, r, s+1\rangle\ \nabla_{r,s+1}$	$A_{r,0}[\circ\circ\gamma] \rightarrow \Upsilon_1 A_{r,s+1}[\circ\circ]\Upsilon_2$
[SCAN]	$\Box \overset{a}{\longmapsto} A_{r,0}$	$A_{r,0}[\,] \rightarrow a$

Tabla 7.12: Reglas del esquema de compilación ascendente de LIG en BEPDA

7.8 Lenguajes de adjunción de árboles y BEPDA

Durante todo este capítulo hemos estado suponiendo que la clase de los lenguajes aceptados por los BEPDA coincide con la clase de los lenguajes de adjunción de árboles, principalmente debido a su dualidad con los EPDA. En esta sección proporcionamos una justificación formal a tal suposición. Para ello haremos uso de los esquemas de compilación definidos previamente.

Teorema 7.8 *Los lenguajes adjunción de árboles son un subconjunto de los lenguajes aceptados por la clase de los autómatas a pila embebidos ascendentes.*

Demostración:

Por el esquema de compilación 7.2, a partir de cualquier gramática de adjunción de árboles es posible construir un BEPDA que acepta el lenguaje reconocido por dicha gramática. Análogamente, por el esquema de compilación 7.6, a partir de cualquier gramática lineal de índices es posible construir un BEPDA que acepta el lenguaje reconocido por dicha gramática. □

Teorema 7.9 *La clase de los lenguajes aceptados por los BEPDA es un subconjunto de los lenguajes de adjunción de árboles.*

Demostración:

Mostraremos que para todo BEPDA existe una gramática lineal de índices tal que el lenguaje reconocido por la gramática coincide con el lenguaje aceptado por el autómata.

Sea $\mathcal{A} = (V_T, V_S, \Theta, \$_0, \$_f)$ un autómata a pila embebido ascendente. Construiremos una gramática lineal de índices $\mathcal{L} = (V_T, V_N, V_I, S, P)$, donde $V_I = V_S$ y el conjunto V_N de no-terminales estará formado por pares $\langle E, B\rangle$ tal que $A, B \in V_S$. Para que $\mathcal{L}$ reconozca el lenguaje aceptado por $\mathcal{A}$ el conjunto de producciones en P ha de construirse a partir de las transiciones en Θ de la siguiente manera:

- Para toda transición $C \overset{a}{\longmapsto} F$ y para todo $E \in V_S$ creamos una producción
$$\langle E, F\rangle[\circ\circ] \rightarrow \langle E, C\rangle[\circ\circ]\ a$$
- Para toda transición $C \overset{a}{\longmapsto} CF$ y para todo $E \in V_S$ creamos una producción
$$\langle E, F\rangle[\circ\circ C] \rightarrow \langle E, C\rangle[\circ\circ]\ a$$
- Para toda transición $CF \overset{a}{\longmapsto} G$ y para todo $E \in V_S$ creamos una producción
$$\langle E, G\rangle[\circ\circ] \rightarrow \langle E, F\rangle[\circ\circ C]\ a$$
- Para toda transición $C, [\!F \overset{a}{\longmapsto} G$ y para todo $E \in V_S$ creamos una producción
$$\langle E, G\rangle[\circ\circ] \rightarrow \langle E, C\rangle[\circ\circ]\ \langle C, F\rangle[\,]\ a$$
- Para toda transición $[\!C, F \overset{a}{\longmapsto} G$ y para todo $E \in V_S$ creamos una producción
$$\langle E, G\rangle[\circ\circ] \rightarrow \langle E, C\rangle[\,]\ \langle C, F\rangle[\circ\circ]\ a$$
- Para toda transición $C \overset{a}{\longmapsto} C, [\!F$ creamos una producción
$$\langle C, F\rangle[\,] \rightarrow a$$

Con respecto al axioma de la gramática, tenemos que $S = \langle \$_0, \$_f\rangle$.

Mediante inducción en la longitud de las derivaciones, es posible mostrar que $\langle E, C\rangle[\alpha] \overset{*}{\Rightarrow} w$ si y sólo si $(E, w) \overset{*}{\vdash} (E[\!\alpha C, \epsilon)$. Esto es así puesto que

- Si una derivación $(E, w) \overset{*}{\vdash} (E[\!\alpha C, \epsilon)$ es el resultado de aplicar la secuencia $t_1, \ldots, t_m$ de transiciones en Θ, entonces existe una secuencia $p_1, \ldots, p_m$ de producciones en P tal que p_i es una producción creada a partir de t_i y la derivación derecha $\langle E, C\rangle[\alpha] \overset{*}{\Rightarrow} w$ resultado de aplicar $p_m, \ldots, p_1$ reconoce w.
- Si una derivación derecha $\langle E, C\rangle[\alpha] \overset{*}{\Rightarrow} w$ reconoce la cadena w como resultado de aplicar la secuencia $p_1, \ldots, p_m$ de producciones en P, entonces existe una secuencia de transiciones $t_1, \ldots, t_m$ tal que la p_i es una producción creada a partir de t_i y la derivación $(E, w) \overset{*}{\vdash} (E[\!\alpha C, \epsilon)$ es el resultado de aplicar la secuencia de transiciones $t_m, \ldots, t_1$.

□

Ejemplo 7.4 El autómata a pila embebido ascendente $(V_T, V_S, \Theta, \$_0, \$_f)$, donde $V_T = \{a, b, c, d\}$ y $V_S = \{B, C, D, E, F, \$_0, \$_f\}$, del ejemplo 7.2 acepta el lenguaje $\{a^n b^n c^n d^n \mid n \geq 0\}$. A partir de dicho autómata construiremos una gramática lineal de índices $(V_T, V_S \times V_S, V_S, \langle \$_0, \$_f\rangle, P)$. La tabla 7.13 muestra el conjunto de transiciones P, que ha sido obtenido a partir de las transiciones del autómata mostradas en la tabla 7.2. Para facilitar la lectura, hemos utilizado Γ para denotar cualquier posible elemento de V_S. La tabla 7.14 muestra la derivación de la cadena *aabbccdd* en esta gramática. La primera columna muestra la producción aplicada para obtener la forma sentencial de la segunda columna. Es interesante resaltar que la secuencia de producciones aplicada en la derivación coincide en orden inverso con la secuencia de transiciones aplicada por el autómata para aceptar al misma entrada (tabla 7.2). ¶

Corolario 7.10 *La clase de los lenguajes aceptados por los autómatas a pila embebidos ascendentes coincide con la clase de los lenguajes de adjunción de árboles.*

Demostración:

Inmediata a partir de los teoremas 7.8 y 7.8. □

(a)	$\langle \$_0, D\rangle[\,] \to a$
(b)	$\langle D, D, [\,]\rangle \to a$
(c)	$\langle D, C\rangle[\,] \to \epsilon$
(d)	$\langle C, C\rangle[\,] \to b$
(e)	$\langle \Gamma, B\rangle[\circ\circ] \to \langle \Gamma, C\rangle[\circ\circ]$
(f)	$\langle \Gamma, E\rangle[\circ\circ B] \to \langle \Gamma, B\rangle[\circ\circ]$
(g)	$\langle \Gamma, C\rangle[\circ\circ] \to \langle \Gamma, C\rangle[\,]\ \langle C, E\rangle[\circ\circ]\ c$
(h)	$\langle \Gamma, B\rangle[\circ\circ] \to \langle \Gamma, C\rangle[\circ\circ B]$
(i)	$\langle \Gamma, D\rangle[\circ\circ] \to \langle \Gamma, D\rangle[\,]\ \langle D, B\rangle[\circ\circ]\ d$
(j)	$\langle \Gamma, B\rangle[\circ\circ] \to \langle \Gamma, D\rangle[\circ\circ B]$
(k)	$\langle \Gamma, \$_f\rangle[\circ\circ] \to \langle \Gamma, D\rangle[\circ\circ]$

Tabla 7.13: Producciones de la LIG derivada del BEPDA que acepta $\{a^n b^n c^n d^n\}$

(k)	$\langle \$_0, \$_f\rangle[\,]$	$\Rightarrow \langle \$_0, D\rangle[\,]$
(i)		$\Rightarrow \langle \$_0, D\rangle[\,]\ \ \langle D, B\rangle[\,]\ \ d$
(j)		$\Rightarrow \langle \$_0, D\rangle[\,]\ \ \langle D, D\rangle[\,]\ \ d$
(i)		$\Rightarrow \langle \$_0, D\rangle[\,]\ \ \langle D, D\rangle[\,]\ \ \langle D, B\rangle[B]\ \ dd$
(i)		$\Rightarrow \langle \$_0, D\rangle[\,]\ \ \langle D, D\rangle[\,]\ \ \langle D, B\rangle[B]\ \ dd$
(h)		$\Rightarrow \langle \$_0, D\rangle[\,]\ \ \langle D, D\rangle[\,]\ \ \langle D, C\rangle[BB]\ \ dd$
(g)		$\Rightarrow \langle \$_0, D\rangle[\,]\ \ \langle D, D\rangle[\,]\ \ \langle D, C\rangle[\,]\ \ \langle C, E\rangle[BB]\ \ cdd$
(f)		$\Rightarrow \langle \$_0, D\rangle[\,]\ \ \langle D, D\rangle[\,]\ \ \langle D, C\rangle[\,]\ \ \langle C, B\rangle[B]\ \ cdd$
(f)		$\Rightarrow \langle \$_0, D\rangle[\,]\ \ \langle D, D\rangle[\,]\ \ \langle D, C\rangle[\,]\ \ \langle C, C\rangle[B]\ \ cdd$
(g)		$\Rightarrow \langle \$_0, D\rangle[\,]\ \ \langle D, D\rangle[\,]\ \ \langle D, C\rangle[\,]\ \ \langle C, C\rangle[\,]\ \ \langle C, E\rangle[B]\ \ ccdd$
(f)		$\Rightarrow \langle \$_0, D\rangle[\,]\ \ \langle D, D\rangle[\,]\ \ \langle D, C\rangle[\,]\ \ \langle C, C\rangle[\,]\ \ \langle C, B\rangle[\,]\ \ ccdd$
(e)		$\Rightarrow \langle \$_0, D\rangle[\,]\ \ \langle D, D\rangle[\,]\ \ \langle D, C\rangle[\,]\ \ \langle C, C\rangle[\,]\ \ \langle C, C\rangle[\,]\ \ ccdd$
(d)		$\Rightarrow \langle \$_0, D\rangle[\,]\ \ \langle D, D\rangle[\,]\ \ \langle D, C\rangle[\,]\ \ \langle C, C\rangle[\,]\ \ bccdd$
(d)		$\Rightarrow \langle \$_0, D\rangle[\,]\ \ \langle D, D\rangle[\,]\ \ \langle D, C\rangle[\,]\ \ bbccdd$
(c)		$\Rightarrow \langle \$_0, D\rangle[\,]\ \ \langle D, D\rangle[\,]\ \ bbccdd$
(b)		$\Rightarrow \langle \$_0, D\rangle[\,]\ \ abbccdd$
(a)		$\Rightarrow aabbccdd$

Tabla 7.14: Derivación de la cadena $aabbccdd$

7.9 Tabulación

Al igual que en el caso de los EPDA, la ejecución directa de los autómatas a pila embebidos ascendentes puede tener complejidad exponencial puesto que será preciso duplicar el contenido de la pila del autómata en aquellos casos en que sea posible aplicar más de una transición a una configuración dada. Para obtener una complejidad polinomial, recurriremos a una técnica de tabulación que nos permita representar las configuraciones en forma compacta mediante ítems que a su vez se almacenarán en una tabla. Para ello debemos distinguir los diferentes tipos de derivaciones que se pueden dar, que en el caso de los BEPDA son los dos tipos siguientes:

Derivaciones de llamada. Son derivaciones que sitúan una pila con un único elemento en la cima de la pila principal:

$$(\Upsilon \ [\![B, a_{i+1} \dots a_n) \overset{*}{\vdash} (\Upsilon \ [\![C, a_{j+1} \dots a_n)$$

La pila Υ no debe haber sido tocada en toda la derivación y no debe existir $([\![F, f) \neq ([\![B, i)$ tal que

$$(\Upsilon \ [\![F, a_{f+1} \dots a_n) \overset{*}{\vdash} (\Upsilon \ [\![B, a_{i+1} \dots a_n) \overset{*}{\vdash} (\Upsilon \ [\![C, a_{j+1} \dots a_n)$$

Las derivaciones de este tipo son independientes del valor concreto de Υ, puesto que para cualquier $\Upsilon' \in ([\![V_S^*)^*$ se cumple que

$$(\Upsilon' \ [\![B, a_{i+1} \dots a_n) \overset{*}{\vdash} (\Upsilon' \ [\![C, a_{j+1} \dots a_n)$$

tal y como se observa en la figura 7.7. En consecuencia, las derivaciones de llamada pueden ser representadas de modo compacto por ítems de la forma

$$[B, i, C, j, - \mid -, -, -, -]$$

Derivaciones de retorno. Son aquellas derivaciones que sitúan en la cima de la pila principal una pila con más de un símbolo de pila:

$$\begin{aligned}(\Upsilon \ [\![B, a_{i+1} \dots a_n) \ &\overset{*}{\vdash} (\Upsilon \ [\![B \ \ \Upsilon_1 \ [\![D, a_{p+1} \dots a_n) \\ &\overset{*}{\vdash} (\Upsilon \ [\![B \ \ \Upsilon_1 \ [\![\alpha E, a_{q+1} \dots a_n) \\ &\overset{*}{\vdash} (\Upsilon \ [\![\alpha X C, a_{j+1} \dots a_n)\end{aligned}$$

No deben existir $([\![F, f) \neq ([\![B, i)$ ni $([\![G, g) \neq ([\![D, p)$ tal que

$$\begin{aligned}(\Upsilon \ [\![F, a_{f+1} \dots a_n) \ &\overset{*}{\vdash} (\Upsilon \ [\![B, a_{i+1} \dots a_n) \\ &\overset{*}{\vdash} (\Upsilon \ [\![B \ \ \Upsilon_1 \ [\![G, a_{g+1} \dots a_n) \\ &\overset{*}{\vdash} (\Upsilon \ [\![B \ \ \Upsilon_1 \ [\![D, a_{p+1} \dots a_n) \\ &\overset{*}{\vdash} (\Upsilon \ [\![B \ \ \Upsilon_1 \ [\![\alpha E, a_{q+1} \dots a_n) \\ &\overset{*}{\vdash} (\Upsilon \ [\![\alpha X C, a_{j+1} \dots a_n)\end{aligned}$$

Las derivaciones de este tipo son independientes de Υ y α, puesto que para cualquier $\Upsilon' \in ([\![V_S^*)^*$ y $\alpha' \in V_S^*$ se cumple que

$$\begin{aligned}(\Upsilon' \ [\![B, a_{i+1} \dots a_n) \ &\overset{*}{\vdash} (\Upsilon' \ [\![B \ \ \Upsilon_1 \ [\![D, a_{p+1} \dots a_n) \\ &\overset{*}{\vdash} (\Upsilon' \ [\![B \ \ \Upsilon_1 \ [\![\alpha' E, a_{q+1} \dots a_n) \\ &\overset{*}{\vdash} (\Upsilon' \ [\![\alpha' X C, a_{j+1} \dots a_n)\end{aligned}$$

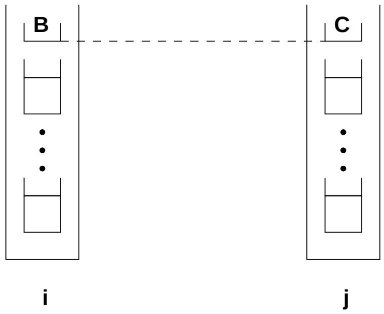

Figura 7.7: Derivaciones de llamada en BEPDA

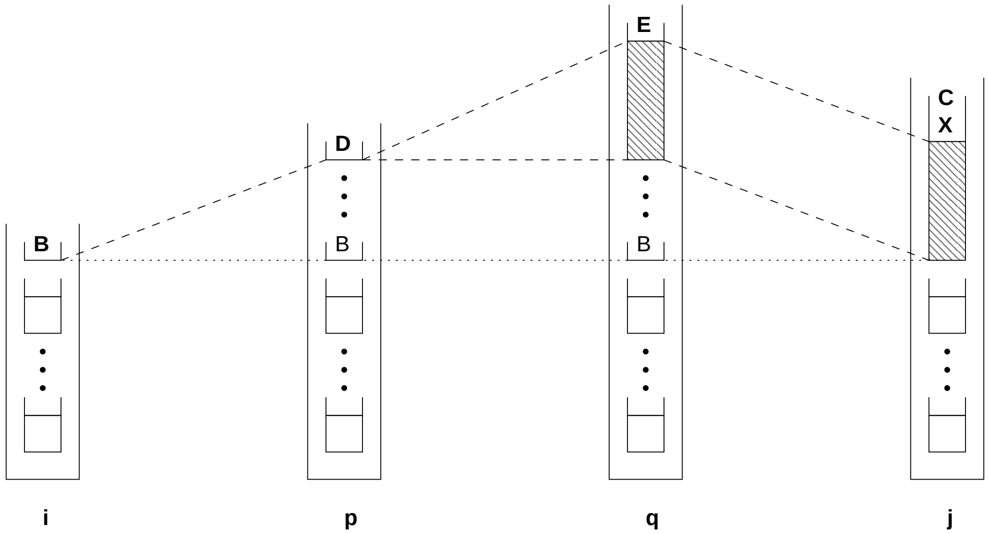

Figura 7.8: Derivaciones de retorno en BEPDA

tal y como se observa en la figura 7.8. En consecuencia, las derivaciones de retorno se pueden representar de forma compacta mediante ítems de la forma

$$[B, i, C, j, X \mid D, p, E, q]$$

Los ítems se combinan mediante las reglas de combinación mostradas en la tabla 7.15, en las cuales se cumple que

- $k = j$ si $a = \epsilon$ y $k = j + 1$ si $a = a_{j+1}$.
- $l = k'$ si $b = \epsilon$ y $l = k' + 1$ si $b = a_{k'+1}$.

El ítem inicial a partir del cual se aplican las reglas de la tabla 7.15 es de la forma

$$[\$_0, 0, \$_0, 0, - \mid -, -, -, -]$$

Si la cadena de entrada $a_1 \ldots a_n$ pertenece al lenguaje aceptado por el autómata, se producirán ítems finales de la forma

$$[B, 0, \$_f, n, - \mid -, -, -, -]$$

tal que existe una transición $\$_0 \longmapsto \$_0, [B$.

$$\frac{[B,i,C,j,X \mid D,p,E,q]}{[B,i,F,k,X \mid D,p,E,q]} \quad C \overset{a}{\longmapsto} F$$

$$\frac{[B,i,C,j,X \mid D,p,E,q]}{[B,i,F,k,C \mid B,i,C,j]} \quad C \overset{a}{\longmapsto} CF$$

$$\frac{\begin{array}{c}[B,i,F,j,C \mid D,p,E,q] \\ [D,p,E,q,X' \mid O,u,P,v]\end{array}}{[B,i,G,k,X' \mid O,u,P,v]} \quad CF \overset{a}{\longmapsto} G$$

$$\frac{[B,i,C,j,X \mid D,p,E,q]}{[F,k,F,k,- \mid -,-,-,-]} \quad C \overset{a}{\longmapsto} C, [F$$

$$\frac{\begin{array}{c}[F,k,F',k',- \mid -,-,-,-] \\ [B,i,C,j,X \mid D,p,E,q]\end{array}}{[B,i,G,l,X \mid D,p,E,q]} \quad \begin{array}{l} C \overset{a}{\longmapsto} C, [F \\ C, [F' \overset{b}{\longmapsto} G \end{array}$$

$$\frac{\begin{array}{c}[F,k,F',k',X \mid D,p,E,q] \\ [B,i,C,j,- \mid -,-,-,-]\end{array}}{[B,i,G,l,X \mid D,p,E,q]} \quad \begin{array}{l} C \overset{a}{\longmapsto} C, [F \\ [C, F' \overset{b}{\longmapsto} G \end{array}$$

Tabla 7.15: Combinación de ítems en BEPDA

Teorema 7.11 *La manipulación de configuraciones mediante la aplicación de transiciones en los autómatas a pila embebidos ascendentes es equivalente a la manipulación de ítems mediante las reglas de combinación de la tabla 7.15.*

Demostración:

Debemos demostrar que para toda derivación del autómata se producirá un ítem que la representa, y que para todo ítem producido por las reglas de combinación de la tabla 7.15 existe una configuración en el autómata a la que dicho ítem representa. Para ello daremos una lista exhaustiva de los diferentes tipos de derivaciones que se pueden producir, junto con la reglas de combinación de ítem que produce al ítem que representa dicha configuración. Veremos que toda reglas de combinación de ítem aparece en dicha lista, por lo que habremos demostrado los puntos. Las posibles derivaciones que se pueden producir son:

- Derivaciones que son el resultado de aplicar una transición $C \overset{a}{\longmapsto} F$
  - a una derivación de llamada:

$$\begin{array}{ll} (\Upsilon\ [B, a_{i+1} \dots a_n) & \overset{*}{\vdash} (\Upsilon\ [C, a_{j+1} \dots a_n) \\ & \vdash (\Upsilon\ [F, a_{k+1} \dots a_n) \end{array}$$

$$\frac{[B,i,C,j,- \mid -,-,-,-]}{[B,i,F,k,- \mid -,-,-,-]} \quad C \overset{a}{\longmapsto} F$$

– a una derivación de retorno:

$$\begin{array}{ll} (\Upsilon\ [B, a_{i+1}\dots a_n) & \overset{*}{\vdash} (\Upsilon\ [B\ \ \Upsilon_1\ [D, a_{p+1}\dots a_n) \\ & \overset{*}{\vdash} (\Upsilon\ [B\ \ \Upsilon_1\ [\alpha E, a_{q+1}\dots a_n) \\ & \overset{*}{\vdash} (\Upsilon\ [\alpha XC, a_{j+1}\dots a_n) \\ & \vdash (\Upsilon\ [\alpha XF, a_{k+1}\dots a_n) \end{array}$$

$$\frac{[B,i,C,j,X \mid D,p,E,q]}{[B,i,F,k,X \mid D,p,E,q]}\quad C \overset{a}{\longmapsto} F$$

- Derivaciones que son el resultado de aplicar una transición $C \overset{a}{\longmapsto} CF$
  – a una derivación de llamada:

$$\begin{array}{ll} (\Upsilon\ [B, a_{i+1}\dots a_n) & \overset{*}{\vdash} (\Upsilon\ [C, a_{j+1}\dots a_n) \\ & \vdash (\Upsilon\ [CF, a_{k+1}\dots a_n) \end{array}$$

$$\frac{[B,i,C,j,- \mid -,-,-,-]}{[B,i,F,k,C \mid B,i,C,j]}\quad C \overset{a}{\longmapsto} CF$$

  – a una derivación de retorno:

$$\begin{array}{ll} (\Upsilon\ [B, a_{i+1}\dots a_n) & \overset{*}{\vdash} (\Upsilon\ [B\ \ \Upsilon_1\ [D, a_{p+1}\dots a_n) \\ & \overset{*}{\vdash} (\Upsilon\ [B\ \ \Upsilon_1\ [\alpha E, a_{q+1}\dots a_n) \\ & \overset{*}{\vdash} (\Upsilon\ [\alpha XC, a_{j+1}\dots a_n) \\ & \vdash (\Upsilon\ [\alpha XCF, a_{k+1}\dots a_n) \end{array}$$

$$\frac{[B,i,C,j,X \mid D,p,E,q]}{[B,i,F,k,C \mid B,i,C,j]}\quad C \overset{a}{\longmapsto} CF$$

- Derivaciones que son el resultado de aplicar una transición $CF \overset{a}{\longmapsto} G$ a una derivación de retorno[1] que a su vez a sido obtenida:
  – a partir de una derivación de llamada:

$$\begin{array}{ll} (\Upsilon\ [B, a_{i+1}\dots a_n) & \overset{*}{\vdash} (\Upsilon\ [B\ \Upsilon_1\ [D, a_{p+1}\dots a_n) \\ & \overset{*}{\vdash} (\Upsilon\ [B\ \Upsilon_1\ [E, a_{p+1}\dots a_n) \\ & \overset{*}{\vdash} (\Upsilon\ [CF, a_{j+1}\dots a_n) \\ & \vdash (\Upsilon\ [G, a_{k+1}\dots a_n) \end{array}$$

$$\frac{\begin{array}{c}[B,i,F,j,C \mid D,p,E,q] \\ [D,p,E,q,- \mid -,-,-,-]\end{array}}{[B,i,G,k,- \mid -,-,-,-]}\quad CF \overset{a}{\longmapsto} G$$

  – a partir de una derivación de retorno:

$$\begin{array}{ll} (\Upsilon\ [B, a_{i+1}\dots a_n) & \overset{*}{\vdash} (\Upsilon\ [B\ \ \Upsilon_1\ [D, a_{p+1}\dots a_n) \\ & \overset{*}{\vdash} (\Upsilon\ [B\ \ \Upsilon_1\ [D\ \Upsilon_2\ \ [O, a_{u+1}\dots a_n) \\ & \overset{*}{\vdash} (\Upsilon\ [B\ \ \Upsilon_1\ [D\ \Upsilon_2\ \ [\alpha P, a_{v+1}\dots a_n) \\ & \overset{*}{\vdash} (\Upsilon\ [B\ \ \Upsilon_1\ [\alpha X'E, a_{q+1}\dots a_n) \\ & \overset{*}{\vdash} (\Upsilon\ [\alpha X'CF, a_{j+1}\dots a_n) \\ & \vdash (\Upsilon\ [\alpha X'G, a_{k+1}\dots a_n) \end{array}$$

[1]Este tipo de transiciones no son aplicables a derivaciones de llamada, pues estas últimas contienen en su cima una pila de la forma $[F$.

$$\frac{\begin{array}{c}[B,i,F,j,C \mid D,p,E,q] \\ [D,p,E,q,X' \mid O,u,P,v]\end{array}}{[B,i,G,k,X' \mid O,u,P,v]} \quad CF \overset{a}{\longmapsto} G$$

- Derivaciones que son el resultado de aplicar una transición $C \overset{a}{\longmapsto} C, \llbracket F$
  - a una derivación de llamada:

$$\begin{array}{ll}(\Upsilon \llbracket B, a_{i+1} \ldots a_n) & \overset{*}{\vdash} (\Upsilon \llbracket C, a_{j+1} \ldots a_n) \\ & \vdash (\Upsilon \llbracket C \; \llbracket F, a_{k+1} \ldots a_n)\end{array}$$

$$\frac{[B,i,C,j,- \mid -,-,-,-]}{[F,k,F,k,- \mid -,-,-,-]} \quad C \overset{a}{\longmapsto} C, \llbracket F$$

  - a una derivación de retorno:

$$\begin{array}{ll}(\Upsilon \llbracket B, a_{i+1} \ldots a_n) & \overset{*}{\vdash} (\Upsilon \llbracket B \;\; \Upsilon_1 \llbracket D, a_{p+1} \ldots a_n) \\ & \overset{*}{\vdash} (\Upsilon \llbracket B \;\; \Upsilon_1 \llbracket \alpha E, a_{q+1} \ldots a_n) \\ & \overset{*}{\vdash} (\Upsilon \llbracket \alpha X C, a_{j+1} \ldots a_n) \\ & \vdash (\Upsilon \llbracket \alpha X C \;\; \llbracket F, a_{k+1} \ldots a_n)\end{array}$$

$$\frac{[B,i,C,j,X \mid D,p,E,q]}{[F,k,F,k,- \mid -,-,-,-]} \quad C \overset{a}{\longmapsto} C, \llbracket F$$

- Derivaciones que son el resultado de aplicar una transición $C, \llbracket F' \overset{b}{\longmapsto} G$ a una derivación obtenida a partir de la aplicación de una transición $C \overset{a}{\longmapsto} C, \llbracket F$:
  - a una derivación de llamada:

$$\begin{array}{ll}(\Upsilon \llbracket B, a_{i+1} \ldots a_n) & \overset{*}{\vdash} (\Upsilon \llbracket C, a_{j+1} \ldots a_n) \\ & \vdash (\Upsilon \llbracket C \;\; \llbracket F, a_{k+1} \ldots a_n) \\ & \overset{*}{\vdash} (\Upsilon \llbracket C \;\; \llbracket F', a_{k'+1} \ldots a_n) \\ & \vdash (\Upsilon \llbracket G, a_{l+1} \ldots a_n)\end{array}$$

$$\frac{\begin{array}{c}[F,k,F',k',- \mid -,-,-,-] \\ [B,i,C,j,- \mid -,-,-,-]\end{array}}{[B,i,G,l,- \mid -,-,-,-]} \quad \begin{array}{l} C \overset{a}{\longmapsto} C, \llbracket F \\ C, \llbracket F' \overset{b}{\longmapsto} G\end{array}$$

  - a una derivación de retorno:

$$\begin{array}{ll}(\Upsilon \llbracket B, a_{i+1} \ldots a_n) & \overset{*}{\vdash} (\Upsilon \llbracket B \;\; \Upsilon_1 \llbracket D, a_{p+1} \ldots a_n) \\ & \overset{*}{\vdash} (\Upsilon \llbracket B \;\; \Upsilon_1 \llbracket \alpha E, a_{q+1} \ldots a_n) \\ & \overset{*}{\vdash} (\Upsilon \llbracket \alpha X C, a_{j+1} \ldots a_n) \\ & \vdash (\Upsilon \llbracket \alpha X C \;\; \llbracket F, a_{k+1} \ldots a_n) \\ & \overset{*}{\vdash} (\Upsilon \llbracket \alpha X C \;\; \llbracket F', a_{k'+1} \ldots a_n) \\ & \vdash (\Upsilon \llbracket \alpha X G, a_{l+1} \ldots a_n)\end{array}$$

$$\frac{\begin{array}{c}[F,k,F',k',- \mid -,-,-,-] \\ [B,i,C,j,X \mid D,p,E,q]\end{array}}{[B,i,G,l,X \mid D,p,E,q]} \quad \begin{array}{l} C \overset{a}{\longmapsto} C, \llbracket F \\ C, \llbracket F' \overset{b}{\longmapsto} G\end{array}$$

- Derivaciones que son el resultado de aplicar una transición $C, [F' \overset{b}{\longmapsto} G$ a una derivación obtenida a partir de la aplicación de una transición $C \overset{a}{\longmapsto} C, [F$ a una derivación de llamada[2], con los dos casos siguientes:
  - la derivación obtenida es una derivación de llamada:

$$\begin{array}{ll}(\Upsilon\ [B, a_{i+1} \ldots a_n) & \overset{*}{\vdash} (\Upsilon\ [C, a_{j+1} \ldots a_n) \\ & \vdash (\Upsilon\ [C\ \ [F, a_{k+1} \ldots a_n) \\ & \overset{*}{\vdash} (\Upsilon\ [C\ \ [F', a_{k'+1} \ldots a_n) \\ & \vdash (\Upsilon\ [G, a_{l+1} \ldots a_n)\end{array}$$

$$\begin{array}{ll}[F, k, F', k', - \mid -, -, -, -] & \\ [B, i, C, j, - \mid -, -, -, -] & C \overset{a}{\longmapsto} C, [F \\ \hline [B, i, G, l, - \mid -, -, -, -] & [C, F' \overset{b}{\longmapsto} G\end{array}$$

  - la derivación obtenida es una derivación de retorno:

$$\begin{array}{ll}(\Upsilon\ [B, a_{i+1} \ldots a_n) & \overset{*}{\vdash} (\Upsilon\ [C, a_{j+1} \ldots a_n) \\ & \vdash (\Upsilon\ [C\ \ [F, a_{k+1} \ldots a_n) \\ & \overset{*}{\vdash} (\Upsilon\ [C\ \ [F\ \ \Upsilon_1\ [D, a_{p+1} \ldots a_n) \\ & \overset{*}{\vdash} (\Upsilon\ [C\ \ [F\ \ \Upsilon_1\ [\alpha E, a_{q+1} \ldots a_n) \\ & \overset{*}{\vdash} (\Upsilon\ [C\ \ [\alpha X F', a_{k'+1} \ldots a_n) \\ & \vdash (\Upsilon\ [\alpha X G, a_{l+1} \ldots a_n)\end{array}$$

$$\begin{array}{ll}[F, k, F', k', X \mid D, p, E, q] & \\ [B, i, C, j, - \mid -, -, -, -] & C \overset{a}{\longmapsto} C, [F \\ \hline [B, i, G, l, X \mid D, p, E, q] & [C, F' \overset{b}{\longmapsto} G\end{array}$$

□

La complejidad espacial de la técnica de tabulación propuesta con respecto a la longitud n de la cadena de entrada es $\mathcal{O}(n^4)$, puesto que cada ítem almacena 4 posiciones de la cadena de entrada. La complejidad temporal es $\mathcal{O}(n^6)$ y es debida a las reglas de combinación correspondientes a las transición $CF \overset{a}{\longmapsto} G$, a la transición $C, [F' \overset{b}{\longmapsto} G$ y a la transición $[C, F' \overset{b}{\longmapsto} G$, pues no debemos olvidar que la posición k está ligada a j y que la posición l está ligada a k'.

[2] Si se hubiese aplicado esta transición a una derivación de retorno, no sería posible tener una pila $[C$ bajo la pila [F'.

Capítulo 8

Autómatas lógicos a pila restringidos

En este capítulo se describe un nuevo tipo de autómata para los lenguajes de adjunción de árboles. Para ello hemos utilizado la estrecha relación existente entre las gramáticas lineales de índices y las gramáticas de cláusulas definidas, que nos ha permitido definir una versión restringida de los autómatas lógicos a pila adecuada para el reconocimiento de LIG y TAG, así como las técnicas de tabulación asociadas a distintas estrategias de análisis. Este capítulo está basado en [14].

8.1 Introducción

Los autómatas lógicos a pila (*Logic Push-Down Automata*, LPDA) [105, 56] son esencialmente una extensión de los autómatas a pila que en lugar de almacenar símbolos elementales en la pila almacenan elementos de la lógica de predicados de primer orden. Ello permite que sean utilizados para describir estrategias de prueba para programas de cláusulas definidas y en particular estrategias de análisis sintáctico para Gramáticas de Cláusulas Definidas (*Definite Clause Grammars*, DCG) [143]. Una característica muy importante de los autómatas lógicos a pila es que, independientemente de la estrategia de análisis que incorporen, pueden ser ejecutados eficientemente mediante una técnica de tabulación estándar [56, 52].

Podemos contemplar las gramáticas lineales de índices como un caso especial de las gramáticas de cláusulas definidas. En efecto, ambas son extensiones de las gramáticas independientes del contexto en las que los símbolos no-terminales han sido enriquecidos con información adicional, una pila de índices en el caso de las gramáticas lineales de índices y un conjunto de términos en el caso de las gramáticas de cláusulas definidas. En particular, una pila de índices se pueden representar como una lista en una gramática de cláusulas definidas, por lo que es posible traducir los elementos que componen una gramática lineal de índices en elementos de una gramática de cláusulas definidas. Para ello es preciso convertir los no-terminales en predicados unarios y las pilas de índices en listas, que constituirán su único argumento, así como utilizar variables para representar las partes dependientes de las pilas. En la tabla 8.1 se muestran las reglas de esta traducción de símbolos LIG en símbolos DCG. Mediante la extensión de este esquema de traducción a las producciones de la gramática obtendremos un conjunto de cláusulas definidas, tal como se muestra en la tabla 8.2, en donde $T(\Upsilon)$ denota la traducción de una secuencia de símbolos LIG en los correspondientes símbolos de una gramática de cláusulas definidas.

En este capítulo utilizamos los autómatas lógicos a pila para el análisis de lenguajes de adjunción de árboles. Para ello restringiremos los elementos que se pueden almacenar en la pila y especializaremos las técnica de tabulación para equiparar su complejidad espacial y temporal a la de los algoritmos tabulares de análisis sintáctico de gramáticas lineales de índices [15, 19] y

Símbolo LIG	Símbolo DCG
$A[\,]$	`big_a([])`
$A[\circ\circ]$	`big_a(X)`
$A[\circ\circ\gamma]$	`big_a([gamma|X])`

Tabla 8.1: Traducción de símbolos LIG en símbolos DCG

Producción LIG	Cláusula definida
$A[\circ\circ] \to \Upsilon_1\, B[\circ\circ]\, \Upsilon_2$	`big_a(X) -->` $T(\Upsilon_1)$, `big_b(X),` $T(\Upsilon_2)$.
$A[\circ\circ\gamma] \to \Upsilon_1\, B[\circ\circ]\, \Upsilon_2$	`big_a([gamma|X]) -->` $T(\Upsilon_1)$, `big_b(X),` $T(\Upsilon_2)$.
$A[\circ\circ] \to \Upsilon_1\, B[\circ\circ\gamma]\, \Upsilon_2$	`big_a(X) -->` $T(\Upsilon_1)$, `big_b([gamma|X]),` $T(\Upsilon_2)$.
$A[\,] \to w$	`big_a([]) -->` w.

Tabla 8.2: Traducción de producciones LIG en cláusulas definidas

de gramáticas de adjunción de árboles [8, 9], creando de este modo los *autómatas lógicos a pila restringidos* (*Restricted Logic Push-Down Automata*, RLPDA).

8.2 Autómatas lógicos a pila

Un autómata a pila es un mecanismo operacional consistente en una memoria de estado finito y una pila que conjuntamente determinan el estado del autómata, más un conjunto finito de transiciones que definen los posibles cambios de estado en computaciones que parten de un estado inicial estándar. Una característica muy importante es que cada transición sólo puede consultar o cambiar un número acotado de elementos de la pila, contados a partir de su cima. Sin pérdida de generalidad, podemos considerar autómatas a pila sin memoria de estado finito, pues el valor que esta posee en un momento determinado puede ser almacenado en el elemento ubicado en la cima de la pila [24].

Los autómatas lógicos a pila [105, 56] son una generalización de los autómatas a pila que almacenan en la pila átomos definidos de acuerdo con un conjunto de predicados, funciones y variables en lugar de simples símbolos gramaticales. En el caso del análisis de gramáticas de cláusulas definidas, los elementos de la pila sirven habitualmente para representar el reconocimiento de un elemento gramatical que expande una parte determinada de la cadena de entrada.

Formalmente, un autómata lógico a pila es una tupla $(P, \mathcal{F}, \mathcal{X}, \$_0, \$_f, \Theta)$ en la que:

- P es un conjunto finito de símbolos de predicado.
- $\mathcal{F}$ es un conjunto finito de símbolos de función.
- $\mathcal{X}$ es un conjunto finito de variables.
- $\$_0$ es el elemento inicial de la pila.
- $\$_f$ es el elemento final de la pila.

- Θ es un conjunto finito de transiciones, cada una de las cuales puede ser de uno de los tres tipos que se muestran a continuación, teniendo en cuenta que $a \in V_T \cup \{\epsilon\}$, las pilas ξ crecen hacia la derecha, A, C, E, F y G representan elementos de la pila y $\text{mgu}(x, y)$ es el unificador más general entre x e y.

  SWAP: Transiciones de la forma $(C \overset{a}{\longmapsto} F)(\xi A) = (\xi F)\sigma$, donde $\sigma = \text{mgu}(A, C)$ y a es el siguiente terminal en la cadena de entrada o ϵ.

  PUSH: Transiciones de la forma $(C \overset{a}{\longmapsto} CF)(\xi A) = (\xi AF)\sigma$, donde $\sigma = \text{mgu}(A, F)$ y a es el siguiente terminal en la cadena de entrada o ϵ.

  POP: Transiciones de la forma $(CF \overset{a}{\longmapsto} G)(\xi EA) = (\xi G)\sigma$, donde $\sigma = \text{mgu}(\langle E, A\rangle, \langle C, F\rangle)$ y a es el siguiente terminal en la cadena de entrada o ϵ.

Definimos los *autómatas lógicos a pila restringidos* o RLPDA como la clase de los autómatas lógicos a pila que almacenan en la pila átomos de la forma $A[\alpha]$, donde A es un predicado unario y α es una lista de funciones 0-arias constantes denominadas índices. En consecuencia, un RLPDA queda definido por un tupla $(V_T, P, V_I, \mathcal{X}, \$_0, \$_f, \Theta)$, donde V_I es un conjunto finito de índices y V_T es un conjunto finito de símbolos terminales[1].

Una *configuración* de un autómata lógico a pila viene dada por un par (ξ, w), donde ξ representa el contenido de la pila y w la parte de la cadena de entrada que resta por analizar. Una configuración (ξ, aw) deriva una configuración (ξ', w), hecho denotado mediante $(\xi, aw) \vdash (\xi', w)$ si y sólo si existe una transición que aplicada a ξ devuelve ξ' y consume a de la cadena de entrada. En caso de ser necesario identificar una derivación d concreta, utilizaremos la notación $\vdash_d$. Denotamos por $\overset{*}{\vdash}$ el cierre reflexivo y transitivo de $\vdash$. Decimos que una cadena de entrada w es aceptada por un autómata lógico a pila si $(\$_0, w) \overset{*}{\vdash} (\$_0\$_f, \epsilon)$.

Una propiedad muy interesante de los autómatas lógicos a pila es que pueden ser interpretados en programación dinámica [56]. Ello se debe a que si tenemos la derivación

$$(B, a_{i+1} \dots a_j \dots a_n) \overset{*}{\vdash} (BC, a_{j+1} \dots a_n)$$

donde B y C son átomos lógicos almacenados en la pila, entonces se cumple

$$(\xi B, a_{i+1} \dots a_j \dots a_n) \overset{*}{\vdash} (\xi BC, a_{j+1} \dots a_n)$$

para toda pila ξ. Denominaremos *derivaciones independientes del contexto* a este tipo de derivaciones. Para representar una derivación independiente del contexto sólo precisamos almacenar los símbolos de pila B y C más las posiciones de la cadena de entrada i y j en las que dichos elementos fueron situados en la cima de la pila, puesto que la derivación es independiente de ξ. Utilizaremos la notación $[B, i, C, j]$ para representar cada uno de dichos ítems.

Puesto que B y C constituyen los dos elementos en la cima de la pila y puesto que todo elemento apilado es consecuencia de la aplicación de una secuencia de transiciones (una derivación), un ítem de la forma $[B, i, C, j]$ representa la existencia de una derivación

$$(\xi\ B, a_{i+1} \dots a_n) \overset{*}{\vdash}_d (\xi\ BC, a_{j+1} \dots a_n)$$

sin que a lo largo de la misma se haya modificado ξB. Abusando de la notación, podemos ver que el resultado de dicha derivación es análogo al de una hipotética transición PUSH

$$B \overset{a_{i+1} \dots a_j}{\longmapsto} BC$$

puesto que sólo depende de B y no se altera ni consulta ξ.

[1] Aunque el conjunto de símbolos terminales forma parte de P en la definición de LPDA, hemos considerado conveniente darle entidad propia en la definición de los RLPDA puesto que estos últimos están orientados a una aplicación específica, como es el análisis sintáctico, en la que los símbolos terminales tienen especial relevancia.

Regla	Tarea
[INIT]	inicia los cálculos a partir de la pila inicial.
[CALL]	requiere el análisis de un determinado elemento gramatical que no es un hijo dependiente.
[SCALL]	(de *spine call*) requiere el análisis de un determinado elemento gramatical que es un hijo dependiente.
[SEL]	selecciona una producción.
[PUB]	determina que un elemento gramatical ya ha sido analizado.
[RET]	continua el proceso de análisis después del reconocimiento de un elemento gramatical que no es un hijo dependiente.
[SRET]	(de *spine ret*) continua el proceso de análisis después del reconocimiento de un elemento gramatical que es un hijo dependiente.
[SCAN]	reconoce los terminales que componen la cadena de entrada.

Tabla 8.3: Reglas para los esquemas de compilación de gramáticas lineales de índices

8.3 Esquemas de compilación de gramáticas lineales de índices

Un *esquema de compilación* es un conjunto de reglas que permite, a partir de una gramática lineal de índices y de una estrategia de análisis sintáctico, construir un autómata lógico a pila restringido que describa los cálculos que se pueden realizar con dicha gramática utilizando la estrategia de análisis elegida. Los esquemas de compilación que se van a mostrar a continuación se basan en el paradigma de llamada/retorno [55], utilizando para ello los 8 tipos de reglas mostrados en la tabla 8.3. A toda regla **[CALL]** le corresponde una regla **[RET]** y viceversa. Lo mismo se aplica a las reglas de tipo **[SCALL]** y **[SRET]**. Las reglas **[INIT]**, **[CALL]**,**[SCALL]** y **[SEL]** definen las transiciones del autómata encargadas de la fase predictiva del algoritmo de análisis mientras que las reglas **[RET]**, **[SRET]** y **[PUB]** definen las transiciones encargadas de propagar la información en la fase ascendente. Por este motivo la fase descendente o predictiva de una estrategia de análisis cuando es implantada en un autómata a pila recibe el nombre de *fase de llamada* mientras que la fase ascendente recibe el nombre de *fase de retorno*.

Con el fin de simplificar el tratamiento de las producciones de la gramática supondremos que todas ellas son de la forma

$$A_{r,0}[\circ\circ\gamma] \rightarrow A_{r,1}[\,] \ldots A_{r,l}[\circ\circ\gamma'] \ldots A_{r,n_r}[\,]$$

donde sólo uno de $\gamma, \gamma' \in V_I$ puede ser distinto de ϵ, y los terminales sólo aparecen en producciones unitarias de la forma $A_{r,0}[\,] \rightarrow a$.

A continuación se mostrarán una serie de esquemas de compilación que incorporan diferentes estrategias de análisis de LIG en autómatas lógicos a pila restringidos. Las diferentes estrategias difieren en la cantidad de información predicha en la fase de llamada y en la cantidad de información propagada en la fase de retorno, teniendo en cuenta que con respecto a una gramática lineal de índices para cada símbolo gramatical se puede predecir el no-terminal y/o la pila de índices y propagar el no-terminal y/o la pila de índices. Teniendo en cuenta estos factores nos referiremos a las diferentes estrategias utilizando el esquema

$$\langle \textit{estrategia-CF} \rangle - \langle \textit{estrategia-índices} \rangle$$

donde *estrategia-CF* se refiere al tipo de estrategia utilizada con respecto al no terminal y *estrategia-índices* al tipo de estrategia aplicado con respecto a la pila de índices asociada al no-terminal. Tanto *estrategia-CF* como *estrategia-índices* pueden tomar alguno de los valores siguientes:

descendente para indicar que se predice pero no se propaga el elemento involucrado (no-terminal o pila de índices).

ascendente para indicar que no se predice pero sí se propaga el elemento involucrado.

Earley para indicar que se predice y se propaga el elemento involucrado.

Para referirnos a todas las estrategias con un determinado comportamiento con respecto al no-terminal independientemente del comportamiento con respecto a la pila de índices utilizaremos la notación $\langle \textit{estrategia-CF} \rangle - *$ mientras que para referirnos al conjunto de estrategias que presentan un determinado comportamiento con respecto a la pila de índices independientemente del comportamiento con respecto al no-terminal utilizaremos la notación $* - \langle \textit{estrategia-índices} \rangle$.

8.3.1 Estrategia genérica

En primer lugar definiremos una estrategia genérica basada en el paradigma llamada/retorno, parametrizada con respecto a la información que se predice y propaga en las fases de llamada y de retorno, respectivamente. Utilizaremos la siguiente notación:

- $\overrightarrow{A_{r,s}}$ para referirnos a la predicción de información con respecto al no-terminal $A_{r,s}$.
- $\overrightarrow{\gamma}$ para indicar la predicción de información del índice γ.
- $\overrightarrow{X}$ y $\overrightarrow{\circ\circ}$ es una extensión de la notación $\overrightarrow{\gamma}$ para representar la información predicha de una pila de índices X o $\circ\circ$.
- $\overleftarrow{A_{r,s}}$ para representar la información propagada con respecto al no-terminal $A_{r,s}$.
- $\overleftarrow{\gamma}$ para referirnos a la información propagada correspondiente al índice γ en la fase ascendente.
- $\overleftarrow{X}$ y $\overleftarrow{\circ\circ}$ es una extensión de la notación $\overleftarrow{\gamma}$ para representar la información propagada de una pila de índices X o $\circ\circ$.

Esquema de compilación 8.1 El esquema de compilación genérico de una gramática lineal de índices en un autómata lógico a pila queda definido por el conjunto de reglas de compilación mostrado en la tabla 8.4 y por los elementos inicial $\$_0()$ y final $\overleftarrow{S}([\,])$. La primera columna de la tabla 8.4 indica el nombre de la regla, la segunda las transiciones generadas por la misma y la tercera las condiciones, generalmente referidas a la forma de las producciones, que se deben cumplir para que la regla sea aplicable. §

Con el fin de adaptar el esquema anterior a una notación más cercana a LIG suprimiremos los paréntesis que encierran los argumentos de los átomos (pues son redundantes al tener sólo uno) , reemplazaremos X por $\circ\circ$ para representar todos los valores almacenados en una pila y denotaremos $[\gamma \mid X]$ mediante $[\circ\circ\gamma]$. Obtenemos así el siguiente esquema de compilación.

Esquema de compilación 8.2 El esquema de compilación genérico utilizando notación LIG de una gramática lineal de índices en un autómata lógico a pila restringido queda definido por el conjunto de reglas mostrado en la tabla 8.5 y por los elementos inicial $\$_0[\,]$ y final $\overleftarrow{S}[\,]$. §

En la tabla 8.6 se muestran los valores que deben tomar los parámetros de predicción y propagación de información para instanciar el esquema de compilación genérico en esquemas correspondientes a diferentes estrategias de análisis. En el caso de estrategias Earley-$*$ es necesario distinguir la llamada de un no-terminal $A_{r,s+1}$ de su retorno, para lo cual utilizamos los símbolos $\overline{A_{r,s+1}}$ y $\overline{\overline{A_{r,s+1}}}$. En el caso de los índices esta diferenciación no es necesaria puesto que podemos determinar si el proceso de análisis se encuentra en la fase descendente o ascendente comprobando el valor del no-terminal, según lo definido para cada estrategia.

Para garantizar la corrección del autómata construido mediante la aplicación de una instancia del esquema de compilación genérico es suficiente con verificar que considerando conjuntamente la información proporcionada por la predicción y la proporcionada por la propagación podemos reconstruir el símbolo de que se trata [55]. Esto es equivalente a verificar que se cumple

$$\mathrm{mgu}(\overrightarrow{A_{r,s}[\alpha]}\ \overleftarrow{A_{r,s}[\alpha]}, \overrightarrow{A_{l,0}[\beta]}\ \overleftarrow{A_{l,0}[\beta]}) = \mathrm{mgu}(A_{r,s}[\alpha], A_{l,0}[\beta])$$

para cualquier producción l.

A continuación describiremos en detalle los esquemas de compilación para las diferentes estrategias de análisis.

8.3.2 Estrategias $*$-ascendentes

En las estrategias de análisis sintáctico $*$-ascendentes, la fase de llamada no predice ninguna información acerca de las pilas de índices, mientras que la fase de retorno propaga toda la información disponible de cada una de dichas pilas. A continuación mostramos diferentes esquemas de compilación para estas estrategias. En estos esquemas se ha aplicado una binarización implícita de las producciones de la gramática de tal modo que una producción

$$A_{r,0}[\circ\circ\gamma] \to A_{r,1}[\,] \dots A_{r,l}[\circ\circ\gamma'] \dots A_{r,n_r}[\,]$$

se ha descompuesto en las siguientes $n_r + 1$ producciones

$$\begin{aligned}
&A_{r,0}[\circ\circ\gamma] \to \nabla_{r,n_r}[\circ\circ\gamma] \\
&\nabla_{r,n_r}[\circ\circ\gamma] \to \nabla_{r,n_r-1}[\circ\circ\gamma]\ A_{r,n_r}[\,] \\
&\vdots \\
&\nabla_{r,l}[\circ\circ\gamma] \to \nabla_{r,l-1}[\,]\ A_{r,l}[\circ\circ\gamma'] \\
&\nabla_{r,l-1}[\circ\circ] \to \nabla_{r,l-2}[\circ\circ]\ A_{r,l-1}[\,] \\
&\vdots \\
&\nabla_{r,1}[\circ\circ] \to \nabla_{r,0}[\circ\circ]\ A_{r,1}[\,] \\
&\nabla_{r,0}[\,] \to \epsilon
\end{aligned}$$

donde la pila de índices asociada a los $\nabla_{r,i}, i \in [0 \dots l-1]$ está vacía al ser heredada de $\nabla_{r,l-1}[\,]$.

[INIT]	$\$_0() \longmapsto \$_0()\ \nabla_{0,0}([\,])$	
[CALL]	$\nabla_{r,s}(X) \longmapsto \nabla_{r,s}(X)\ \overrightarrow{A_{r,s+1}}([\,])$	$A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\,]\Upsilon_2$
[SCALL]	$\nabla_{r,s}([\overrightarrow{\gamma} \mid X]) \longmapsto \nabla_{r,s}([\overrightarrow{\gamma} \mid X])\ \overrightarrow{A_{r,s+1}}([\overrightarrow{\gamma'} \mid \overrightarrow{X}])$	$A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\circ\circ\gamma']\Upsilon_2$
[SEL]	$\overrightarrow{A_{r,0}}(X) \longmapsto \nabla_{r,0}(X)$	$r \neq 0$
[PUB]	$\nabla_{r,n_r}(X) \longmapsto \overleftarrow{A_{r,0}}(X)$	
[RET]	$\nabla_{r,s}(X)\ \overleftarrow{A_{r,s+1}}([\,]) \longmapsto \nabla_{r,s+1}(X)$	$A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\,]\Upsilon_2$
[SRET]	$\nabla_{r,s}([\overrightarrow{\gamma} \mid \overrightarrow{X}])\ \overleftarrow{A_{r,s+1}}([\overleftarrow{\gamma'} \mid \overleftarrow{X}]) \longmapsto \nabla_{r,s+1}([\overleftarrow{\gamma} \mid \overleftarrow{X}])$	$A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\circ\circ\gamma']\Upsilon_2$
[SCAN]	$\overrightarrow{A_{r,0}}([\,]) \overset{a}{\longmapsto} \overleftarrow{A_{r,0}}([\,])$	$A_{r,0}[\,] \to a$

Tabla 8.4: Reglas del primer esquema de compilación genérico de LIG en RLPDA

[INIT]	$\$_0[\circ\circ] \longmapsto \$_0[\circ\circ]\ \nabla_{0,0}[\,]$	
[CALL]	$\nabla_{r,s}[\circ\circ] \longmapsto \nabla_{r,s}[\circ\circ]\ \overrightarrow{A_{r,s+1}}[\,]$	$A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\,]\Upsilon_2$
[SCALL]	$\nabla_{r,s}[\circ\circ\overrightarrow{\gamma}] \longmapsto \nabla_{r,s}[\circ\circ\overrightarrow{\gamma}]\ \overrightarrow{A_{r,s+1}}[\overrightarrow{\circ\circ}\ \overrightarrow{\gamma'}]$	$A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\circ\circ\gamma']\Upsilon_2$
[SEL]	$\overrightarrow{A_{r,0}}[\circ\circ] \longmapsto \nabla_{r,0}[\circ\circ]$	$r \neq 0$
[PUB]	$\nabla_{r,n_r}[\circ\circ] \longmapsto \overleftarrow{A_{r,0}}[\circ\circ]$	
[RET]	$\nabla_{r,s}[\circ\circ]\ \overleftarrow{A_{r,s+1}}[\,] \longmapsto \nabla_{r,s+1}[\circ\circ]$	$A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\,]\Upsilon_2$
[SRET]	$\nabla_{r,s}[\overrightarrow{\circ\circ}\ \overrightarrow{\gamma}]\ \overleftarrow{A_{r,s+1}}[\overleftarrow{\circ\circ}\ \overleftarrow{\gamma'}] \longmapsto \nabla_{r,s+1}[\overleftarrow{\circ\circ}\ \overleftarrow{\gamma}]$	$A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\circ\circ\gamma']\Upsilon_2$
[SCAN]	$\overrightarrow{A_{r,0}}[\,] \overset{a}{\longmapsto} \overleftarrow{A_{r,0}}[\,]$	$A_{r,0}[\,] \to a$

Tabla 8.5: Reglas del esquema de compilación genérico de LIG en RLPDA

estrategia-CF	estrategia-índices	$\overrightarrow{A_{r,s+1}}$	$\overrightarrow{\gamma}$	$\overrightarrow{\circ\circ}$	$\overleftarrow{A_{r,s+1}}$	$\overleftarrow{\gamma}$	$\overleftarrow{\circ\circ}$
Ascendente	ascendente	$\Box$	ϵ	ϵ	$A_{r,s+1}$	γ	$\circ\circ$
Earley		$\overline{A_{r,s+1}}$	ϵ	ϵ	$\overline{\overline{A_{r,s+1}}}$	γ	$\circ\circ$
Descendente		$A_{r,s+1}$	ϵ	ϵ	$\Box$	γ	$\circ\circ$
Ascendente	Earley	$\Box$	γ	$\circ\circ$	$A_{r,s+1}$	γ	$\circ\circ$
Earley		$\overline{A_{r,s+1}}$	γ	$\circ\circ$	$\overline{\overline{A_{r,s+1}}}$	γ	$\circ\circ$
Descendente		$A_{r,s+1}$	γ	$\circ\circ$	$\Box$	γ	$\circ\circ$
Ascendente	descendente	$\Box$	γ	$\circ\circ$	$A_{r,s+1}$	ϵ	ϵ
Earley		$\overline{A_{r,s+1}}$	γ	$\circ\circ$	$\overline{\overline{A_{r,s+1}}}$	ϵ	ϵ
Descendente		$A_{r,s+1}$	γ	$\circ\circ$	$\Box$	ϵ	ϵ

Tabla 8.6: Parámetros del esquema de compilación genérico de LIG en RLPDA

Estrategia ascendente-ascendente

Esquema de compilación 8.3 El esquema de compilación ascendente-ascendente de una gramática lineal de índices en un autómata lógico a pila restringido queda definido por el conjunto de reglas mostrado en la tabla 8.7 y por los elementos inicial $\$_0[\,]$ y final $S[\,]$. Por construcción del esquema de compilación, la pila de índices asociada a cada no terminal de llamada $\overrightarrow{A}$ es siempre $[\,]$, por lo que la regla **[SCAN]** que crea transiciones de tipo SWAP, ha sido convertida en una regla que define transiciones en las que el contenido de la pila de índices es propagado. §

Estrategia Earley-ascendente

Esquema de compilación 8.4 El esquema de compilación Earley-ascendente de una gramática lineal de índices en un autómata lógico a pila restringido utilizando propagación del contenido de la pila de índices en las reglas **[SCAN]** queda definido por el conjunto de reglas mostrado en la tabla 8.8 y por los elementos inicial $\$_0[\,]$ y final $\overline{\overline{S}}[\,]$. §

Estrategia descendente-ascendente

Esquema de compilación 8.5 El esquema de compilación descendente-ascendente de una gramática lineal de índices en un autómata lógico a pila restringido utilizando propagación del contenido de la pila de índices en la regla **[SCAN]** queda definido por el conjunto de reglas mostrado en la tabla 8.9 y por los elementos inicial $\$_0[\,]$ y final $\Box[\,]$. §

8.3.3 Estrategias $*$-Earley

Las estrategias $*$-Earley son estrategias mixtas en lo que respecta al tratamiento de la pila de índices, puesto que en la fase de llamada predice información acerca de la pila de índices mientras que en la fase de retorno propaga información de dicha pila. En los esquemas de compilación correspondientes a estas estrategias se ha aplicado una binarización implícita de las producciones de la gramática de tal modo que cada producción

$$A_{r,0}[\circ\circ\gamma] \rightarrow A_{r,1}[\,] \ldots A_{r,l}[\circ\circ\gamma'] \ldots A_{r,n_r}[\,]$$

ha sido descompuesta en las siguientes $n_r + 1$ producciones

$$\begin{aligned}
&\overrightarrow{A_{r,0}[\circ\circ]} \rightarrow \nabla_{r,0}[\circ\circ] \\
&\nabla_{r,0}[\circ\circ] \rightarrow A_{r,1}[\,]\ \nabla_{r,1}[\circ\circ] \\
&\vdots \\
&\nabla_{r,l-1}[\circ\circ\gamma] \rightarrow \overrightarrow{A_{r,l}[\circ\circ\gamma']} \\
&\overleftarrow{A_{r,l}[\circ\circ\gamma']} \rightarrow \nabla_{r,l}[\circ\circ\gamma] \\
&\nabla_{r,l}[\circ\circ] \rightarrow A_{r,l+1}[\,]\ \nabla_{r,l+1}[\circ\circ] \\
&\vdots
\end{aligned}$$

[INIT]	$\$_0[\circ\circ] \longmapsto \$_0[\circ\circ]\ \nabla_{0,0}[\,]$	
[CALL]	$\nabla_{r,s}[\circ\circ] \longmapsto \nabla_{r,s}[\circ\circ]\ \ \Box[\,]$	$A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\,]\Upsilon_2$
[SCALL]	$\nabla_{r,s}[\circ\circ] \longmapsto \nabla_{r,s}[\circ\circ]\ \ \Box[\,]$	$A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\circ\circ\gamma']\Upsilon_2$
[SEL]	$\Box[\circ\circ] \longmapsto \nabla_{r,0}[\circ\circ]$	$r \neq 0$
[PUB]	$\nabla_{r,n_r}[\circ\circ] \longmapsto A_{r,0}[\circ\circ]$	
[RET]	$\nabla_{r,s}[\circ\circ]\ \ A_{r,s+1}[\,] \longmapsto \nabla_{r,s+1}[\circ\circ]$	$A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\,]\Upsilon_2$
[SRET]	$\nabla_{r,s}[\,]\ \ A_{r,s+1}[\circ\circ\gamma'] \longmapsto \nabla_{r,s+1}[\circ\circ\gamma]$	$A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\circ\circ\gamma']\Upsilon_2$
[SCAN]	$\Box[\circ\circ] \stackrel{a}{\longmapsto} A_{r,0}[\circ\circ]$	tal que $A_{r,0}[\,] \to a$

Tabla 8.7: Reglas del esquema de compilación ascendente-ascendente de LIG en RLPDA

[INIT]	$\$_0[\circ\circ] \longmapsto \$_0[\circ\circ]\ \nabla_{0,0}[\,]$	
[CALL]	$\nabla_{r,s}[\circ\circ] \longmapsto \nabla_{r,s}[\circ\circ]\ \ \overline{A_{r,s+1}}[\,]$	$A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\,]\Upsilon_2$
[SCALL]	$\nabla_{r,s}[\circ\circ] \longmapsto \nabla_{r,s}[\circ\circ]\ \ \overline{A_{r,s+1}}[\,]$	$A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\circ\circ\gamma']\Upsilon_2$
[SEL]	$\overline{A_{r,0}}[\circ\circ] \longmapsto \nabla_{r,0}[\circ\circ]$	$r \neq 0$
[PUB]	$\nabla_{r,n_r}[\circ\circ] \longmapsto \overline{\overline{A_{r,0}}}[\circ\circ]$	
[RET]	$\nabla_{r,s}[\circ\circ]\ \ \overline{\overline{A_{r,s+1}}}[\,] \longmapsto \nabla_{r,s+1}[\circ\circ]$	$A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\,]\Upsilon_2$
[SRET]	$\nabla_{r,s}[\,]\ \ \overline{\overline{A_{r,s+1}}}[\circ\circ\gamma'] \longmapsto \nabla_{r,s+1}[\circ\circ\gamma]$	$A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\circ\circ\gamma']\Upsilon_2$
[SCAN]	$\overline{A_{r,0}}[\circ\circ] \stackrel{a}{\longmapsto} \overline{\overline{A_{r,0}}}[\circ\circ]$	tal que $A_{r,0}[\,] \to a$

Tabla 8.8: Reglas del esquema de compilación Earley-ascendente de LIG en RLPDA

[INIT]	$\$_0[\circ\circ] \longmapsto \$_0[\circ\circ]\ \nabla_{0,0}[\,]$	
[CALL]	$\nabla_{r,s}[\circ\circ] \longmapsto \nabla_{r,s}[\circ\circ]\ \ A_{r,s+1}[\,]$	$A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\,]\Upsilon_2$
[SCALL]	$\nabla_{r,s}[\circ\circ] \longmapsto \nabla_{r,s}[\circ\circ]\ \ A_{r,s+1}[\,]$	$A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\circ\circ\gamma']\Upsilon_2$
[SEL]	$A_{r,0}[\circ\circ] \longmapsto \nabla_{r,0}[\circ\circ]$	$r \neq 0$
[PUB]	$\nabla_{r,n_r}[\circ\circ] \longmapsto \Box[\circ\circ]$	
[RET]	$\nabla_{r,s}[\circ\circ]\ \ \Box[\,] \longmapsto \nabla_{r,s+1}[\circ\circ]$	$A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\,]\Upsilon_2$
[SRET]	$\nabla_{r,s}[\,]\ \ \Box[\circ\circ\gamma'] \longmapsto \nabla_{r,s+1}[\circ\circ\gamma]$	$A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\circ\circ\gamma']\Upsilon_2$
[SCAN]	$A_{r,0}[\circ\circ] \stackrel{a}{\longmapsto} \Box[\circ\circ]$	tal que $A_{r,0}[\,] \to a$

Tabla 8.9: Reglas del esquema de compilación descendente-ascendente de LIG en RLPDA

$$\nabla_{r,n_r-1}[\circ\circ] \to A_{r,n_r}[\,]\ \nabla_{r,n_r}[\circ\circ]$$
$$\nabla_{r,n_r}[\circ\circ] \to \overleftarrow{A_{r,0}[\circ\circ]}$$

Estrategia ascendente-Earley

Esquema de compilación 8.6 El esquema de compilación ascendente-Earley de una gramática lineal de índices en un autómata lógico a pila restringido queda definido en notación LIG por el conjunto de reglas mostrado en la tabla 8.10 y por los elementos inicial $\$_0[\,]$ y final $S[\,]$. §

Especial atención merece la regla **[SRET]**, en la que las pilas $\circ\circ_1$ y $\circ\circ_2$ contienen los mismos elementos en el mismo orden y por consiguiente son unificables pero son pilas distintas, puesto que $\circ\circ_1$ es la pila predicha en la fase de llamada y $\circ\circ_2$ es la pila propagada en la fase de retorno. La igualdad de sus contenidos viene determinada por la definición de la espina de las gramáticas lineales de índices, pues dada una derivación, la pila en la espina puede ser construida de modo descendente o ascendente, siendo los valores almacenados en la misma idénticos en ambos casos [31, 32].

Podremos prescindir de los subíndices que diferencian la pila de índices de llamada y la de retorno en la regla de compilación **[SRET]** en aquellos casos en los que se entienda claramente el papel de cada una de las pilas de índices. En tales casos la regla

$$\nabla_{r,s}[\circ\circ\gamma]\ \overline{\overline{A_{r,s+1}}}[\circ\circ\gamma'] \longmapsto \nabla_{r,s+1}[\circ\circ\gamma]$$

actuaría como abreviatura de la regla de compilación original.

Al contrario de lo que ocurre con otras estrategias, no es posible convertir la regla **[SCAN]** en $\overline{A}[\circ\circ] \overset{a}{\longmapsto} \overline{\overline{A_{r,0}}}[\circ\circ]$ ya que es preciso comprobar que la pila está vacía en el momento de realizar el reconocimiento de un terminal puesto que el esquema de compilación no impone ninguna restricción en el contenido de la pila ni en la fase de llamada (tal como hacen los esquemas de compilación $*$-ascendentes, que predicen una pila vacía) ni en la fase de retorno (tal como hacen los esquemas $*$-descendentes, que imponen la propagación de una pila de índices vacía). De no hacerlo así cualquier pila predicha en la fase de llamada podría ser erróneamente propagada en la fase de retorno, con lo cual el esquema de compilación no sería correcto.

Estrategia Earley-Earley

Esquema de compilación 8.7 El esquema de compilación Earley-Earley de una gramática lineal de índices en un autómata lógico a pila restringido queda definido por el conjunto de reglas mostrado en la tabla 8.11 y por los elementos inicial $\$_0[\,]$ y final $\overline{\overline{S}}[\,]$. §

Estrategia descendente-Earley

Esquema de compilación 8.8 El esquema de compilación Earley-Earley de una gramática lineal de índices en un autómata lógico a pila restringido queda definido por el conjunto de reglas mostrado en la tabla 8.12 y por los elementos inicial $\$_0[\,]$ y final $\Box[\,]$. §

[INIT]	$\$_0[\circ\circ] \longmapsto \$_0[\circ\circ]\ \nabla_{0,0}[\,]$	
[CALL]	$\nabla_{r,s}[\circ\circ] \longmapsto \nabla_{r,s}[\circ\circ]\ \ \Box[\,]$	$A_{r,0}[\circ\circ\gamma] \rightarrow \Upsilon_1 A_{r,s+1}[\,]\Upsilon_2$
[SCALL]	$\nabla_{r,s}[\circ\circ\gamma] \longmapsto \nabla_{r,s}[\circ\circ\gamma]\ \ \Box[\circ\circ\gamma']$	$A_{r,0}[\circ\circ\gamma] \rightarrow \Upsilon_1 A_{r,s+1}[\circ\circ\gamma']\Upsilon_2$
[SEL]	$\Box[\circ\circ] \longmapsto \nabla_{r,0}[\circ\circ]$	$r \neq 0$
[PUB]	$\nabla_{r,n_r}[\circ\circ] \longmapsto A_{r,0}[\circ\circ]$	
[RET]	$\nabla_{r,s}[\circ\circ]\ \ A_{r,s+1}[\,] \longmapsto \nabla_{r,s+1}[\circ\circ]$	$A_{r,0}[\circ\circ\gamma] \rightarrow \Upsilon_1 A_{r,s+1}[\,]\Upsilon_2$
[SRET]	$\nabla_{r,s}[\circ\circ_1\gamma]\ \ A_{r,s+1}[\circ\circ_2\gamma'] \longmapsto \nabla_{r,s+1}[\circ\circ_2\gamma]$	$A_{r,0}[\circ\circ\gamma] \rightarrow \Upsilon_1 A_{r,s+1}[\circ\circ\gamma']\Upsilon_2$
[SCAN]	$\Box[\,] \overset{a}{\longmapsto} A_{r,0}[\,]$	tal que $A_{r,0}[\,] \rightarrow a$

Tabla 8.10: Reglas del esquema de compilación ascendente-Earley de LIG en RLPDA

[INIT]	$\$_0[\circ\circ] \longmapsto \$_0[\circ\circ]\ \nabla_{0,0}[\,]$	
[CALL]	$\nabla_{r,s}[\circ\circ] \longmapsto \nabla_{r,s}[\circ\circ]\ \ \overline{A_{r,s+1}}[\,]$	$A_{r,0}[\circ\circ\gamma] \rightarrow \Upsilon_1 A_{r,s+1}[\,]\Upsilon_2$
[SCALL]	$\nabla_{r,s}[\circ\circ\gamma] \longmapsto \nabla_{r,s}[\circ\circ\gamma]\ \ \overline{A_{r,s+1}}[\circ\circ\gamma']$	$A_{r,0}[\circ\circ\gamma] \rightarrow \Upsilon_1 A_{r,s+1}[\circ\circ\gamma']\Upsilon_2$
[SEL]	$\overline{A_{r,s+1}}[\circ\circ] \longmapsto \nabla_{r,0}[\circ\circ]$	$r \neq 0$
[PUB]	$\nabla_{r,n_r}[\circ\circ] \longmapsto \overline{\overline{A_{r,0}}}[\circ\circ]$	
[RET]	$\nabla_{r,s}[\circ\circ]\ \ \overline{\overline{A_{r,s+1}}}[\,] \longmapsto \nabla_{r,s+1}[\circ\circ]$	$A_{r,0}[\circ\circ\gamma] \rightarrow \Upsilon_1 A_{r,s+1}[\,]\Upsilon_2$
[SRET]	$\nabla_{r,s}[\circ\circ_1\gamma]\ \ \overline{\overline{A_{r,s+1}}}[\circ\circ_2\gamma'] \longmapsto \nabla_{r,s+1}[\circ\circ_2\gamma]$	$A_{r,0}[\circ\circ\gamma] \rightarrow \Upsilon_1 A_{r,s+1}[\circ\circ\gamma']\Upsilon_2$
[SCAN]	$\overline{A}[\,] \overset{a}{\longmapsto} \overline{\overline{A_{r,0}}}[\,]$	tal que $A_{r,0}[\,] \rightarrow a$

Tabla 8.11: Reglas del esquema de compilación Earley-Earley de LIG en RLPDA

[INIT]	$\$_0[\circ\circ] \longmapsto \$_0[\circ\circ]\ \nabla_{0,0}[\,]$	
[CALL]	$\nabla_{r,s}[\circ\circ] \longmapsto \nabla_{r,s}[\circ\circ]\ \ A_{r,s+1}[\,]$	$A_{r,0}[\circ\circ\gamma] \rightarrow \Upsilon_1 A_{r,s+1}[\,]\Upsilon_2$
[SCALL]	$\nabla_{r,s}[\circ\circ\gamma] \longmapsto \nabla_{r,s}[\circ\circ\gamma]\ \ A_{r,s+1}[\circ\circ\gamma']$	$A_{r,0}[\circ\circ\gamma] \rightarrow \Upsilon_1 A_{r,s+1}[\circ\circ\gamma']\Upsilon_2$
[SEL]	$A_{r,s+1}[\circ\circ] \longmapsto \nabla_{r,0}[\circ\circ]$	$r \neq 0$
[PUB]	$\nabla_{r,n_r}[\circ\circ] \longmapsto \Box[\circ\circ]$	
[RET]	$\nabla_{r,s}[\circ\circ]\ \ \Box[\,] \longmapsto \nabla_{r,s+1}[\circ\circ]$	$A_{r,0}[\circ\circ\gamma] \rightarrow \Upsilon_1 A_{r,s+1}[\,]\Upsilon_2$
[SRET]	$\nabla_{r,s}[\circ\circ_1\gamma]\ \ \Box[\circ\circ_2\gamma'] \longmapsto \nabla_{r,s+1}[\circ\circ_2\gamma]$	$A_{r,0}[\circ\circ\gamma] \rightarrow \Upsilon_1 A_{r,s+1}[\circ\circ\gamma']\Upsilon_2$
[SCAN]	$A[\,] \overset{a}{\longmapsto} \Box[\,]$	tal que $A_{r,0}[\,] \rightarrow a$

Tabla 8.12: Reglas del esquema de compilación descendente-Earley de LIG en RLPDA

8.3.4 Estrategias $*$-descendentes

En una estrategia de análisis descendente, existe una fase de llamada en la que se predice toda la información posible y una fase de retorno en la que se propaga la mínima información posible. Esto se traduce, en el caso concreto de las estrategias $*$-descendentes, en que la fase de llamada predice la información correspondiente a las pilas de índices, mientras que no propaga ninguna información acerca de dichas pilas en la fase de retorno. A continuación mostramos diversos esquemas de compilación para estas estrategias. En estos esquemas se ha aplicado una binarización implícita de las producciones de la gramática de tal modo que una producción

$$A_{r,0}[\circ\circ\gamma] \to A_{r,1}[\,] \ldots A_{r,l}[\circ\circ\gamma'] \ldots A_{r,n_r}[\,]$$

se ha descompuesto en las siguientes $n_r + 1$ producciones

$$\begin{array}{l}
A_{r,0}[\circ\circ\gamma] \to \nabla_{r,0}[\circ\circ\gamma] \\
\nabla_{r,0}[\circ\circ\gamma] \to A_{r,1}[\,]\ \nabla_{r,1}[\circ\circ\gamma] \\
\vdots \\
\nabla_{r,l-1}[\circ\circ\gamma] \to A_{r,l}[\circ\circ\gamma']\ \nabla_{r,l}[\,] \\
\nabla_{r,l}[\circ\circ] \to A_{r,l+1}[\,]\ \nabla_{r,l+1}[\circ\circ] \\
\vdots \\
\nabla_{r,n_r-1}[\circ\circ] \to A_{r,n_r}[\,]\ \nabla_{r,n_r}[\circ\circ] \\
\nabla_{r,n_r}[\,] \to \epsilon
\end{array}$$

donde la pila de índices asociada a los $\nabla_{r,i}, i \in [l \ldots n_r]$ está vacía al ser heredada de $\nabla_{r,l}[\,]$.

Estrategia ascendente-descendente

Esquema de compilación 8.9 El esquema de compilación ascendente-descendente de una gramática lineal de índices en un autómata lógico a pila restringido queda definido por el conjunto de reglas mostrado en la tabla 8.13 y por los elementos inicial $\$_0[\ ]$ y final $S[\ ]$. Por construcción del esquema de compilación, la pila de índices asociada a cada no terminal de retorno $\overleftarrow{A}$ es siempre $[\,]$, por lo que la regla **[SCAN]** que crea transiciones de tipo SWAP, ha sido convertida en una regla que define transiciones en las que el contenido de la pila de índices es propagado. §

Estrategia Earley-descendente

Esquema de compilación 8.10 El esquema de compilación Earley-descendente de una gramática lineal de índices en un autómata lógico a pila restringido utilizando propagación del contenido de la pila de índices en las reglas **[SCAN]** queda definido por el conjunto de reglas mostrado en la tabla 8.14 y por los elementos inicial $\$_0[\,]$ y final $\overline{\overline{S}}[\,]$. §

[INIT]	$\$_0[\circ\circ] \longmapsto \$_0[\circ\circ]\ \nabla_{0,0}[\,]$	
[CALL]	$\nabla_{r,s}[\circ\circ] \longmapsto \nabla_{r,s}[\circ\circ]\ \ \Box[\,]$	$A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\,]\Upsilon_2$
[SCALL]	$\nabla_{r,s}[\circ\circ\gamma] \longmapsto \nabla_{r,s}[\circ\circ\gamma]\ \ \Box[\circ\circ\gamma']$	$A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\circ\circ\gamma']\Upsilon_2$
[SEL]	$\Box[\circ\circ] \longmapsto \nabla_{r,0}[\circ\circ]$	$r \neq 0$
[PUB]	$\nabla_{r,n_r}[\circ\circ] \longmapsto A_{r,0}[\circ\circ]$	
[RET]	$\nabla_{r,s}[\circ\circ]\ \ A_{r,s+1}[\,] \longmapsto \nabla_{r,s+1}[\circ\circ]$	$A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\,]\Upsilon_2$
[SRET]	$\nabla_{r,s}[\circ\circ\gamma]\ \ A_{r,s+1}[\,] \longmapsto \nabla_{r,s+1}[\,]$	$A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\circ\circ\gamma']\Upsilon_2$
[SCAN]	$\Box[\,] \overset{a}{\longmapsto}\ A_{r,0}[\,]$	$A_{r,0}[\,] \to a$

Tabla 8.13: Reglas del esquema de compilación ascendente-descendente de LIG en RLPDA

[INIT]	$\$_0[\circ\circ] \longmapsto \$_0[\circ\circ]\ \nabla_{0,0}[\,]$	
[CALL]	$\nabla_{r,s}[\circ\circ] \longmapsto \nabla_{r,s}[\circ\circ]\ \ \overline{A_{r,s+1}}[\,]$	$A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\,]\Upsilon_2$
[SCALL]	$\nabla_{r,s}[\circ\circ\gamma] \longmapsto \nabla_{r,s}[\circ\circ\gamma]\ \ \overline{A_{r,s+1}}[\circ\circ\gamma']$	$A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\circ\circ\gamma']\Upsilon_2$
[SEL]	$\overline{A_{r,0}}[\circ\circ] \longmapsto \nabla_{r,0}[\circ\circ]$	$r \neq 0$
[PUB]	$\nabla_{r,n_r}[\circ\circ] \longmapsto \overline{\overline{A_{r,0}}}[\circ\circ]$	
[RET]	$\nabla_{r,s}[\circ\circ]\ \ \overline{\overline{A_{r,s+1}}}[\,] \longmapsto \nabla_{r,s+1}[\circ\circ]$	$A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\,]\Upsilon_2$
[SRET]	$\nabla_{r,s}[\circ\circ\gamma]\ \ \overline{\overline{A_{r,s+1}}}[\,] \longmapsto \nabla_{r,s+1}[\,]$	$A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\circ\circ\gamma']\Upsilon_2$
[SCAN]	$\overline{A_{r,0}}[\,] \overset{a}{\longmapsto}\ \overline{\overline{A_{r,0}}}[\,]$	$A_{r,0}[\,] \to a$

Tabla 8.14: Reglas del esquema de compilación Earley-descendente de LIG en RLPDA

[INIT]	$\$_0[\circ\circ] \longmapsto \$_0[\circ\circ]\ \nabla_{0,0}[\,]$	
[CALL]	$\nabla_{r,s}[\circ\circ] \longmapsto \nabla_{r,s}[\circ\circ]\ \ A_{r,s+1}[\,]$	$A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\,]\Upsilon_2$
[SCALL]	$\nabla_{r,s}[\circ\circ\gamma] \longmapsto \nabla_{r,s}[\circ\circ\gamma]\ \ A_{r,s+1}[\circ\circ\gamma']$	$A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\circ\circ\gamma']\Upsilon_2$
[SEL]	$A_{r,0}[\circ\circ] \longmapsto \nabla_{r,0}[\circ\circ]$	$r \neq 0$
[PUB]	$\nabla_{r,n_r}[\circ\circ] \longmapsto \Box[\circ\circ]$	$r \neq 0$
[RET]	$\nabla_{r,s}[\circ\circ]\ \ \Box[\,] \longmapsto \nabla_{r,s+1}[\circ\circ]$	$A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\,]\Upsilon_2$
[SRET]	$\nabla_{r,s}[\circ\circ\gamma]\ \ \Box[\,] \longmapsto \nabla_{r,s+1}[\,]$	$A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\circ\circ\gamma']\Upsilon_2$
[SCAN]	$A_{r,0}[\,] \overset{a}{\longmapsto}\ \Box[\,]$	$A_{r,0}[\,] \to a$

Tabla 8.15: Reglas del esquema de compilación descendente-descendente de LIG en RLPDA

Estrategia descendente-descendente

Esquema de compilación 8.11 El esquema de compilación descendente-descendente de una gramática lineal de índices en un autómata lógico a pila restringido utilizando propagación del contenido de la pila de índices en las reglas **[SCAN]** queda definido por el conjunto de reglas mostrado en la tabla 8.15 y por los elementos inicial $\$_0[\,]$ y final $\Box[\,]$. §

8.4 Estrategias de análisis de gramáticas de adjunción de árboles

Los autómatas lógicos a pila restringidos también son adecuadas para el análisis de gramáticas de adjunción de árboles. Para ello dotaremos de una nueva semántica a los elementos $A[\alpha]$ almacenados en la pila, puesto que en lugar de considerarlos elementos gramaticales de una gramática lineal de índices, consideraremos que A indica un nodo de un árbol elemental y que α indica la pila de adjunciones pendientes en dicho nodo.

De forma análoga a como hicimos en el caso de LIG, definiremos una serie de esquemas de compilación basados en el paradigma llamada/retorno [55], utilizando para ello los 12 tipos de reglas de compilación que se muestran en la tabla 8.16, cuyo significado intuitivo se muestra gráficamente en la figura 6.6 de la página 158. Las reglas de compilación **[INIT]**, **[CALL]**, **[SCALL]**, **[SEL]**, **[ACALL]** y **[FCALL]** serán las encargadas de crear las transiciones que definirán la *fase de llamada* del autómata, mientras que las reglas de compilación **[RET]**, **[SRET]**, **[PUB]**, **[ARET]** y **[FRET]** serán las encargadas de crear las transiciones que definirán la *fase de retorno* del autómata.

Las estrategias utilizadas en el análisis de gramáticas de adjunción de árboles se definen mediante un par $\langle$*estrategia-CF*$\rangle - \langle$*estrategia-adjunción*$\rangle$ en el que *estrategia-CF* se refiere a la estrategia utilizada para el recorrido de los árboles elementales, que puede ser:

descendente si se predice el nodo que va a ser visitado pero no se propaga información acerca del nodo que acaba de ser visitado.

ascendente si no se predice el nodo que va a ser visitado pero se propaga información acerca del nodo que acaba de ser visitado.

Earley si se predice el nodo que va a ser visitado y se propaga información acerca del nodo que acaba de ser visitado.

mientras que *estrategia-adjunción* se refiere a la estrategia utilizada para el tratamiento de las adjunciones, que puede ser:

descendente si se almacena información del nodo de adjunción cuando se inicia la visita de la raíz del árbol auxiliar adjuntado.

ascendente si se almacena información del nodo de adjunción cuando se termina de visitar el nodo pie del árbol auxiliar adjuntado.

Earley si se almacena información del nodo de adjunción tanto cuando se inicia la visita de la raíz como cuando se termina de visitar el nodo pie del árbol auxiliar adjuntado.

Regla	Tarea
[INIT]	inicia los cálculos a partir de la pila inicial.
[CALL]	requiere el análisis de un determinado nodo que no forma parte de la espina de un árbol auxiliar.
[SCALL]	(de *spine call*) requiere el análisis de un determinado nodo que forma parte de la espina de un árbol auxiliar.
[SEL]	selecciona una producción de un árbol elemental.
[PUB]	determina que un nodo ha sido analizado.
[RET]	continua el proceso de análisis después del reconocimiento de un nodo que no forma parte de la espina de un árbol auxiliar.
[SRET]	(de *spine ret*) continua el proceso de análisis después del reconocimiento de un nodo que forma parte de la espina de un árbol auxiliar.
[SCAN]	reconoce los terminales que componen la cadena de entrada.
[ACALL]	inicia la operación de adjunción de un árbol a un nodo.
[ARET]	termina la operación de adjunción de un árbol a un nodo.
[FCALL]	comienza a reconocer la parte escindida de un árbol elemental que debe ser pegada al nodo pie de un árbol auxiliar.
[FRET]	termina de reconocer el nodo pie de un árbol auxiliar.

Tabla 8.16: Reglas para los esquemas de compilación de gramáticas de adjunción de árboles

Para referirnos a todas las estrategias con un determinado comportamiento con respecto al recorrido de los árboles elementales, independientemente del comportamiento con respecto al tratamiento de la adjunción, utilizaremos la notación $\langle$*estrategia-CF*$\rangle - *$. Para referirnos al conjunto de estrategias que presentan un determinado comportamiento con respecto al tratamiento de las adjunciones, independientemente del comportamiento con respecto al recorrido de los árboles elementales, utilizaremos la notación $* - \langle$*estrategia-adjunción*$\rangle$. Con ello logramos homogeneizar la nomenclatura de las estrategias de análisis, tanto para el caso de las TAG como de las LIG.

8.4.1 Estrategia genérica

En primer lugar definiremos una estrategia genérica basada en el paradigma llamada/retorno, parametrizada con respecto a la información que se predice y propaga en las fases de llamada y de retorno, respectivamente. Utilizaremos la siguiente notación:

- $\overrightarrow{N^\gamma_{r,s}}$ para referirnos a la predicción de información con respecto al nodo $N^\gamma_{r,s}$ del árbol elemental γ.
- $\overrightarrow{\circ\circ}$ representa la información predicha de una pila completa de adjunciones pendientes.
- $\overleftarrow{N^\gamma_{r,s}}$ para representar la información propagada con respecto al nodo $N^\gamma_{r,s}$ del árbol elemental γ.
- $\overleftarrow{\circ\circ}$ representa la información propagada de una pila completa de adjunciones pendientes.

Esquema de compilación 8.12 El esquema de compilación genérico de una gramática de adjunción de árboles en un autómata lógico a pila restringido queda definido por el conjunto de reglas mostrado en la tabla 8.17 y los elementos inicial $\$_0[\,]$ y final $\overleftarrow{\top^\alpha}[\,]$, con $\alpha \in \boldsymbol{I}$. §

En la tabla 8.18 se muestran los valores que deben tomar los parámetros de predicción y propagación de información para instanciar el esquema de compilación genérico en esquemas correspondientes a diferentes estrategias de análisis. En el caso de estrategias Earley-$*$ es necesario distinguir la llamada de un nodo $N^\gamma_{r,s+1}$ de su retorno, para lo cual utilizamos los símbolos $\overline{N^\gamma_{r,s+1}}$ y $\overline{\overline{N^\gamma_{r,s+1}}}$. En aquellas estrategias en las que $\overrightarrow{\circ\circ}$ y $\overleftarrow{\circ\circ}$ son ambos distintos de ϵ, deben de ser incluidas restricciones adicionales en la regla de compilación **[SRET]** que establezcan que el contenido de $\overrightarrow{\circ\circ}$ y $\overleftarrow{\circ\circ}$ debe unificar, es decir, ambas pilas de adjunciones pendientes deben contener los mismos nodos.

A continuación se muestran los esquemas de compilación correspondientes a las diferentes estrategias de análisis, según los valores de la tabla 8.18.

8.4.2 Estrategias $*$-ascendentes

Las estrategias $*$-ascendentes se caracterizan por no almacenar ninguna información acerca del nodo de adjunción en el momento de predecir una adjunción. En cambio, cuando se ha terminado de reconocer el subárbol que cuelga de un nodo de adjunción N^γ y se trata de pasar al nodo pie de un árbol auxiliar β que puede ser adjuntado en dicho nodo, se guarda información acerca de N^γ, información que será propagada ascendentemente por la espina de β hasta llegar a la raíz, momento en el cual esa información será utilizada para garantizar que continuamos el análisis en el nodo N^γ.

Estrategia ascendente-ascendente

Esquema de compilación 8.13 El esquema de compilación ascendente-ascendente de una gramática de adjunción de árboles en un autómata lógico a pila restringido queda definido por el conjunto de reglas mostrado en la tabla 8.19 y por los elementos inicial $\$_0[\,]$ y final $\top^\alpha[\,]$, con $\alpha \in \boldsymbol{I}$. Con respecto al esquema de compilación genérico, la regla de compilación **[SCAN]** ha sido modificada aunque dicha modificación no afecta a su comportamiento puesto que por construcción del esquema de compilación, la pila de índices asociada a cada nodo $\overrightarrow{N^\gamma}$ es siempre $[\,]$. §

[INIT]	$\$_0[\circ\circ] \longmapsto \$_0[\circ\circ]\ \nabla^{\alpha}_{0,0}[\,]$	$\alpha \in \boldsymbol{I}$
[CALL]	$\nabla^{\gamma}_{r,s}[\circ\circ] \longmapsto \nabla^{\gamma}_{r,s}[\circ\circ]\ \ \overrightarrow{N^{\gamma}_{r,s+1}}[\,]$	$N^{\gamma}_{r,s+1} \notin \text{espina}(\gamma),\ \ \mathbf{nil} \in \text{adj}(N_{r,s+1})$
[SCALL]	$\nabla^{\gamma}_{r,s}[\circ\circ] \longmapsto \nabla^{\gamma}_{r,s}[\circ\circ]\ \ \overrightarrow{N^{\gamma}_{r,s+1}}[\overrightarrow{\circ\circ}]$	$N^{\gamma}_{r,s+1} \in \text{espina}(\gamma),\ \ \mathbf{nil} \in \text{adj}(N_{r,s+1})$
[SEL]	$\overrightarrow{N^{\gamma}_{r,0}}[\circ\circ] \longmapsto \nabla^{\gamma}_{r,0}[\circ\circ]$	$r \neq 0$
[PUB]	$\nabla^{\gamma}_{r,n_r}[\circ\circ] \longmapsto \overleftarrow{N^{\gamma}_{r,0}}[\circ\circ]$	
[RET]	$\nabla^{\gamma}_{r,s}[\circ\circ]\ \ \overleftarrow{N^{\gamma}_{r,s+1}}[\,] \longmapsto \nabla^{\gamma}_{r,s+1}[\circ\circ]$	$N^{\gamma}_{r,s+1} \notin \text{espina}(\gamma),\ \ \mathbf{nil} \in \text{adj}(N_{r,s+1})$
[SRET]	$\nabla^{\gamma}_{r,s}[\circ\circ]\ \ \overleftarrow{N^{\gamma}_{r,s+1}}[\overleftarrow{\circ\circ}] \longmapsto \nabla^{\gamma}_{r,s+1}[\overleftarrow{\circ\circ}]$	$N^{\gamma}_{r,s+1} \in \text{espina}(\gamma),\ \ \mathbf{nil} \in \text{adj}(N_{r,s+1})$
[SCAN]	$\overrightarrow{N^{\gamma}_{r,0}}[\,] \overset{a}{\longmapsto} \overleftarrow{N^{\gamma}_{r,0}}[\,]$	$N^{\gamma}_{r,0}[\,] \to a$
[ACALL]	$\nabla^{\gamma}_{r,s}[\circ\circ] \longmapsto \nabla^{\gamma}_{r,s}[\circ\circ]\ \ \overrightarrow{\top^{\beta}}[\overrightarrow{\circ\circ}\,\overrightarrow{N^{\gamma}_{r,s+1}}]$	$\beta \in \text{adj}(N^{\gamma}_{r,s+1})$
[ARET]	$\nabla^{\gamma}_{r,s}[\overrightarrow{\circ\circ}]\ \ \overleftarrow{\top^{\beta}}[\overleftarrow{\circ\circ}\,\overleftarrow{N^{\gamma}_{r,s+1}}] \longmapsto \nabla^{\gamma}_{r,s+1}[\overleftarrow{\circ\circ}]$	$\beta \in \text{adj}(N^{\gamma}_{r,s+1})$
[FCALL]	$\nabla^{\beta}_{f,0}[\circ\circ\overrightarrow{N^{\gamma}_{r,s+1}}] \longmapsto \nabla^{\beta}_{f,0}[\circ\circ\overrightarrow{N^{\gamma}_{r,s+1}}]\ \ \overrightarrow{N^{\gamma}_{r,s+1}}[\overrightarrow{\circ\circ}]$	$N^{\beta}_{f,0} = \mathbf{F}^{\beta},\ \beta \in \text{adj}(N^{\gamma}_{r,s+1})$
[FRET]	$\nabla^{\beta}_{f,0}[\overrightarrow{\circ\circ}\,\overrightarrow{N^{\gamma}_{r,s+1}}]\ \ \overleftarrow{N^{\gamma}_{r,s+1}}[\overleftarrow{\circ\circ}] \longmapsto \nabla^{\beta}_{f,1}[\overleftarrow{\circ\circ}\,\overleftarrow{N^{\gamma}_{r,s+1}}]$	$N^{\beta}_{f,0} = \mathbf{F}^{\beta},\ \beta \in \text{adj}(N^{\gamma}_{r,s+1})$

Tabla 8.17: Reglas del esquema de compilación genérico de TAG en RLPDA

Estrategia-CF	**Estrategia-adjunción**	$\overrightarrow{N^{\gamma}_{r,s+1}}$	$\overrightarrow{\circ\circ}\,\overrightarrow{N^{\gamma}_{r,s+1}}$	$\overleftarrow{N^{\gamma}_{r,s+1}}$	$\overleftarrow{\circ\circ}\,\overleftarrow{N^{\gamma}_{r,s+1}}$
Ascendente		$\square$	ϵ	$N^{\gamma}_{r,s+1}$	$\circ\circ N^{\gamma}_{r,s+1}$
Earley	ascendente	$\overline{N^{\gamma}_{r,s+1}}$	ϵ	$\overline{\overline{N^{\gamma}_{r,s+1}}}$	$\circ\circ N^{\gamma}_{r,s+1}$
Descendente		$N^{\gamma}_{r,s+1}$	ϵ	$\square$	$\circ\circ N^{\gamma}_{r,s+1}$
Ascendente		$\square$	$\circ\circ N^{\gamma}_{r,s+1}$	$N^{\gamma}_{r,s+1}$	$\circ\circ N^{\gamma}_{r,s+1}$
Earley	Earley	$\overline{N^{\gamma}_{r,s+1}}$	$\circ\circ N^{\gamma}_{r,s+1}$	$\overline{\overline{N^{\gamma}_{r,s+1}}}$	$\circ\circ N^{\gamma}_{r,s+1}$
Descendente		$N^{\gamma}_{r,s+1}$	$\circ\circ N^{\gamma}_{r,s+1}$	$\square$	$\circ\circ N^{\gamma}_{r,s+1}$
Ascendente		$\square$	$\circ\circ N^{\gamma}_{r,s+1}$	$N^{\gamma}_{r,s+1}$	ϵ
Earley	descendente	$\overline{N^{\gamma}_{r,s+1}}$	$\circ\circ N^{\gamma}_{r,s+1}$	$\overline{\overline{N^{\gamma}_{r,s+1}}}$	ϵ
Descendente		$N^{\gamma}_{r,s+1}$	$\circ\circ N^{\gamma}_{r,s+1}$	$\square$	ϵ

Tabla 8.18: Parámetros del esquema de compilación genérico de TAG en RLPDA

Estrategia Earley-ascendente

Esquema de compilación 8.14 El esquema de compilación Earley-ascendente de una gramática de adjunción de árboles en un autómata lógico a pila restringido queda definido por el conjunto de reglas mostrado en la tabla 8.20 y por los elementos inicial $\$_0[\,]$ y final $\overline{\overline{\top^\alpha}}[\,]$, con $\alpha \in \boldsymbol{I}$. §

Estrategia descendente-ascendente

Esquema de compilación 8.15 El esquema de compilación descendente-ascendente de una gramática de adjunción de árboles en un autómata lógico a pila restringido queda definido por el conjunto de reglas mostrado en la tabla 8.21 y por los elementos inicial $\$_0[\,]$ y final $\Box[\,]$. §

8.4.3 Estrategias $*$-Earley

Las estrategias $*$-Earley se caracterizan porque almacenan información acerca del nodo de adjunción en el momento de predecir una adjunción y también en el momento en el que se retorna al nodo pie del árbol auxiliar una vez que ha sido recorrido el subárbol que cuelga del nodo de adjunción. En consecuencia, se chequea la consistencia de los árboles involucrados en una adjunción tanto en la fase de llamada como en la fase de retorno.

Estrategia ascendente-Earley

Esquema de compilación 8.16 El esquema de compilación ascendente-Earley de una gramática de adjunción de árboles en un autómata lógico a pila restringido queda definido por el conjunto de reglas mostrado en la tabla 8.22 y por los elementos inicial $\$_0[\,]$ y final $\top^\alpha[\,]$, con $\alpha \in \boldsymbol{I}$. §

Estrategia Earley-Earley

Esquema de compilación 8.17 El esquema de compilación Earley-Earley de una gramática de adjunción de árboles en un autómata lógico a pila restringido queda definido por el conjunto de reglas mostrado en la tabla 8.23 y por los elementos inicial $\$_0[\,]$ y final $\overline{\overline{\top^\alpha}}[\,]$, con $\alpha \in \boldsymbol{I}$. §

Estrategia descendente-Earley

Esquema de compilación 8.18 El esquema de compilación descendente-Earley de una gramática de adjunción de árboles en un autómata lógico a pila restringido queda definido por el conjunto de reglas mostrado en la tabla 8.24 y por los elementos inicial $\$_0[\,]$ y final $\Box[\,]$. §

[INIT]	$\$_0[\circ\circ] \longmapsto \$_0[\circ\circ]\ \nabla^{\alpha}_{0,0}[\,]$	$\alpha \in \boldsymbol{I}$
[CALL]	$\nabla^{\gamma}_{r,s}[\circ\circ] \longmapsto \nabla^{\gamma}_{r,s}[\circ\circ]\ \ \Box[\,]$	$N^{\gamma}_{r,s+1} \notin \text{espina}(\gamma),\ \ \mathbf{nil} \in \text{adj}(N_{r,s+1})$
[SCALL]	$\nabla^{\gamma}_{r,s}[\circ\circ] \longmapsto \nabla^{\gamma}_{r,s}[\circ\circ]\ \ \Box[\,]$	$N^{\gamma}_{r,s+1} \in \text{espina}(\gamma),\ \ \mathbf{nil} \in \text{adj}(N_{r,s+1})$
[SEL]	$\Box[\circ\circ] \longmapsto \nabla^{\gamma}_{r,0}[\circ\circ]$	$r \neq 0$
[PUB]	$\nabla^{\gamma}_{r,n_r}[\circ\circ] \longmapsto N^{\gamma}_{r,0}[\circ\circ]$	
[RET]	$\nabla^{\gamma}_{r,s}[\circ\circ]\ \ N^{\gamma}_{r,s+1}[\,] \longmapsto \nabla^{\gamma}_{r,s+1}[\circ\circ]$	$N^{\gamma}_{r,s+1} \notin \text{espina}(\gamma),\ \ \mathbf{nil} \in \text{adj}(N_{r,s+1})$
[SRET]	$\nabla^{\gamma}_{r,s}[\,]\ \ N^{\gamma}_{r,s+1}[\circ\circ] \longmapsto \nabla^{\gamma}_{r,+1}[\circ\circ]$	$N^{\gamma}_{r,s+1} \in \text{espina}(\gamma),\ \ \mathbf{nil} \in \text{adj}(N_{r,s+1})$
[SCAN]	$\Box[\circ\circ] \overset{a}{\longmapsto}\ N^{\gamma}_{r,0}[\circ\circ]$	$N^{\gamma}_{r,0}[\,] \rightarrow a$
[ACALL]	$\nabla^{\gamma}_{r,s}[\circ\circ] \longmapsto \nabla^{\gamma}_{r,s}[\circ\circ]\ \ \Box[\,]$	$\beta \in \text{adj}(N^{\gamma}_{r,s+1})$
[ARET]	$\nabla^{\gamma}_{r,s}[\,]\ \ \top^{\beta}[\circ\circ N^{\gamma}_{r,s+1}] \longmapsto \nabla^{\gamma}_{r,s+1}[\circ\circ]$	$\beta \in \text{adj}(N^{\gamma}_{r,s+1})$
[FCALL]	$\nabla^{\beta}_{f,0}[\circ\circ] \longmapsto \nabla^{\beta}_{f,0}[\circ\circ]\ \ \Box[\,]$	$N^{\beta}_{f,0} = \mathbf{F}^{\beta},\ \beta \in \text{adj}(N^{\gamma}_{r,s+1})$
[FRET]	$\nabla^{\beta}_{f,0}[\,]\ \ N^{\gamma}_{r,s+1}[\circ\circ] \longmapsto \nabla^{\beta}_{f,1}[\circ\circ N^{\gamma}_{r,s+1}]$	$N^{\beta}_{f,0} = \mathbf{F}^{\beta},\ \beta \in \text{adj}(N^{\gamma}_{r,s+1})$

Tabla 8.19: Reglas del esquema de compilación ascendente-ascendente de TAG en RLPDA

[INIT]	$\$_0[\circ\circ] \longmapsto \$_0[\circ\circ]\ \nabla^{\alpha}_{0,0}[\,]$	$\alpha \in \boldsymbol{I}$
[CALL]	$\nabla^{\gamma}_{r,s}[\circ\circ] \longmapsto \nabla^{\gamma}_{r,s}[\circ\circ]\ \ \overline{N^{\gamma}_{r,s+1}}[\,]$	$N^{\gamma}_{r,s+1} \notin \text{espina}(\gamma),\ \ \mathbf{nil} \in \text{adj}(N_{r,s+1})$
[SCALL]	$\nabla^{\gamma}_{r,s}[\circ\circ] \longmapsto \nabla^{\gamma}_{r,s}[\circ\circ]\ \ \overline{N^{\gamma}_{r,s+1}}[\,]$	$N^{\gamma}_{r,s+1} \in \text{espina}(\gamma),\ \ \mathbf{nil} \in \text{adj}(N_{r,s+1})$
[SEL]	$\overline{N^{\gamma}_{r,0}}[\circ\circ] \longmapsto \nabla^{\gamma}_{r,0}[\circ\circ]$	$r \neq 0$
[PUB]	$\nabla^{\gamma}_{r,n_r}[\circ\circ] \longmapsto \overline{\overline{N^{\gamma}_{r,0}}}[\circ\circ]$	
[RET]	$\nabla^{\gamma}_{r,s}[\circ\circ]\ \ \overline{\overline{N^{\gamma}_{r,s+1}}}[\,] \longmapsto \nabla^{\gamma}_{r,s+1}[\circ\circ]$	$N^{\gamma}_{r,s+1} \notin \text{espina}(\gamma),\ \ \mathbf{nil} \in \text{adj}(N_{r,s+1})$
[SRET]	$\nabla^{\gamma}_{r,s}[\,]\ \ \overline{\overline{N^{\gamma}_{r,s+1}}}[\circ\circ] \longmapsto \nabla^{\gamma}_{r,s+1}[\circ\circ]$	$N^{\gamma}_{r,s+1} \in \text{espina}(\gamma),\ \ \mathbf{nil} \in \text{adj}(N_{r,s+1})$
[SCAN]	$\overline{N^{\gamma}_{r,0}}[\circ\circ] \overset{a}{\longmapsto}\ \overline{\overline{N^{\gamma}_{r,0}}}[\circ\circ]$	$N^{\gamma}_{r,0}[\,] \rightarrow a$
[ACALL]	$\nabla^{\gamma}_{r,s}[\circ\circ] \longmapsto \nabla^{\gamma}_{r,s}[\circ\circ]\ \ \overline{\top^{\beta}}[\,]$	$\beta \in \text{adj}(N^{\gamma}_{r,s+1})$
[ARET]	$\nabla^{\gamma}_{r,s}[\circ\circ]\ \ \overline{\overline{\top^{\beta}}}[\circ\circ N^{\gamma}_{r,s+1}] \longmapsto \nabla^{\gamma}_{r,s+1}[\circ\circ]$	$\beta \in \text{adj}(N^{\gamma}_{r,s+1})$
[FCALL]	$\nabla^{\beta}_{f,0}[\circ\circ] \longmapsto \nabla^{\beta}_{f,0}[\circ\circ]\ \ \overline{N^{\gamma}_{r,s+1}}[\,]$	$N^{\beta}_{f,0} = \mathbf{F}^{\beta},\ \beta \in \text{adj}(N^{\gamma}_{r,s+1})$
[FRET]	$\nabla^{\beta}_{f,0}[\,]\ \ \overline{\overline{N^{\gamma}_{r,s+1}}}[\circ\circ] \longmapsto \nabla^{\beta}_{f,1}[\circ\circ N^{\gamma}_{r,s+1}]$	$N^{\beta}_{f,0} = \mathbf{F}^{\beta},\ \beta \in \text{adj}(N^{\gamma}_{r,s+1})$

Tabla 8.20: Reglas del esquema de compilación Earley-ascendente de TAG en RLPDA

[INIT]	$\$_0[\circ\circ] \longmapsto \$_0[\circ\circ]\ \nabla^{\alpha}_{0,0}[\,]$	$\alpha \in \boldsymbol{I}$
[CALL]	$\nabla^{\gamma}_{r,s}[\circ\circ] \longmapsto \nabla^{\gamma}_{r,s}[\circ\circ]\ \ N^{\gamma}_{r,s+1}[\,]$	$N^{\gamma}_{r,s+1} \not\in \text{espina}(\gamma),\ \ \mathbf{nil} \in \text{adj}(N_{r,s+1})$
[SCALL]	$\nabla^{\gamma}_{r,s}[\circ\circ] \longmapsto \nabla^{\gamma}_{r,s}[\circ\circ]\ \ N^{\gamma}_{r,s+1}[\,]$	$N^{\gamma}_{r,s+1} \in \text{espina}(\gamma),\ \ \mathbf{nil} \in \text{adj}(N_{r,s+1})$
[SEL]	$N^{\gamma}_{r,0}[\circ\circ] \longmapsto \nabla^{\gamma}_{r,0}[\circ\circ]$	$r \neq 0$
[PUB]	$\nabla^{\gamma}_{r,n_r}[\circ\circ] \longmapsto \Box[\circ\circ]$	
[RET]	$\nabla^{\gamma}_{r,s}[\circ\circ]\ \ \Box[\,] \longmapsto \nabla^{\gamma}_{r,s+1}[\circ\circ]$	$N^{\gamma}_{r,s+1} \not\in \text{espina}(\gamma),\ \ \mathbf{nil} \in \text{adj}(N_{r,s+1})$
[SRET]	$\nabla^{\gamma}_{r,s}[\,]\ \ \Box[\circ\circ] \longmapsto \nabla^{\gamma}_{r,s+1}[\circ\circ]$	$N^{\gamma}_{r,s+1} \in \text{espina}(\gamma),\ \ \mathbf{nil} \in \text{adj}(N_{r,s+1})$
[SCAN]	$N^{\gamma}_{r,0}[\circ\circ] \overset{a}{\longmapsto}\ \Box[\circ\circ]$	$N^{\gamma}_{r,0}[\,] \to a$
[ACALL]	$\nabla^{\gamma}_{r,s}[\circ\circ] \longmapsto \nabla^{\gamma}_{r,s}[\circ\circ]\ \ \top^{\beta}[\,]$	$\beta \in \text{adj}(N^{\gamma}_{r,s+1})$
[ARET]	$\nabla^{\gamma}_{r,s}[\circ\circ]\ \ \Box[\circ\circ N^{\gamma}_{r,s+1}] \longmapsto \nabla^{\gamma}_{r,s+1}[\circ\circ]$	$\beta \in \text{adj}(N^{\gamma}_{r,s+1})$
[FCALL]	$\nabla^{\beta}_{f,0}[\circ\circ] \longmapsto \nabla^{\beta}_{f,0}[\circ\circ]\ \ N^{\gamma}_{r,s+1}[\,]$	$N^{\beta}_{f,0} = \mathbf{F}^{\beta},\ \beta \in \text{adj}(N^{\gamma}_{r,s+1})$
[FRET]	$\nabla^{\beta}_{f,0}[\,]\ \ \Box[\circ\circ] \longmapsto \nabla^{\beta}_{f,1}[\circ\circ N^{\gamma}_{r,s+1}]$	$N^{\beta}_{f,0} = \mathbf{F}^{\beta},\ \beta \in \text{adj}(N^{\gamma}_{r,s+1})$

Tabla 8.21: Reglas del esquema de compilación descendente-ascendente de TAG en RLPDA

[INIT]	$\$_0[\circ\circ] \longmapsto \$_0[\circ\circ]\ \nabla^{\alpha}_{0,0}[\,]$	$\alpha \in \boldsymbol{I}$
[CALL]	$\nabla^{\gamma}_{r,s}[\circ\circ] \longmapsto \nabla^{\gamma}_{r,s}[\circ\circ]\ \ \Box[\,]$	$N^{\gamma}_{r,s+1} \not\in \text{espina}(\gamma),\ \ \mathbf{nil} \in \text{adj}(N_{r,s+1})$
[SCALL]	$\nabla^{\gamma}_{r,s}[\circ\circ] \longmapsto \nabla^{\gamma}_{r,s}[\circ\circ]\ \ \Box[\circ\circ]$	$N^{\gamma}_{r,s+1} \in \text{espina}(\gamma),\ \ \mathbf{nil} \in \text{adj}(N_{r,s+1})$
[SEL]	$\Box[\circ\circ] \longmapsto \nabla^{\gamma}_{r,0}[\circ\circ]$	$r \neq 0$
[PUB]	$\nabla^{\gamma}_{r,n_r}[\circ\circ] \longmapsto N^{\gamma}_{r,0}[\circ\circ]$	
[RET]	$\nabla^{\gamma}_{r,s}[\circ\circ]\ \ N^{\gamma}_{r,s+1}[\,] \longmapsto \nabla^{\gamma}_{r,s+1}[\circ\circ]$	$N^{\gamma}_{r,s+1} \not\in \text{espina}(\gamma),\ \ \mathbf{nil} \in \text{adj}(N_{r,s+1})$
[SRET]	$\nabla^{\gamma}_{r,s}[\circ\circ]\ \ N^{\gamma}_{r,s+1}[\circ\circ] \longmapsto \nabla^{\gamma}_{r,s+1}[\circ\circ]$	$N^{\gamma}_{r,s+1} \in \text{espina}(\gamma),\ \ \mathbf{nil} \in \text{adj}(N_{r,s+1})$
[SCAN]	$\Box[\,] \overset{a}{\longmapsto}\ N^{\gamma}_{r,0}[\,]$	$N^{\gamma}_{r,0}[\,] \to a$
[ACALL]	$\nabla^{\gamma}_{r,s}[\circ\circ] \longmapsto \nabla^{\gamma}_{r,s}[\circ\circ]\ \ \Box[\circ\circ N^{\gamma}_{r,s+1}]$	$\beta \in \text{adj}(N^{\gamma}_{r,s+1})$
[ARET]	$\nabla^{\gamma}_{r,s}[\circ\circ]\ \ \top^{\beta}[\circ\circ N^{\gamma}_{r,s+1}] \longmapsto \nabla^{\gamma}_{r,s+1}[\circ\circ]$	$\beta \in \text{adj}(N^{\gamma}_{r,s+1})$
[FCALL]	$\nabla^{\beta}_{f,0}[\circ\circ N^{\gamma}_{r,s+1}] \longmapsto \nabla^{\beta}_{f,0}[\circ\circ N^{\gamma}_{r,s+1}]\ \ \Box[\circ\circ]$	$N^{\beta}_{f,0} = \mathbf{F}^{\beta},\ \beta \in \text{adj}(N^{\gamma}_{r,s+1})$
[FRET]	$\nabla^{\beta}_{f,0}[\circ\circ N^{\gamma}_{r,s+1}]\ \ N^{\gamma}_{r,s+1}[\circ\circ] \longmapsto \nabla^{\beta}_{f,1}[\circ\circ N^{\gamma}_{r,s+1}]$	$N^{\beta}_{f,0} = \mathbf{F}^{\beta},\ \beta \in \text{adj}(N^{\gamma}_{r,s+1})$

Tabla 8.22: Reglas del esquema de compilación ascendente-Earley de TAG en RLPDA

[INIT]	$\$_0[\circ\circ] \longmapsto \$_0[\circ\circ]\ \nabla^{\alpha}_{0,0}[\,]$	$\alpha \in \boldsymbol{I}$
[CALL]	$\nabla^{\gamma}_{r,s}[\circ\circ] \longmapsto \nabla^{\gamma}_{r,s}[\circ\circ]\ \ \overline{N^{\gamma}_{r,s+1}}[\,]$	$N^{\gamma}_{r,s+1} \notin \text{espina}(\gamma),\ \mathbf{nil} \in \text{adj}(N_{r,s+1})$
[SCALL]	$\nabla^{\gamma}_{r,s}[\circ\circ] \longmapsto \nabla^{\gamma}_{r,s}[\circ\circ]\ \ \overline{N^{\gamma}_{r,s+1}}[\circ\circ]$	$N^{\gamma}_{r,s+1} \in \text{espina}(\gamma),\ \mathbf{nil} \in \text{adj}(N_{r,s+1})$
[SEL]	$\overline{N^{\gamma}_{r,0}}[\circ\circ] \longmapsto \nabla^{\gamma}_{r,0}[\circ\circ]$	$r \neq 0$
[PUB]	$\nabla^{\gamma}_{r,n_r}[\circ\circ] \longmapsto \overline{\overline{\overline{N^{\gamma}_{r,0}}}}[\circ\circ]$	
[RET]	$\nabla^{\gamma}_{r,s}[\circ\circ]\ \ \overline{\overline{\overline{N^{\gamma}_{r,s+1}}}}[\,] \longmapsto \nabla^{\gamma}_{r,s+1}[\circ\circ]$	$N^{\gamma}_{r,s+1} \notin \text{espina}(\gamma),\ \mathbf{nil} \in \text{adj}(N_{r,s+1})$
[SRET]	$\nabla^{\gamma}_{r,s}[\circ\circ]\ \ \overline{\overline{\overline{N^{\gamma}_{r,s+1}}}}[\circ\circ] \longmapsto \nabla^{\gamma}_{r,s+1}[\circ\circ]$	$N^{\gamma}_{r,s+1} \in \text{espina}(\gamma),\ \mathbf{nil} \in \text{adj}(N_{r,s+1})$
[SCAN]	$\overline{N^{\gamma}_{r,0}}[\,] \overset{a}{\longmapsto} \overline{\overline{\overline{N^{\gamma}_{r,0}}}}[\,]$	$N^{\gamma}_{r,0}[\,] \to a$
[ACALL]	$\nabla^{\gamma}_{r,s}[\circ\circ] \longmapsto \nabla^{\gamma}_{r,s}[\circ\circ]\ \ \overline{\top^{\beta}}[\circ\circ N^{\gamma}_{r,s+1}]$	$\beta \in \text{adj}(N^{\gamma}_{r,s+1})$
[ARET]	$\nabla^{\gamma}_{r,s}[\circ\circ]\ \ \overline{\overline{\overline{\top^{\beta}}}}[\circ\circ N^{\gamma}_{r,s+1}] \longmapsto \nabla^{\gamma}_{r,s+1}[\circ\circ]$	$\beta \in \text{adj}(N^{\gamma}_{r,s+1})$
[FCALL]	$\nabla^{\beta}_{f,0}[\circ\circ N^{\gamma}_{r,s+1}] \longmapsto \nabla^{\beta}_{f,0}[\circ\circ N^{\gamma}_{r,s+1}]\ \ \overline{N^{\gamma}_{r,s+1}}[\circ\circ]$	$N^{\beta}_{f,0} = \mathbf{F}^{\beta},\ \beta \in \text{adj}(N^{\gamma}_{r,s+1})$
[FRET]	$\nabla^{\beta}_{f,0}[\circ\circ N^{\gamma}_{r,s+1}]\ \ \overline{\overline{\overline{N^{\gamma}_{r,s+1}}}}[\circ\circ] \longmapsto \nabla^{\beta}_{f,1}[\circ\circ N^{\gamma}_{r,s+1}]$	$N^{\beta}_{f,0} = \mathbf{F}^{\beta},\ \beta \in \text{adj}(N^{\gamma}_{r,s+1})$

Tabla 8.23: Reglas del esquema de compilación Earley-Earley de TAG en RLPDA

[INIT]	$\$_0[\circ\circ] \longmapsto \$_0[\circ\circ]\ \nabla^{\alpha}_{0,0}[\,]$	$\alpha \in \boldsymbol{I}$
[CALL]	$\nabla^{\gamma}_{r,s}[\circ\circ] \longmapsto \nabla^{\gamma}_{r,s}[\circ\circ]\ \ N^{\gamma}_{r,s+1}[\,]$	$N^{\gamma}_{r,s+1} \notin \text{espina}(\gamma),\ \mathbf{nil} \in \text{adj}(N_{r,s+1})$
[SCALL]	$\nabla^{\gamma}_{r,s}[\circ\circ] \longmapsto \nabla^{\gamma}_{r,s}[\circ\circ]\ \ N^{\gamma}_{r,s+1}[\circ\circ]$	$N^{\gamma}_{r,s+1} \in \text{espina}(\gamma),\ \mathbf{nil} \in \text{adj}(N_{r,s+1})$
[SEL]	$N^{\gamma}_{r,0}[\circ\circ] \longmapsto \nabla^{\gamma}_{r,0}[\circ\circ]$	$r \neq 0$
[PUB]	$\nabla^{\gamma}_{r,n_r}[\circ\circ] \longmapsto \Box[\circ\circ]$	
[RET]	$\nabla^{\gamma}_{r,s}[\circ\circ]\ \ \Box[\,] \longmapsto \nabla^{\gamma}_{r,s+1}[\circ\circ]$	$N^{\gamma}_{r,s+1} \notin \text{espina}(\gamma),\ \mathbf{nil} \in \text{adj}(N_{r,s+1})$
[SRET]	$\nabla^{\gamma}_{r,s}[\circ\circ]\ \ \Box[\circ\circ] \longmapsto \nabla^{\gamma}_{r,s+1}[\circ\circ]$	$N^{\gamma}_{r,s+1} \in \text{espina}(\gamma),\ \mathbf{nil} \in \text{adj}(N_{r,s+1})$
[SCAN]	$N^{\gamma}_{r,0}[\,] \overset{a}{\longmapsto} \Box[\,]$	$N^{\gamma}_{r,0}[\,] \to a$
[ACALL]	$\nabla^{\gamma}_{r,s}[\circ\circ] \longmapsto \nabla^{\gamma}_{r,s}[\circ\circ]\ \ \top^{\beta}[\circ\circ N^{\gamma}_{r,s+1}]$	$\beta \in \text{adj}(N^{\gamma}_{r,s+1})$
[ARET]	$\nabla^{\gamma}_{r,s}[\circ\circ]\ \ \Box[\circ\circ N^{\gamma}_{r,s+1}] \longmapsto \nabla^{\gamma}_{r,s+1}[\circ\circ]$	$\beta \in \text{adj}(N^{\gamma}_{r,s+1})$
[FCALL]	$\nabla^{\beta}_{f,0}[\circ\circ N^{\gamma}_{r,s+1}] \longmapsto \nabla^{\beta}_{f,0}[\circ\circ N^{\gamma}_{r,s+1}]\ \ N^{\gamma}_{r,s+1}[\circ\circ]$	$N^{\beta}_{f,0} = \mathbf{F}^{\beta},\ \beta \in \text{adj}(N^{\gamma}_{r,s+1})$
[FRET]	$\nabla^{\beta}_{f,0}[\circ\circ N^{\gamma}_{r,s+1}]\ \ \Box[\circ\circ] \longmapsto \nabla^{\beta}_{f,1}[\circ\circ N^{\gamma}_{r,s+1}]$	$N^{\beta}_{f,0} = \mathbf{F}^{\beta},\ \beta \in \text{adj}(N^{\gamma}_{r,s+1})$

Tabla 8.24: Reglas del esquema de compilación descendente-Earley de TAG en RLPDA

8.4.4 Estrategias $*$-descendentes

Las estrategias $*$-descendentes se caracterizan porque cuando se predice una adjunción se almacena información acerca del nodo de adjunción. Dicha información es transmitida por la espina del árbol auxiliar adjuntado, de tal modo que al llegar al nodo pie puede ser utilizada para determinar con seguridad el subárbol que debe colgar de dicho nodo pie. En cambio, una vez que se termina el análisis de dicho subárbol y se retorna al nodo pie, no se almacena información alguna en referencia al nodo de adjunción.

Estrategia ascendente-descendente

Esquema de compilación 8.19 El esquema de compilación ascendente-descendente de una gramática de adjunción de árboles en un autómata lógico a pila restringido queda definido por el conjunto de reglas mostrado en la tabla 8.25 y por los elementos inicial $\$_0[\,]$ y final $\top^\alpha[\,]$, con $\alpha \in \boldsymbol{I}$. Con respecto al esquema de compilación genérico, la regla de compilación **[SCAN]** ha sido modificada aunque dicha modificación no afecta a su comportamiento puesto que por construcción del esquema de compilación, la pila de índices asociada a cada nodo $\overleftarrow{N^\gamma}$ es siempre $[\,]$. §

Estrategia Earley-descendente

Esquema de compilación 8.20 El esquema de compilación Earley-descendente de una gramática de adjunción de árboles en un autómata lógico a pila restringido queda definido por el conjunto de reglas mostrado en la tabla 8.26 y por los elementos inicial $\$_0[\,]$ y final $\overline{\overline{\top^\alpha}}[\,]$, con $\alpha \in \boldsymbol{I}$. §

Estrategia descendente-descendente

Esquema de compilación 8.21 El esquema de compilación descendente-descendente de una gramática de adjunción de árboles en un autómata lógico a pila restringido queda definido por el conjunto de reglas mostrado en la tabla 8.27 y por los elementos inicial $\$_0[\,]$ y final $\Box[\,]$.

§

8.5 Tabulación de los autómatas lógicos a pila restringidos

Cualquier esquema de compilación de DCG en LPDA se convierte automáticamente en un esquema de compilación para LIG cuando se restringe su aplicación a este tipo de gramáticas y, por tanto, es posible aplicar la técnica general de tabulación de los autómatas lógicos a pila al caso de los RLPDA. Efectivamente, en un autómata lógico a pila restringido tenemos que dada una derivación

$$(B[\alpha], a_{i+1} \dots a_n) \overset{*}{\vdash}_d (B[\alpha]\ C[\beta], a_{j+1} \dots a_n)$$

se cumple

$$(\xi\ B[\alpha], a_{i+1} \dots a_n) \overset{*}{\vdash}_d (\xi\ B[\alpha]\ C[\beta], a_{j+1} \dots a_n)$$

[INIT]	$\$_0[\circ\circ] \longmapsto \$_0[\circ\circ]\ \nabla^{\alpha}_{0,0}[\,]$	$\alpha \in \boldsymbol{I}$
[CALL]	$\nabla^{\gamma}_{r,s}[\circ\circ] \longmapsto \nabla^{\gamma}_{r,s}[\circ\circ]\ \Box[\,]$	$N^{\gamma}_{r,s+1} \not\in \text{espina}(\gamma),\ \mathbf{nil} \in \text{adj}(N_{r,s+1})$
[SCALL]	$\nabla^{\gamma}_{r,s}[\circ\circ] \longmapsto \nabla^{\gamma}_{r,s}[\circ\circ]\ \Box[\circ\circ]$	$N^{\gamma}_{r,s+1} \in \text{espina}(\gamma),\ \mathbf{nil} \in \text{adj}(N_{r,s+1})$
[SEL]	$\Box[\circ\circ] \longmapsto \nabla^{\gamma}_{r,0}[\circ\circ]$	$r \neq 0$
[PUB]	$\nabla^{\gamma}_{r,n_r}[\circ\circ] \longmapsto N^{\gamma}_{r,0}[\circ\circ]$	
[RET]	$\nabla^{\gamma}_{r,s}[\circ\circ]\ \ N^{\gamma}_{r,s+1}[\,] \longmapsto \nabla^{\gamma}_{r,s+1}[\circ\circ]$	$N^{\gamma}_{r,s+1} \not\in \text{espina}(\gamma),\ \mathbf{nil} \in \text{adj}(N_{r,s+1})$
[SRET]	$\nabla^{\gamma}_{r,s}[\circ\circ]\ \ N^{\gamma}_{r,s+1}[\,] \longmapsto \nabla^{\gamma}_{r,s+1}[\,]$	$N^{\gamma}_{r,s+1} \in \text{espina}(\gamma),\ \mathbf{nil} \in \text{adj}(N_{r,s+1})$
[SCAN]	$\Box[\circ\circ] \overset{a}{\longmapsto} N^{\gamma}_{r,0}[\circ\circ]$	$N^{\gamma}_{r,0}[\,] \rightarrow a$
[ACALL]	$\nabla^{\gamma}_{r,s}[\circ\circ] \longmapsto \nabla^{\gamma}_{r,s}[\circ\circ]\ \Box[\circ\circ N^{\gamma}_{r,s+1}]$	$\beta \in \text{adj}(N^{\gamma}_{r,s+1})$
[ARET]	$\nabla^{\gamma}_{r,s}[\circ\circ]\ \ \top^{\beta}[\,] \longmapsto \nabla^{\gamma}_{r,s+1}[\,]$	$\beta \in \text{adj}(N^{\gamma}_{r,s+1})$
[FCALL]	$\nabla^{\beta}_{f,0}[\circ\circ N^{\gamma}_{r,s+1}] \longmapsto \nabla^{\beta}_{f,0}[\circ\circ N^{\gamma}_{r,s+1}]\ \Box[\circ\circ]$	$N^{\beta}_{f,0} = \mathbf{F}^{\beta},\ \beta \in \text{adj}(N^{\gamma}_{r,s+1})$
[FRET]	$\nabla^{\beta}_{f,0}[\circ\circ N^{\gamma}_{r,s+1}]\ \ N^{\gamma}_{r,s+1}[\,] \longmapsto \nabla^{\beta}_{f,1}[\,]$	$N^{\beta}_{f,0} = \mathbf{F}^{\beta},\ \beta \in \text{adj}(N^{\gamma}_{r,s+1})$

Tabla 8.25: Reglas del esquema de compilación ascendente-descendente de TAG en RLPDA

[INIT]	$\$_0[\circ\circ] \longmapsto \$_0[\circ\circ]\ \nabla^{\alpha}_{0,0}[\,]$	$\alpha \in \boldsymbol{I}$
[CALL]	$\nabla^{\gamma}_{r,s}[\circ\circ] \longmapsto \nabla^{\gamma}_{r,s}[\circ\circ]\ \overline{N^{\gamma}_{r,s+1}}[\,]$	$N^{\gamma}_{r,s+1} \not\in \text{espina}(\gamma),\ \mathbf{nil} \in \text{adj}(N_{r,s+1})$
[SCALL]	$\nabla^{\gamma}_{r,s}[\circ\circ] \longmapsto \nabla^{\gamma}_{r,s}[\circ\circ]\ \overline{N^{\gamma}_{r,s+1}}[\circ\circ]$	$N^{\gamma}_{r,s+1} \in \text{espina}(\gamma),\ \mathbf{nil} \in \text{adj}(N_{r,s+1})$
[SEL]	$\overline{N^{\gamma}_{r,0}}[\circ\circ] \longmapsto \nabla^{\gamma}_{r,0}[\circ\circ]$	$r \neq 0$
[PUB]	$\nabla^{\gamma}_{r,n_r}[\circ\circ] \longmapsto \overline{\overline{N^{\gamma}_{r,0}}}[\circ\circ]$	
[RET]	$\nabla^{\gamma}_{r,s}[\circ\circ]\ \ \overline{\overline{N^{\gamma}_{r,s+1}}}[\,] \longmapsto \nabla^{\gamma}_{r,s+1}[\circ\circ]$	$N^{\gamma}_{r,s+1} \not\in \text{espina}(\gamma),\ \mathbf{nil} \in \text{adj}(N_{r,s+1})$
[SRET]	$\nabla^{\gamma}_{r,s}[\circ\circ]\ \ \overline{\overline{N^{\gamma}_{r,s+1}}}[\,] \longmapsto \nabla^{\gamma}_{r,s+1}[\,]$	$N^{\gamma}_{r,s+1} \in \text{espina}(\gamma),\ \mathbf{nil} \in \text{adj}(N_{r,s+1})$
[SCAN]	$\overline{N^{\gamma}_{r,0}}[\,] \overset{a}{\longmapsto} \overline{\overline{N^{\gamma}_{r,0}}}[\,]$	$N^{\gamma}_{r,0}[\,] \rightarrow a$
[ACALL]	$\nabla^{\gamma}_{r,s}[\circ\circ] \longmapsto \nabla^{\gamma}_{r,s}[\circ\circ]\ \overline{\top^{\beta}}[\circ\circ N^{\gamma}_{r,s+1}]$	$\beta \in \text{adj}(N^{\gamma}_{r,s+1})$
[ARET]	$\nabla^{\gamma}_{r,s}[\circ\circ]\ \ \overline{\overline{\top^{\beta}}}[\,] \longmapsto \nabla^{\gamma}_{r,s+1}[\,]$	$\beta \in \text{adj}(N^{\gamma}_{r,s+1})$
[FCALL]	$\nabla^{\beta}_{f,0}[\circ\circ N^{\gamma}_{r,s+1}] \longmapsto \nabla^{\beta}_{f,0}[\circ\circ N^{\gamma}_{r,s+1}]\ \overline{N^{\gamma}_{r,s+1}}[\circ\circ]$	$N^{\beta}_{f,0} = \mathbf{F}^{\beta},\ \beta \in \text{adj}(N^{\gamma}_{r,s+1})$
[FRET]	$\nabla^{\beta}_{f,0}[\circ\circ N^{\gamma}_{r,s+1}]\ \ \overline{\overline{N^{\gamma}_{r,s+1}}}[\,] \longmapsto \nabla^{\beta}_{f,1}[\,]$	$N^{\beta}_{f,0} = \mathbf{F}^{\beta},\ \beta \in \text{adj}(N^{\gamma}_{r,s+1})$

Tabla 8.26: Reglas del esquema de compilación Earley-descendente de TAG en RLPDA

[INIT]	$\$_0[\circ\circ] \longmapsto \$_0[\circ\circ]\ \nabla^{\alpha}_{0,0}[\,]$	$\alpha \in \boldsymbol{I}$
[CALL]	$\nabla^{\gamma}_{r,s}[\circ\circ] \longmapsto \nabla^{\gamma}_{r,s}[\circ\circ]\ \ N^{\gamma}_{r,s+1}[\,]$	$N^{\gamma}_{r,s+1} \not\in \text{espina}(\gamma),\ \ \mathbf{nil} \in \text{adj}(N_{r,s+1})$
[SCALL]	$\nabla^{\gamma}_{r,s}[\circ\circ] \longmapsto \nabla^{\gamma}_{r,s}[\circ\circ]\ \ N^{\gamma}_{r,s+1}[\circ\circ]$	$N^{\gamma}_{r,s+1} \in \text{espina}(\gamma),\ \ \mathbf{nil} \in \text{adj}(N_{r,s+1})$
[SEL]	$N^{\gamma}_{r,0}[\circ\circ] \longmapsto \nabla^{\gamma}_{r,0}[\circ\circ]$	$r \neq 0$
[PUB]	$\nabla^{\gamma}_{r,n_r}[\circ\circ] \longmapsto \Box[\circ\circ]$	
[RET]	$\nabla^{\gamma}_{r,s}[\circ\circ]\ \ \Box[\,] \longmapsto \nabla^{\gamma}_{r,s+1}[\circ\circ]$	$N^{\gamma}_{r,s+1} \not\in \text{espina}(\gamma),\ \ \mathbf{nil} \in \text{adj}(N_{r,s+1})$
[SRET]	$\nabla^{\gamma}_{r,s}[\circ\circ]\ \ \Box[\,] \longmapsto \nabla^{\gamma}_{r,s+1}[\,]$	$N^{\gamma}_{r,s+1} \in \text{espina}(\gamma),\ \ \mathbf{nil} \in \text{adj}(N_{r,s+1})$
[SCAN]	$N^{\gamma}_{r,0}[\,] \overset{a}{\longmapsto} \Box[\,]$	$N^{\gamma}_{r,0}[\,] \to a$
[ACALL]	$\nabla^{\gamma}_{r,s}[\circ\circ] \longmapsto \nabla^{\gamma}_{r,s}[\circ\circ]\ \ \top^{\beta}[\circ\circ N^{\gamma}_{r,s+1}]$	$\beta \in \text{adj}(N^{\gamma}_{r,s+1})$
[ARET]	$\nabla^{\gamma}_{r,s}[\circ\circ]\ \ \Box[\,] \longmapsto \nabla^{\gamma}_{r,s+1}[\,]$	$\beta \in \text{adj}(N^{\gamma}_{r,s+1})$
[FCALL]	$\nabla^{\beta}_{f,0}[\circ\circ N^{\gamma}_{r,s+1}] \longmapsto \nabla^{\beta}_{f,0}[\circ\circ N^{\gamma}_{r,s+1}]\ \ N^{\gamma}_{r,s+1}[\circ\circ]$	$N^{\beta}_{f,0} = \mathbf{F}^{\beta},\ \beta \in \text{adj}(N^{\gamma}_{r,s+1})$
[FRET]	$\nabla^{\beta}_{f,0}[\circ\circ N^{\gamma}_{r,s+1}]\ \ \Box[\,] \longmapsto \nabla^{\beta}_{f,1}[\,]$	$N^{\beta}_{f,0} = \mathbf{F}^{\beta},\ \beta \in \text{adj}(N^{\gamma}_{r,s+1})$

Tabla 8.27: Reglas del esquema de compilación descendente-descendente de TAG en RLPDA

En este caso, una derivación independiente del contexto podría ser representada de forma condensada por un ítem $[B[\alpha], i, C[\beta], j]$ que almacenase $B[\alpha]$, $C[\beta]$ y las posiciones i y j en que dichos elementos fueron situados en la cima de la pila del RLPDA.

Existe un diferencia fundamental entre LPDA y RLPDA en lo referente a la complejidad computacional: mientras que las DCG son no-decidibles en el caso general, las gramáticas lineales de índices y las gramáticas de adjunción de árboles son analizables con complejidad polinómica. Desafortunadamente, los esquemas de compilación para gramáticas de cláusulas definidas cuando son aplicados a gramáticas lineales de índices no garantizan la decidibilidad, puesto que las gramáticas lineales de índices no poseen en general las propiedades clásicas que garantizan la terminación del análisis, como la capacidad de ser analizadas en modo diferido[2] [144] o tener profundidad acotada[3] [77].

Las gramáticas lineales de índices y las gramáticas de adjunción de árboles poseen propiedades que utilizadas adecuadamente permite asegurar la decidibilidad de su análisis sintáctico, incluso en tiempo polinómico. En el caso de TAG nos referimos a la independencia del contexto de la operación de adjunción, puesto que podemos determinar si un árbol auxiliar es adjuntable o no en un nodo a partir de la información proporcionada por dicho nodo. En el caso de LIG nos referimos a la propiedad de independencia del contexto de las gramáticas lineales de índices (definición 2.1, página 32), según la cual la aplicabilidad de una producción en un momento dado viene determinada por la coincidencia del no-terminal que se trate de reducir o expandir con el lado izquierdo de dicha producción y por cierta información del contexto que viene dada por a lo sumo un elemento en la cima de la pila de índices asociada con dicho no-terminal.

Utilizando estas propiedades podemos diseñar una técnica de tabulación específica de los

[2] *off-line parseability.*
[3] *depth-boundness.*

autómatas lógicos a pila restringidos. Para ello, diferenciaremos los siguientes tipos de derivaciones:

Derivaciones de llamada. Son aquellas derivaciones producidas durante la fase de llamada o descendente de la estrategia.

Derivaciones de retorno. Son aquellas derivaciones producidas durante la fase de retorno o ascendente de la estrategia de análisis.

Derivaciones de puntos especiales. Son derivaciones que llevan a configuraciones que representan el inicio o el final de una pila de índices. En las estrategias $*$-ascendentes estas derivaciones son asimilables a las derivaciones de llamada. Sin embargo, en las estrategias $*$-Earley y $*$-descendentes ambos tipos de derivaciones deben ser diferenciados.

Para representar estos tipos de derivaciones definiremos las correspondientes clases de *ítems de llamada*, *ítems de retorno* e *ítems de puntos especiales*. En ellos se almacenará información suficiente para reconstruir la evolución de la pila del autómata. Para facilitar su manejo, consideraremos que cada ítem consta de dos partes diferenciadas:

- Una *cabeza*[4] que almacena información del estado del autómata necesaria para poder determinar si una transición es aplicable. Habitualmente incluirá los no-terminales de uno o dos elementos en la cima de la pila y el índice en la cima de la pila de índices.
- Una *cola*[5] que almacena la información que permitirá recuperar el resto de los componentes de las pilas de índices que se necesiten para verificar la congruencia de los ítems durante la aplicación de las transiciones.

Utilizaremos la notación [cabeza | cola] para representar un ítem. Dependiendo de la estrategia de análisis utilizada, diferentes esquemas de compilación precisan incorporar más o menos información en la cabeza y/o en la cola.

8.5.1 Tabulación de estrategias $*$-ascendentes

En la familia de estrategias $*$-ascendentes la información correspondiente a los no-terminales puede predecirse y/o propagarse sin limitación alguna, pero la información correspondiente a las pilas de índices sólo se propaga en la fase de retorno, prohibiéndose su predicción en la fase de llamada.

En la tabla 8.28 se muestran las transiciones obtenidas de los esquemas de compilación de LIG y TAG que incorporan estrategias $*$-ascendentes. Debemos reseñar que en dicha tabla hemos generalizado a $C[\circ\circ\gamma] \overset{a}{\longmapsto} F[\circ\circ\gamma']$ las transiciones $C[\circ\circ] \overset{a}{\longmapsto} F[\circ\circ]$ utilizadas por las reglas de compilación **[SEL]**, **[PUB]** y **[SCAN]**. También hemos generalizado a $C[\circ\circ\gamma]\ F[\,] \longmapsto G[\circ\circ\gamma']$ las transiciones $C[\circ\circ]\ F[\ ] \longmapsto G[\circ\circ]$ que utiliza la regla de compilación **[RET]**. Por último, hemos permitido que cualquier transición pueda avanzar en el reconocimiento de la cadena de entrada. Los nuevos tipos de transiciones introducidos con respecto a los que aparecen en los esquemas de compilación antes citados pueden ser de utilidad en la definición de esquemas de compilación para otras estrategias de análisis, por ejemplo aquellas de tipo desplazamiento-reducción[6] y no afectan a la forma de las derivaciones, por lo que no añaden complicaciones adicionales a la técnica de tabulación.

[4] *head*.

[5] *rest*.

[6] En la sección C.8 se muestra un esquema de compilación de tipo LR para LIG que necesita transiciones del tipo $C[\circ\circ\gamma]\ F[\,] \longmapsto G[\circ\circ\gamma']$.

Transición	**Compilación de LIG**	**Compilación de TAG**
$C[\circ\circ] \overset{a}{\longmapsto} C[\circ\circ]\ F[\,]$	**[INIT][CALL][SCALL]**	**[INIT][CALL][SCALL][ACALL][FCALL]**
$C[\circ\circ\gamma] \overset{a}{\longmapsto} F[\circ\circ\gamma']$	**[SEL][PUB][SCAN]**	**[SEL][PUB][SCAN]**
$C[\circ\circ\gamma]\ F[\,] \overset{a}{\longmapsto} G[\circ\circ\gamma']$	**[RET]**	**[RET]**
$C[\,]\ F[\circ\circ\gamma] \overset{a}{\longmapsto} G[\circ\circ\gamma']$	**[SRET]**	**[SRET][ARET][FRET]**

Tabla 8.28: Tipos de transiciones en las estrategias $*$-ascendentes

Teorema 8.1 *Los autómatas lógicos a pila restringidos que utilizan el juego de transiciones de la tabla 8.28 aceptan la clase de los lenguajes de adjunción de árboles.*

Demostración:

Por los esquemas de compilación de LIG y TAG que incorporan estrategias $*$-ascendentes sabemos que los lenguajes de adjunciones de árboles son aceptados por los autómatas lógicos a pila restringidos que utilizan las instrucciones de la tabla 8.28.

Para mostrar que todo lenguaje aceptado por un RLPDA que utilice las transiciones de la tabla 8.28 es un lenguaje de adjunción de árboles, definiremos un procedimiento para crear una gramática lineal de índices a partir de tales autómatas.

Sea $\mathcal{A} = (V_T, P, V_I, \mathcal{X}, \$_0, \$_f, \Theta)$ un autómata lógico a pila restringido. Construiremos una gramática lineal de índices $\mathcal{L} = (V_T, V_N, V_I, S, P)$, donde el conjunto V_N de no-terminales estará formado por pares $\langle E, B\rangle$ tal que $A, B \in P$. Para que $\mathcal{L}$ reconozca el lenguaje aceptado por $\mathcal{A}$ el conjunto de producciones en P ha de construirse a partir de las transiciones en Θ de la siguiente manera:

- Para toda transición $C[\circ\circ] \overset{a}{\longmapsto} C[\circ\circ]\ F[\,]$ creamos una producción

$$\langle C, F\rangle[\,] \rightarrow a$$

- Para toda transición $C[\circ\circ] \overset{a}{\longmapsto} F[\circ\circ]$ y para todo $E \in P$ creamos una producción

$$\langle E, F\rangle[\circ\circ] \rightarrow \langle E, C\rangle[\circ\circ]\ a$$

- Para toda transición $C[\circ\circ] \overset{a}{\longmapsto} F[\circ\circ\gamma']$ y para todo $E \in P$ creamos una producción

$$\langle E, F\rangle[\circ\circ\gamma'] \rightarrow \langle E, C\rangle[\circ\circ]\ a$$

- Para toda transición $C[\circ\circ\gamma] \overset{a}{\longmapsto} F[\circ\circ]$ y para todo $E \in P$ creamos una producción

$$\langle E, F\rangle[\circ\circ] \rightarrow \langle E, C\rangle[\circ\circ\gamma]\ a$$

- Para toda transición $C[\circ\circ]\ F[\,] \overset{a}{\longmapsto} G[\circ\circ]$ y para todo $E \in P$ creamos una producción

$$\langle E, G\rangle[\circ\circ] \rightarrow \langle E, C\rangle[\circ\circ]\ \langle C, F\rangle[\,]\ a$$

- Para toda transición $C[\circ\circ]\ F[\,] \overset{a}{\longmapsto} G[\circ\circ\gamma']$ y para todo $E \in P$ creamos una producción

$$\langle E, G\rangle[\circ\circ\gamma'] \rightarrow \langle E, C\rangle[\circ\circ]\ \langle C, F\rangle[\,]\ a$$

- Para toda transición $C[\circ\circ\gamma]\ F[\,] \overset{a}{\longmapsto} G[\circ\circ]$ y para todo $E \in P$ creamos una producción

$$\langle E, G\rangle[\circ\circ] \rightarrow \langle E, C\rangle[\circ\circ\gamma]\ \langle C, F\rangle[\,]\ a$$

- Para toda transición $C[\,]\ F[\circ\circ] \stackrel{a}{\longmapsto} G[\circ\circ]$ y para todo $E \in P$ creamos una producción

$$\langle E, G\rangle[\circ\circ] \rightarrow \langle E, C\rangle[\,]\ \langle C, F\rangle[\circ\circ]\ a$$

- Para toda transición $C[\,]\ F[\circ\circ] \stackrel{a}{\longmapsto} G[\circ\circ\gamma']$ y para todo $E \in P$ creamos una producción

$$\langle E, G\rangle[\circ\circ\gamma'] \rightarrow \langle E, C\rangle[\,]\ \langle C, F\rangle[\circ\circ]\ a$$

- Para toda transición $C[\,]\ F[\circ\circ\gamma] \stackrel{a}{\longmapsto} G[\circ\circ]$ y para todo $E \in P$ creamos una producción

$$\langle E, G\rangle[\circ\circ] \rightarrow \langle E, C\rangle[\,]\ \langle C, F\rangle[\circ\circ\gamma]\ a$$

Con respecto al axioma de la gramática, tenemos que $S = \langle \$_0, \$_f\rangle$.

Mediante inducción en la longitud de las derivaciones, es posible mostrar que $\langle E, B\rangle[\alpha] \stackrel{*}{\Rightarrow} w$ si y sólo si $(E, w) \stackrel{*}{\vdash} (E\ B[\alpha], \epsilon)$. Esto es así puesto que:

- Si una derivación $(E, w) \stackrel{*}{\vdash} (E\ B[\alpha], \epsilon)$ es el resultado de aplicar la secuencia $t_1, \ldots, t_m$ de transiciones en Θ, entonces existe una secuencia $p_1, \ldots, p_m$ de producciones en P tal que p_i es una producción creada a partir de t_i y la derivación derecha $\langle E, B\rangle[\alpha] \stackrel{*}{\Rightarrow} w$ resultado de aplicar $p_m, \ldots, p_1$ reconoce w.
- Si una derivación derecha $\langle E, B\rangle[\alpha] \stackrel{*}{\Rightarrow} w$ reconoce la cadena w como resultado de aplicar la secuencia $p_1, \ldots, p_m$ de producciones en P, entonces existe una secuencia de transiciones $t_1, \ldots, t_m$ tal que la p_i es una producción creada a partir de t_i y la derivación $(E, w) \stackrel{*}{\vdash} (E\ B[\alpha], \epsilon)$ es el resultado de aplicar la secuencia de transiciones $t_m, \ldots, t_1$.

□

Toda configuración $(\xi\ B[\alpha]\ C[\beta\gamma], a_{j+1} \ldots a_n)$ de un autómata a pila obtenida a partir de las transiciones mostradas en la tabla 8.28, puede ser clasificada en uno de los dos tipos de derivaciones que se definen a continuación:

Derivaciones de llamada. Corresponden a configuraciones en las que $\beta\gamma = \epsilon$ e implican la existencia de la siguiente derivación:

$$(\xi\ B[\alpha], a_{i+1} \ldots a_n) \stackrel{*}{\vdash} (\xi\ B[\alpha]\ C[\,], a_{j+1} \ldots a_n)$$

donde en toda la derivación no se ha modificado $\xi\ B[\alpha]$ y a lo sumo se ha consultado B. En la figura 8.1 se muestra una representación gráfica de este tipo de transiciones.

Para cualquier $\xi' \in (P[V_I^*])^*$ y $\alpha' \in V_I^*$ se cumple

$$(\xi'\ B[\alpha'], a_{i+1} \ldots a_n) \stackrel{*}{\vdash} (\xi'\ B[\alpha']\ C[\,], a_{j+1} \ldots a_n)$$

por lo que podemos representar este tipo de derivaciones mediante ítems de la forma

$$[B, i, C, j, - \mid -, -, -, -]$$

En los autómatas lógicos a pila restringidos que hacen uso únicamente de las transiciones mostradas en la tabla 8.28, las derivaciones de puntos especiales no son distinguibles de las derivaciones de llamada.

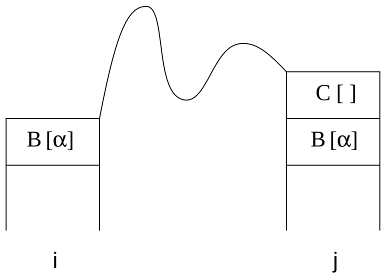

Figura 8.1: Derivaciones de llamada en estrategias $*$-ascendentes

Derivaciones de retorno. Corresponden a configuraciones en la que $\beta \in V_I^*$ y $\gamma \in V_I$ e implican la existencia de la siguiente secuencia de subderivaciones:

$$\begin{aligned}(\xi\ B[\alpha_1], a_{i+1} \ldots a_n) & \overset{*}{\vdash}_{d_1} (\xi\ B[\alpha_1]\ \xi_1\ D[\alpha_2], a_{p+1} \ldots a_n) \\ & \overset{*}{\vdash}_{d_2} (\xi\ B[\alpha_1]\ \xi_1\ D[\alpha_2]\ E[\beta], a_{q+1} \ldots a_n) \\ & \overset{*}{\vdash}_{d_3} (\xi\ B[\alpha_1]\ C[\beta\gamma], a_{j+1} \ldots a_n)\end{aligned}$$

donde en toda la derivación no se ha modificado $\xi\ B[\ ]$ y a lo sumo se ha consultado B. En la subderivación d_2 no se ha modificado $\xi\ B[\ ]\ \xi_1 D[\ ]$ y a lo sumo se ha consultado D. Las dos ocurrencias de β son la misma pila en el sentido de que ha sido transmitida sin modificación alguna a través de d_3: es posible que se hayan apilado índices en β y que posteriormente se hayan extraído, pero no se permite la extracción de elementos de β, aunque posteriores operaciones de apilamiento den como resultado una copia de β. En la figura 8.2 se muestra una representación gráfica de este tipo de transiciones.

Para cualquier $\xi' \in (P[V_I^*])^*$ se cumple

$$\begin{aligned}(\xi'\ B[\alpha_1], a_{i+1} \ldots a_n) & \overset{*}{\vdash}_{d_1} (\xi'\ B[\alpha_1]\ \xi_1\ D[\alpha_2], a_{p+1} \ldots a_n) \\ & \overset{*}{\vdash}_{d_2} (\xi'\ B[\alpha_1]\ \xi_1\ D[\alpha_2]\ E[\beta], a_{q+1} \ldots a_n) \\ & \overset{*}{\vdash}_{d_3} (\xi'\ B[\alpha_1]\ C[\beta\gamma], a_{j+1} \ldots a_n)\end{aligned}$$

dado que α_1 es independiente tanto de α_2 como de β ya que en las transiciones de apilamiento no se puede transmitir la pila de índices. Por la misma razón α_2 y β son independientes. En consecuencia, podemos utilizar ítems de la forma

$$[B, i, C, j, \gamma \mid D, p, E, q]$$

para representar este tipo de derivaciones.

Para completar la técnica de tabulación únicamente nos falta definir el mecanismo de combinación de ítems para los autómatas resultantes de los esquemas de compilación $*$-ascendentes. Las reglas de combinación de ítems de acuerdo con esas transiciones se muestran en la tabla 8.29, donde:

- Si k aparece en un ítem consecuente, entonces $k = j$ si $a = \epsilon$ y $k = j + 1$ si $a = a_{j+1}$.
- Si l aparece en un ítem consecuente, entonces $l = k$ si $a = \epsilon$ y $l = k + 1$ si $a = a_{k+1}$.

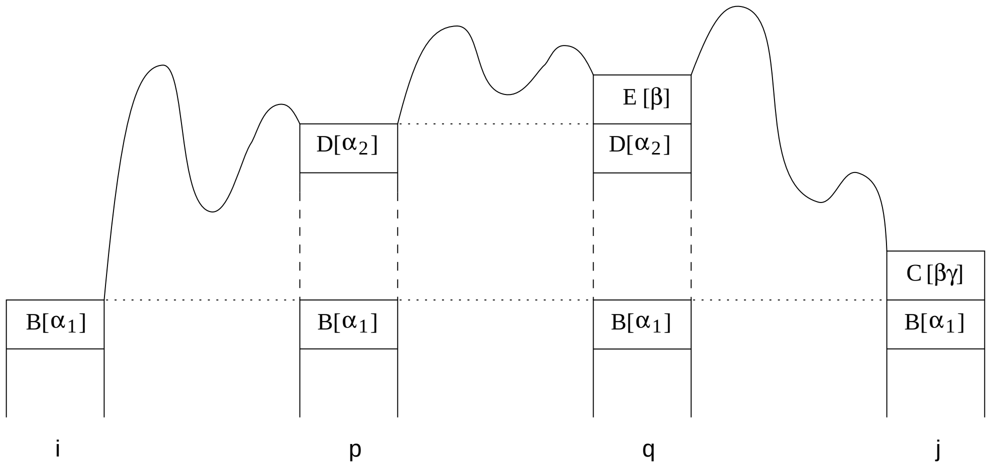

Figura 8.2: Derivaciones de retorno en estrategias $*$-ascendentes

El papel de ítem inicial le corresponde a

$$[-, 0, \$_0, 0 \mid -, -, -, -]$$

mientras que los ítems finales son de la forma

$$[\$_0, 0, S, n, - \mid -, -, -, -]$$

Teorema 8.2 *La manipulación de configuraciones mediante la aplicación de transiciones en los autómatas a pila restringidos con estrategias $*$-ascendentes es equivalente a la manipulación de ítems mediante las reglas de combinación de la tabla 8.29.*

Demostración:

Puesto que un ítem representa una derivación y toda derivación debe ser representada por algún ítem, es suficiente con demostrar que la combinación de los ítems produce ítems que se corresponden con derivaciones válidas y que para toda derivación que se pueda producir como resultado de la aplicación de una transición, existe una regla de combinación de ítems que produce un ítem que representa a dicha derivación. A continuación se muestra una lista de todos los casos posibles de derivación que se pueden dar junto con la correspondiente regla de combinación de ítems.

- Derivaciones que son el resultado de aplicar una transición $C[\circ\circ] \stackrel{a}{\longmapsto} C[\circ\circ]\ F[\,]$
  - a una derivación de llamada:

$$\begin{array}{ll}(\xi\ B[\alpha], a_{i+1} \dots a_n) & \stackrel{*}{\vdash} (\xi\ B[\alpha]\ C[\,], a_{j+1} \dots a_n) \\ & \vdash (\xi\ B[\alpha]\ C[\,]\ F[\,], a_{k+1} \dots a_n)\end{array}$$

$$\frac{[B, i, C, j, - \mid -, -, -, -]}{[C, j, F, k, - \mid -, -, -, -]}\ C[\circ\circ] \stackrel{a}{\longmapsto} C[\circ\circ]\ F[\,]$$

  - a una derivación de retorno:

$$\begin{array}{ll}(\xi\ B[\alpha_1], a_{i+1} \dots a_n) & \stackrel{*}{\vdash} (\xi\ B[\alpha_1]\ \xi_1\ D[\alpha_2], a_{p+1} \dots a_n) \\ & \stackrel{*}{\vdash} (\xi\ B[\alpha_1]\ \xi_1\ D[\alpha_2]\ E[\beta], a_{q+1} \dots a_n) \\ & \stackrel{*}{\vdash} (\xi\ B[\alpha_1]\ C[\beta\gamma], a_{j+1} \dots a_n) \\ & \vdash (\xi\ B[\alpha_1]\ C[\beta\gamma]\ F[\,], a_{k+1} \dots a_n)\end{array}$$

$$\frac{[B, i, C, j, \gamma \mid D, p, E, q]}{[C, j, F, k, - \mid -, -, -, -]}\ C[\circ\circ] \stackrel{a}{\longmapsto} C[\circ\circ]\ F[\,]$$

$$\frac{[B,i,C,j,\gamma \mid D,p,E,q]}{[C,j,F,k,- \mid -,-,-,-]} \quad C[\circ\circ] \overset{a}{\longmapsto} C[\circ\circ]\ F[\,]$$

$$\frac{[B,i,C,j,\gamma \mid D,p,E,q]}{[B,i,F,k,\gamma \mid D,p,E,q]} \quad C[\circ\circ] \overset{a}{\longmapsto} F[\circ\circ]$$

$$\frac{[B,i,C,j,\gamma \mid D,p,E,q]}{[B,i,F,k,\gamma' \mid B,i,C,j]} \quad C[\circ\circ] \overset{a}{\longmapsto} F[\circ\circ\gamma']$$

$$\frac{\begin{array}{c}[B,i,C,j,\gamma \mid D,p,E,q]\\ [D,p,E,q,\gamma' \mid O,u,P,v]\end{array}}{[B,i,F,k,\gamma' \mid O,u,P,v]} \quad C[\circ\circ\gamma] \overset{a}{\longmapsto} F[\circ\circ]$$

$$\frac{\begin{array}{c}[C,j,F,k,- \mid -,-,-,-]\\ [B,i,C,j,\gamma \mid D,p,E,q]\end{array}}{[B,i,G,l,\gamma \mid D,p,E,q]} \quad C[\circ\circ]\ F[\,] \overset{a}{\longmapsto} G[\circ\circ]$$

$$\frac{\begin{array}{c}[C,j,F,k,- \mid -,-,-,-]\\ [B,i,C,j,\gamma \mid D,p,E,q]\end{array}}{[B,i,G,l,\gamma' \mid B,i,C,j]} \quad C[\circ\circ]\ F[\,] \overset{a}{\longmapsto} G[\circ\circ\gamma']$$

$$\frac{\begin{array}{c}[C,j,F,k,- \mid -,-,-,-]\\ [B,i,C,j,\gamma \mid D,p,E,q]\\ [D,p,E,q,\gamma' \mid O,u,P,v]\end{array}}{[B,i,G,l,\gamma' \mid O,u,P,v]} \quad C[\circ\circ\gamma]\ F[\,] \overset{a}{\longmapsto} G[\circ\circ]$$

$$\frac{\begin{array}{c}[C,j,F,k,\gamma \mid D,p,E,q]\\ [B,i,C,j,- \mid -,-,-,-]\end{array}}{[B,i,G,l,\gamma \mid D,p,E,q]} \quad C[\,]\ F[\circ\circ] \overset{a}{\longmapsto} G[\circ\circ]$$

$$\frac{\begin{array}{c}[C,j,F,k,\gamma \mid D,p,E,q]\\ [B,i,C,j,- \mid -,-,-,-]\end{array}}{[B,i,G,l,\gamma' \mid C,j,F,j]} \quad C[\,]\ F[\circ\circ] \overset{a}{\longmapsto} G[\circ\circ\gamma']$$

$$\frac{\begin{array}{c}[C,j,F,k,\gamma \mid D,p,E,q]\\ [B,i,C,j,- \mid -,-,-,-]\\ [D,p,E,q,\gamma' \mid O,u,P,v]\end{array}}{[B,i,G,l,\gamma' \mid O,u,P,v]} \quad C[\,]\ F[\circ\circ\gamma] \overset{a}{\longmapsto} G[\circ\circ]$$

Tabla 8.29: Combinación de ítems en las estrategias $*$-ascendentes

- Derivaciones que son el resultado de aplicar una transición $C[\circ\circ] \overset{a}{\longmapsto} F[\circ\circ]$
  - a una derivación de llamada:

$$\begin{array}{ll}(\xi\ B[\alpha], a_{i+1}\dots a_n) & \overset{*}{\vdash} (\xi\ B[\alpha]\ C[\,], a_{j+1}\dots a_n)\\ & \vdash (\xi\ B[\alpha]\ F[\,], a_{k+1}\dots a_n)\end{array}$$

$$\frac{[B,i,C,j,-\mid -,-,-,-]}{[B,i,F,k,-\mid -,-,-,-]}\quad C[\circ\circ] \overset{a}{\longmapsto} F[\circ\circ]$$

  - a una derivación de retorno:

$$\begin{array}{ll}(\xi\ B[\alpha_1], a_{i+1}\dots a_n) & \overset{*}{\vdash} (\xi\ B[\alpha_1]\ \xi_1\ D[\alpha_2], a_{p+1}\dots a_n)\\ & \overset{*}{\vdash} (\xi\ B[\alpha_1]\ \xi_1\ D[\alpha_2]\ E[\beta], a_{q+1}\dots a_n)\\ & \overset{*}{\vdash} (\xi\ B[\alpha_1]\ C[\beta\gamma], a_{j+1}\dots a_n)\\ & \vdash (\xi\ B[\alpha_1]\ F[\beta\gamma], a_{k+1}\dots a_n)\end{array}$$

$$\frac{[B,i,C,j,\gamma\mid D,p,E,q]}{[B,i,F,k,\gamma\mid D,p,E,q]}\quad C[\circ\circ] \overset{a}{\longmapsto} F[\circ\circ]$$

- Derivaciones que son el resultado de aplicar una transición $C[\circ\circ] \overset{a}{\longmapsto} F[\circ\circ\gamma']$
  - a una derivación de llamada:

$$\begin{array}{ll}(\xi\ B[\alpha], a_{i+1}\dots a_n) & \overset{*}{\vdash} (\xi\ B[\alpha]\ C[\,], a_{j+1}\dots a_n)\\ & \vdash (\xi\ B[\alpha]\ F[\gamma'], a_{k+1}\dots a_n)\end{array}$$

$$\frac{[B,i,C,j,-\mid -,-,-,-]}{[B,i,F,k,\gamma'\mid B,i,C,j]}\quad C[\circ\circ] \overset{a}{\longmapsto} F[\circ\circ\gamma']$$

  - a una derivación de retorno:

$$\begin{array}{ll}(\xi\ B[\alpha_1], a_{i+1}\dots a_n) & \overset{*}{\vdash} (\xi\ B[\alpha_1]\ \xi_1\ D[\alpha_2], a_{p+1}\dots a_n)\\ & \overset{*}{\vdash} (\xi\ B[\alpha_1]\ \xi_1\ D[\alpha_2]\ E[\beta], a_{q+1}\dots a_n)\\ & \overset{*}{\vdash} (\xi\ B[\alpha_1]\ C[\beta\gamma], a_{j+1}\dots a_n)\\ & \vdash (\xi\ B[\alpha_1]\ F[\beta\gamma\gamma'], a_{k+1}\dots a_n)\end{array}$$

$$\frac{[B,i,C,j,\gamma\mid D,p,E,q]}{[B,i,F,k,\gamma'\mid B,i,C,j]}\quad C[\circ\circ] \overset{a}{\longmapsto} F[\circ\circ\gamma']$$

- Derivaciones que son el resultado de aplicar una transición $C[\circ\circ\gamma] \overset{a}{\longmapsto} F[\circ\circ]$ a una derivación de retorno que a su vez ha sido obtenida
  - a partir de una derivación de llamada:

$$\begin{array}{ll}(\xi\ B[\alpha_1], a_{i+1}\dots a_n) & \overset{*}{\vdash} (\xi\ B[\alpha_1]\ \xi_1\ D[\alpha_2], a_{p+1}\dots a_n)\\ & \overset{*}{\vdash} (\xi\ B[\alpha_1]\ \xi_1\ D[\alpha_2]\ E[\,], a_{q+1}\dots a_n)\\ & \overset{*}{\vdash} (\xi\ B[\alpha_1]\ C[\gamma], a_{j+1}\dots a_n)\\ & \vdash (\xi\ B[\alpha_1]\ F[\,], a_{k+1}\dots a_n)\end{array}$$

$$\frac{\begin{array}{c}[B,i,C,j,\gamma\mid D,p,E,q]\\ [D,p,E,q,-\mid -,-,-,-]\end{array}}{[B,i,F,k,-\mid -,-,-,-]}\quad C[\circ\circ\gamma] \overset{a}{\longmapsto} F[\circ\circ]$$

– a partir de una derivación de retorno:

$$\begin{array}{ll}(\xi\ B[\alpha_1], a_{i+1}\dots a_n) & \overset{*}{\vdash} (\xi\ B[\alpha_1]\ \xi_1\ D[\alpha_2], a_{p+1}\dots a_n)\\ & \overset{*}{\vdash} (\xi\ B[\alpha_1]\ \xi_1\ D[\alpha_2]\ \xi_2\ O[\alpha_3], a_{u+1}\dots a_n)\\ & \overset{*}{\vdash} (\xi\ B[\alpha_1]\ \xi_1\ D[\alpha_2]\ \xi_2\ O[\alpha_3]\ P[\beta], a_{v+1}\dots a_n)\\ & \overset{*}{\vdash} (\xi\ B[\alpha_1]\ \xi_1\ D[\alpha_2]\ E[\beta\gamma'], a_{q+1}\dots a_n)\\ & \overset{*}{\vdash} (\xi\ B[\alpha_1]\ C[\beta\gamma'\gamma], a_{j+1}\dots a_n)\\ & \vdash (\xi\ B[\alpha_1]\ F[\beta\gamma'], a_{k+1}\dots a_n)\end{array}$$

$$\frac{\begin{array}{c}[B,i,C,j,\gamma \mid D,p,E,q]\\ [D,p,E,q,\gamma' \mid O,u,P,v]\end{array}}{[B,i,F,k,\gamma' \mid O,u,P,v]}\ C[\circ\circ\gamma] \overset{a}{\longmapsto} F[\circ\circ]$$

- Derivaciones que son el resultado de aplicar una transición $C[\circ\circ]\ F[\,] \overset{a}{\longmapsto} G[\circ\circ]$ a una derivación obtenida tras aplicar una transición $C[\circ\circ] \overset{b}{\longmapsto} C[\circ\circ]\ F'[\,]$
  – a una derivación de llamada:

$$\begin{array}{ll}(\xi\ B[\alpha_1], a_{i+1}\dots a_n) & \overset{*}{\vdash} (\xi\ B[\alpha_1]\ C[\,], a_{j+1}\dots a_n)\\ & \vdash (\xi\ B[\alpha_1]\ C[\,]\ F'[\,], a_{k'+1}\dots a_n)\\ & \overset{*}{\vdash} (\xi\ B[\alpha_1]\ C[\,]\ F[\,], a_{k+1}\dots a_n)\\ & \vdash (\xi\ B[\alpha_1]\ G[\,], a_{l+1}\dots a_n)\end{array}$$

$$\frac{\begin{array}{c}[C,j,F,k,- \mid -,-,-,-]\\ [B,i,C,j,- \mid -,-,-,-]\end{array}}{[B,i,G,l,- \mid -,-,-,-]}\ C[\circ\circ]\ F[\,] \overset{a}{\longmapsto} G[\circ\circ]$$

  – a una derivación de retorno:

$$\begin{array}{ll}(\xi\ B[\alpha_1], a_{i+1}\dots a_n) & \overset{*}{\vdash} (\xi\ B[\alpha_1]\ \xi_1\ D[\alpha_2], a_{p+1}\dots a_n)\\ & \overset{*}{\vdash} (\xi\ B[\alpha_1]\ \xi_1\ D[\alpha_2]\ E[\beta], a_{q+1}\dots a_n)\\ & \overset{*}{\vdash} (\xi\ B[\alpha_1]\ C[\beta\gamma], a_{j+1}\dots a_n)\\ & \vdash (\xi\ B[\alpha_1]\ C[\beta\gamma]\ F'[\,], a_{k'+1}\dots a_n)\\ & \overset{*}{\vdash} (\xi\ B[\alpha_1]\ C[\beta\gamma]\ F[\,], a_{k+1}\dots a_n)\\ & \vdash (\xi\ B[\alpha_1]\ G[\beta\gamma], a_{l+1}\dots a_n)\end{array}$$

$$\frac{\begin{array}{c}[C,j,F,k,- \mid -,-,-,-]\\ [B,i,C,j,\gamma \mid D,p,E,q]\end{array}}{[B,i,G,l,\gamma \mid D,p,E,q]}\ C[\circ\circ]\ F[\,] \overset{a}{\longmapsto} G[\circ\circ]$$

  donde $k' = j$ si $b = \epsilon$ y $k' = j+1$ si $b = a_{j+1}$.

- Derivaciones que son el resultado de aplicar una transición $C[\circ\circ]\ F[\,] \overset{a}{\longmapsto} G[\circ\circ\gamma']$ a una derivación obtenida tras aplicar una transición $C[\circ\circ] \overset{b}{\longmapsto} C[\circ\circ]\ F'[\,]$
  – a una derivación de llamada:

$$\begin{array}{ll}(\xi\ B[\alpha_1], a_{i+1}\dots a_n) & \overset{*}{\vdash} (\xi\ B[\alpha_1]\ C[\,], a_{j+1}\dots a_n)\\ & \vdash (\xi\ B[\alpha_1]\ C[\,]\ F'[\,], a_{k'+1}\dots a_n)\\ & \overset{*}{\vdash} (\xi\ B[\alpha_1]\ C[\,]\ F[\,], a_{k+1}\dots a_n)\\ & \vdash (\xi\ B[\alpha_1]\ G[\gamma'], a_{l+1}\dots a_n)\end{array}$$

$$\frac{\begin{array}{c}[C,j,F,k,- \mid -,-,-,-]\\ [B,i,C,j,- \mid -,-,-,-]\end{array}}{[B,i,G,l,\gamma' \mid B,i,C,j]}\ C[\circ\circ]\ F[\,] \overset{a}{\longmapsto} G[\circ\circ\gamma']$$

– a una derivación de retorno:

$$\begin{array}{ll}(\xi\ B[\alpha_1], a_{i+1}\dots a_n) & \overset{*}{\vdash} (\xi\ B[\alpha_1]\ \xi_1\ D[\alpha_2], a_{p+1}\dots a_n)\\ & \overset{*}{\vdash} (\xi\ B[\alpha_1]\ \xi_1\ D[\alpha_2]\ E[\beta], a_{q+1}\dots a_n)\\ & \overset{*}{\vdash} (\xi\ B[\alpha_1]\ C[\beta\gamma], a_{j+1}\dots a_n)\\ & \vdash (\xi\ B[\alpha_1]\ C[\beta\gamma]\ F'[\,], a_{k'+1}\dots a_n)\\ & \overset{*}{\vdash} (\xi\ B[\alpha_1]\ C[\beta\gamma]\ F[\,], a_{k+1}\dots a_n)\\ & \vdash (\xi\ B[\alpha_1]\ G[\beta\gamma\gamma'], a_{l+1}\dots a_n)\end{array}$$

$$\frac{\begin{array}{c}[C,j,F,k,-\mid -,-,-,-]\\ [B,i,C,j,\gamma\mid D,p,E,q]\end{array}}{[B,i,G,l,\gamma'\mid B,i,C,j]}\quad C[\circ\circ]\ F[\,] \overset{a}{\longmapsto} G[\circ\circ\gamma']$$

donde $k' = j$ si $b = \epsilon$ y $k' = j + 1$ si $b = a_{j+1}$.

- Derivaciones que son el resultado de aplicar una transición $C[\circ\circ\gamma]\ F[\,] \overset{a}{\longmapsto} G[\circ\circ]$ a una derivación obtenida tras aplicar una transición $C[\circ\circ] \overset{b}{\longmapsto} C[\circ\circ]\ F'[\,]$ a una derivación de retorno, con los dos casos siguientes:
  – la derivación de retorno es a su vez obtenida a partir de una derivación de llamada:

$$\begin{array}{ll}(\xi\ B[\alpha_1], a_{i+1}\dots a_n) & \overset{*}{\vdash} (\xi\ B[\alpha_1]\ \xi_1\ D[\alpha_2], a_{p+1}\dots a_n)\\ & \overset{*}{\vdash} (\xi\ B[\alpha_1]\ \xi_1\ D[\alpha_2]\ E[\,], a_{q+1}\dots a_n)\\ & \overset{*}{\vdash} (\xi\ B[\alpha_1]\ C[\gamma], a_{j+1}\dots a_n)\\ & \vdash (\xi\ B[\alpha_1]\ C[\gamma]\ F'[\,], a_{k'+1}\dots a_n)\\ & \overset{*}{\vdash} (\xi\ B[\alpha_1]\ C[\gamma]\ F[\,], a_{k+1}\dots a_n)\\ & \vdash (\xi\ B[\alpha_1]\ G[\,], a_{l+1}\dots a_n)\end{array}$$

$$\frac{\begin{array}{c}[C,j,F,k,-\mid -,-,-,-]\\ [B,i,C,j,\gamma\mid D,p,E,q]\\ [D,p,E,q,-\mid -,-,-,-]\end{array}}{[B,i,G,l,-\mid -,-,-,-]}\quad C[\circ\circ\gamma]\ F[\,] \overset{a}{\longmapsto} G[\circ\circ]$$

  – la derivación de retorno es a su vez obtenida a partir de a una derivación de retorno:

$$\begin{array}{ll}(\xi\ B[\alpha_1], a_{i+1}\dots a_n) & \overset{*}{\vdash} (\xi\ B[\alpha_1]\ \xi_1\ D[\alpha_2], a_{p+1}\dots a_n)\\ & \overset{*}{\vdash} (\xi\ B[\alpha_1]\ \xi_1\ D[\alpha_2]\ \xi_2\ O[\alpha_3], a_{u+1}\dots a_n)\\ & \overset{*}{\vdash} (\xi\ B[\alpha_1]\ \xi_1\ D[\alpha_2]\ \xi_2\ O[\alpha_3]\ P[\beta], a_{v+1}\dots a_n)\\ & \overset{*}{\vdash} (\xi\ B[\alpha_1]\ \xi_1\ D[\alpha_2]\ E[\beta\gamma'], a_{q+1}\dots a_n)\\ & \overset{*}{\vdash} (\xi\ B[\alpha_1]\ C[\beta\gamma'\gamma], a_{j+1}\dots a_n)\\ & \vdash (\xi\ B[\alpha_1]\ C[\beta\gamma'\gamma]\ F'[\,], a_{k'+1}\dots a_n)\\ & \overset{*}{\vdash} (\xi\ B[\alpha_1]\ C[\beta\gamma'\gamma]\ F[\,], a_{k+1}\dots a_n)\\ & \vdash (\xi\ B[\alpha_1]\ G[\beta\gamma'], a_{l+1}\dots a_n)\end{array}$$

$$\frac{\begin{array}{c}[C,j,F,k,-\mid -,-,-,-]\\ [B,i,C,j,\gamma\mid D,p,E,q]\\ [D,p,E,q,\gamma'\mid O,u,P,v]\end{array}}{[B,i,G,l,\gamma'\mid O,u,P,v]}\quad C[\circ\circ\gamma]\ F[\,] \overset{a}{\longmapsto} G[\circ\circ]$$

donde $k' = j$ si $b = \epsilon$ y $k' = j + 1$ si $b = a_{j+1}$.

- Derivaciones que son el resultado de aplicar una transición $C[\,]\ F[\circ\circ] \overset{a}{\longmapsto} G[\circ\circ]$ a una derivación obtenida tras aplicar una transición $C[\circ\circ] \overset{b}{\longmapsto} C[\circ\circ]\ F'[\,]$ a una derivación de llamada[7], con los dos casos siguientes:

[7] Si se hubiese aplicado esta transición a una derivación de retorno, no sería posible tener $C[\,]$ bajo $F[\,]$.

– la derivación obtenida es una derivación de llamada:

$$\begin{array}{ll}(\xi\ B[\alpha_1], a_{i+1}\dots a_n) & \overset{*}{\vdash} (\xi\ B[\alpha_1]\ C[\,], a_{j+1}\dots a_n)\\ & \vdash (\xi\ B[\alpha_1]\ C[\,]\ F'[\,], a_{k'+1}\dots a_n)\\ & \overset{*}{\vdash} (\xi\ B[\alpha_1]\ C[\,]\ F[\,], a_{k+1}\dots a_n)\\ & \vdash (\xi\ B[\alpha_1]\ G[\,], a_{l+1}\dots a_n)\end{array}$$

$$\frac{\begin{array}{c}[C,j,F,k,-\mid -,-,-,-]\\ [B,i,C,j,-\mid -,-,-,-]\end{array}}{[B,i,G,l,-\mid -,-,-,-]}\ C[\,]\ F[\circ\circ] \overset{a}{\longmapsto} G[\circ\circ]$$

– la derivación obtenida es una derivación de retorno:

$$\begin{array}{ll}(\xi\ B[\alpha_1], a_{i+1}\dots a_n) & \overset{*}{\vdash} (\xi\ B[\alpha_1]\ C[\,], a_{j+1}\dots a_n)\\ & \vdash (\xi\ B[\alpha_1]\ C[\,]\ F'[\,], a_{k'+1}\dots a_n)\\ & \overset{*}{\vdash} (\xi\ B[\alpha_1]\ C[\,]\ F'[\,]\ \xi_1\ D[\alpha_2], a_{p+1}\dots a_n)\\ & \overset{*}{\vdash} (\xi\ B[\alpha_1]\ C[\,]\ F'[\,]\ \xi_1\ D[\alpha_2]\ E[\beta], a_{q+1}\dots a_n)\\ & \overset{*}{\vdash} (\xi\ B[\alpha_1]\ C[\,]\ F[\beta\gamma], a_{k+1}\dots a_n)\\ & \vdash (\xi\ B[\alpha_1]\ G[\beta\gamma], a_{l+1}\dots a_n)\end{array}$$

$$\frac{\begin{array}{c}[C,j,F,k,\gamma\mid D,p,E,q]\\ [B,i,C,j,-\mid -,-,-,-]\end{array}}{[B,i,G,l,\gamma\mid D,p,E,q]}\ C[\,]\ F[\circ\circ] \overset{a}{\longmapsto} G[\circ\circ]$$

donde $k' = j$ si $b = \epsilon$ y $k' = j+1$ si $b = a_{j+1}$.

- Derivaciones que son el resultado de aplicar una transición $C[\,]\ F[\circ\circ] \overset{a}{\longmapsto} G[\circ\circ\gamma']$ a una derivación obtenida tras aplicar una transición $C[\circ\circ] \overset{b}{\longmapsto} C[\circ\circ]\ F'[\,]$ a una derivación de llamada, con los dos casos siguientes:

  – la derivación obtenida es una derivación de llamada:

$$\begin{array}{ll}(\xi\ B[\alpha_1], a_{i+1}\dots a_n) & \overset{*}{\vdash} (\xi\ B[\alpha_1]\ C[\,], a_{j+1}\dots a_n)\\ & \vdash (\xi\ B[\alpha_1]\ C[\,]\ F'[\,], a_{k'+1}\dots a_n)\\ & \overset{*}{\vdash} (\xi\ B[\alpha_1]\ C[\,]\ F[\,], a_{k+1}\dots a_n)\\ & \vdash (\xi\ B[\alpha_1]\ G[\gamma'], a_{l+1}\dots a_n)\end{array}$$

$$\frac{\begin{array}{c}[C,j,F,k,-\mid -,-,-,-]\\ [B,i,C,j,-\mid -,-,-,-]\end{array}}{[B,i,G,l,\gamma'\mid C,j,F,j]}\ C[\,]\ F[\circ\circ] \overset{a}{\longmapsto} G[\circ\circ\gamma']$$

  – la derivación obtenida es una derivación de retorno:

$$\begin{array}{ll}(\xi\ B[\alpha_1], a_{i+1}\dots a_n) & \overset{*}{\vdash} (\xi\ B[\alpha_1]\ C[\,], a_{j+1}\dots a_n)\\ & \vdash (\xi\ B[\alpha_1]\ C[\,]\ F'[\,], a_{k'+1}\dots a_n)\\ & \overset{*}{\vdash} (\xi\ B[\alpha_1]\ C[\,]\ F'[\,]\ \xi_1\ D[\alpha_2], a_{p+1}\dots a_n)\\ & \overset{*}{\vdash} (\xi\ B[\alpha_1]\ C[\,]\ F'[\,]\ \xi_1\ D[\alpha_2]\ E[\beta], a_{q+1}\dots a_n)\\ & \overset{*}{\vdash} (\xi\ B[\alpha_1]\ C[\,]\ F[\beta\gamma], a_{k+1}\dots a_n)\\ & \vdash (\xi\ B[\alpha_1]\ G[\beta\gamma\gamma'], a_{l+1}\dots a_n)\end{array}$$

$$\frac{\begin{array}{c}[C,j,F,k,\gamma\mid D,p,E,q]\\ [B,i,C,j,-\mid -,-,-,-]\end{array}}{[B,i,G,l,\gamma'\mid C,j,F,j]}\ C[\,]\ F[\circ\circ] \overset{a}{\longmapsto} G[\circ\circ\gamma']$$

donde $k' = j$ si $b = \epsilon$ y $k' = j+1$ si $b = a_{j+1}$.

- Derivaciones que son el resultado de aplicar una transición $C[\,]\ F[\circ\circ\gamma] \overset{a}{\longmapsto} G[\circ\circ]$ a una derivación obtenida tras aplicar una transición $C[\circ\circ] \overset{b}{\longmapsto} C[\circ\circ]\ F'[\,]$ a una derivación de llamada, con los dos casos siguientes:
  - la derivación obtenida es una derivación de llamada:

$$
\begin{array}{ll}
(\xi\ B[\alpha_1], a_{i+1}\dots a_n) & \overset{*}{\vdash} (\xi\ B[\alpha_1]\ C[\,], a_{j+1}\dots a_n) \\
& \vdash (\xi\ B[\alpha_1]\ C[\,]\ F'[\,], a_{k'+1}\dots a_n) \\
& \overset{*}{\vdash} (\xi\ B[\alpha_1]\ C[\,]\ F'[\,]\ \xi_1\ D[\alpha_2], a_{p+1}\dots a_n) \\
& \overset{*}{\vdash} (\xi\ B[\alpha_1]\ C[\,]\ F'[\,]\ \xi_1\ D[\alpha_2]\ E[\,], a_{q+1}\dots a_n) \\
& \overset{*}{\vdash} (\xi\ B[\alpha_1]\ C[\,]\ F[\gamma'], a_{k+1}\dots a_n) \\
& \vdash (\xi\ B[\alpha_1]\ G[\,], a_{l+1}\dots a_n)
\end{array}
$$

$$
\frac{\begin{array}{l}[C,j,F,k,\gamma \mid D,p,E,q] \\ [B,i,C,j,- \mid -,-,-,-] \\ [D,p,E,q,- \mid -,-,-,-]\end{array}}{[B,i,G,l,- \mid -,-,-,-]} \quad C[\,]\ F[\circ\circ\gamma] \overset{a}{\longmapsto} G[\circ\circ]
$$

  - la derivación obtenida es una derivación de retorno:

$$
\begin{array}{ll}
(\xi\ B[\alpha_1], a_{i+1}\dots a_n) & \overset{*}{\vdash} (\xi\ B[\alpha_1]\ C[\,], a_{j+1}\dots a_n) \\
& \vdash (\xi\ B[\alpha_1]\ C[\,]\ F'[\,], a_{k'+1}\dots a_n) \\
& \overset{*}{\vdash} (\xi\ B[\alpha_1]\ C[\,]\ F'[\,]\ \xi_1\ D[\alpha_2], a_{p+1}\dots a_n) \\
& \overset{*}{\vdash} (\xi\ B[\alpha_1]\ C[\,]\ F'[\,]\ \xi_1\ D[\alpha_2]\ \xi_2\ O[\alpha_3], a_{u+1}\dots a_n) \\
& \overset{*}{\vdash} (\xi\ B[\alpha_1]\ C[\,]\ F'[\,]\ \xi_1\ D[\alpha_2]\ \xi_2\ O[\alpha_3]\ P[\beta], a_{v+1}\dots a_n) \\
& \overset{*}{\vdash} (\xi\ B[\alpha_1]\ C[\,]\ F'[\,]\ \xi_1\ D[\alpha_2]\ E[\beta\gamma'], a_{q+1}\dots a_n) \\
& \overset{*}{\vdash} (\xi\ B[\alpha_1]\ C[\,]\ F[\beta\gamma'\gamma], a_{k+1}\dots a_n) \\
& \vdash (\xi\ B[\alpha_1]\ G[\beta\gamma'], a_{l+1}\dots a_n)
\end{array}
$$

$$
\frac{\begin{array}{l}[C,j,F,k,\gamma \mid D,p,E,q] \\ [B,i,C,j,- \mid -,-,-,-] \\ [D,p,E,q,\gamma' \mid O,u,P,v]\end{array}}{[B,i,G,l,\gamma' \mid O,u,P,v]} \quad C[\,]\ F[\circ\circ\gamma] \overset{a}{\longmapsto} G[\circ\circ]
$$

  donde $k' = j$ si $b = \epsilon$ y $k' = j + 1$ si $b = a_{j+1}$.

Si aplicamos inducción en la longitud de una derivación, a partir de la lista observamos que para cualquier derivación obtenida mediante la aplicación de una transición, existe una regla de combinación que busca los ítems correspondientes a las subderivaciones relevantes y que produce el ítem correspondiente a la derivación resultante. También podemos observar que dada una regla de combinación de ítems, existen subderivaciones en el autómata que se corresponden con cada uno de los ítems antecedentes y que combinadas entre sí producen la derivación correspondiente al ítem consecuente. □

La complejidad temporal de la técnica de tabulación con respecto a longitud n de la cadena de entrada es de orden $\mathcal{O}(n^6)$ en el peor caso. Dicha complejidad viene dada por la regla de combinación de ítems correspondiente a producciones del tipo $B[\,]\ C[\circ\circ\gamma] \overset{a}{\longmapsto} F[\circ\circ]$, que involucran el manejo de 7 posiciones con respecto a la cadena de entrada, aunque sólo 6 de ellas de manera simultánea. La complejidad espacial es de orden $\mathcal{O}(n^4)$ puesto que cada ítem contiene cuatro posiciones con respeto a la cadena de entrada.

8.5.2 Tabulación de estrategias ascendentes-ascendentes

Como un caso particular de las estrategias $*$-ascendentes tenemos las estrategias de análisis sintáctico ascendentes-ascendentes, en las cuales no se realiza ningún tipo de predicción en la fase de llamada, ni siquiera sobre los no-terminales, mientras que toda la información disponible acerca de los no-terminales y las pilas de índices es propagada en la fase de retorno. Con respecto a la parte independiente del contexto, las estrategias ascendentes están incluidas en la clase de los autómatas lógicos a pila débilmente predictivos, que pueden ser tabulados mediante ítems que almacenan el elemento en la cima de la pila y las posiciones de la cadena de entrada cuando se apilaron los dos elementos en la cima de la pila [52].

Toda configuración de un autómata a pila construido según una estrategia de este tipo puede ser clasificada en uno de los dos tipos de derivaciones que se relacionan a continuación:

Derivaciones de llamada. Corresponden a configuraciones

$$(\xi, a_{i+1} \dots a_n) \overset{*}{\vdash} (\xi\ C[\,], a_{j+1} \dots a_n)$$

donde en toda la derivación no se ha modificado ξ y a lo sumo se consulta la cima de ξ. Para cualquier $\xi' \in (P[V_I^*])^*$ se cumple

$$(\xi', a_{i+1} \dots a_n) \overset{*}{\vdash} (\xi'\ C[\,], a_{j+1} \dots a_n)$$

por lo que este tipo de derivaciones puede ser representado mediante ítems de la forma

$$[i, C, j, - \mid -, -, -]$$

Derivaciones de retorno. Corresponden a configuraciones

$$\begin{aligned}(\xi, a_{i+1} \dots a_n) \ &\overset{*}{\vdash}_{d_1} (\xi\ \xi_1, a_{p+1} \dots a_n)\\ &\overset{*}{\vdash}_{d_2} (\xi\ \xi_1\ E[\beta], a_{q+1} \dots a_n)\\ &\overset{*}{\vdash}_{d_3} (\xi\ C[\beta\gamma], a_{j+1} \dots a_n)\end{aligned}$$

en las que ξ no ha sido modificado en toda la derivación, permitiéndose la consulta de la cima de ξ. En d_2 no se permite la modificación de $\xi\xi_1$ pero sí la consulta de la cima de ξ_1. Las dos ocurrencias de β se refieren a una sola pila de índices. Para cualquier $\xi' \in (P[V_I^*])^*$ se cumple

$$\begin{aligned}(\xi', a_{i+1} \dots a_n) \ &\overset{*}{\vdash}_{d_1} (\xi'\ \xi_1, a_{p+1} \dots a_n)\\ &\overset{*}{\vdash}_{d_2} (\xi'\ \xi_1\ E[\beta], a_{q+1} \dots a_n)\\ &\overset{*}{\vdash}_{d_3} (\xi'\ C[\beta\gamma], a_{j+1} \dots a_n)\end{aligned}$$

por lo que pueden utilizarse ítems de la forma

$$[i, C, j, \gamma \mid p, E, q]$$

para representar este tipo de derivaciones.

En la tabla 8.30 se muestran las reglas de combinación de ítems para las transiciones de la tabla 8.28. El ítem inicial es de la forma

$$[0, \$_0, 0 \mid -, -, -]$$

mientras que los ítems finales tienen de la forma

$$[0, S, n, - \mid -, -, -]$$

Esta técnica de tabulación sólo es correcta para aquellos esquemas de compilación que se definen utilizando las transiciones de la tabla 8.28 y cuya estrategia independiente del contexto es débilmente predictiva. La prueba de corrección y completud es análoga a la del caso de los RLPDA $*$-ascendentes y será omitida. En la sección C.8 se muestra la aplicación de esta técnica de tabulación a un esquema de compilación de tipo LR para LIG.

La complejidad con respecto a la longitud de la cadena de entrada de esta técnica de tabulación no experimenta variación con respecto a la complejidad de la técnica general para estrategias $*$-ascendente. En consecuencia, la complejidad temporal en el peor caso es de orden $\mathcal{O}(n^6)$ y la temporal $\mathcal{O}(n^4)$.

8.5.3 Tabulación de estrategias $*$-Earley

Las estrategias de tipo $*$-Earley se caracterizan por el tratamiento que reciben las pilas de índices, que son utilizadas tanto en la fase de llamada como en la de retorno. Durante la fase de llamada se predice qué elementos deben ser apilados y extraídos de la pila de índices. Durante la fase de retorno, los mismos elementos de las pilas de índices son apilados y extraídos, en orden inverso al de la fase de llamada. En cualquier punto de una derivación, las pilas construidas en las fases de llamada y de retorno deben contener los mismo elementos, aunque las pilas en sí sean diferentes, pues una nace y muere en la fase descendente y la otra nace y muere en la fase ascendente.

La tabla 8.31 muestra las transiciones utilizadas por los esquemas de compilación de LIG y TAG que incorporan estrategias $*$-Earley. Con este conjunto de transiciones es posible construir autómatas a pila restringidos que aceptan lenguajes que no pertenecen a la clase de los lenguajes de adjunción de árboles. Ello se debe a que la utilización combinada de transiciones $C[\circ\circ\gamma] \longmapsto C[\circ\circ\gamma]\ F[\circ\circ\gamma']$ y $C[\circ\circ]\ F[\,] \longmapsto G[\circ\circ]$ permitiría crear derivaciones de la forma

$$\begin{array}{rl}
(\xi C_1[\delta], w_1) & \vdash (\xi C[\delta] F_1[\delta], w_1) \\
& \overset{*}{\vdash} (\xi C[\delta] F_1'[\,], w_2) \\
& \vdash (\xi C_2[\delta], w_2) \\
& \vdash (\xi C_2[\delta] F_2[\,], w_2) \\
& \overset{*}{\vdash} (\xi C_2[\delta] F_2'[\delta], w_3) \\
& \vdash (\xi C_3[\delta], w_3) \\
& \vdash (\xi C_3[\delta] F_3[\,], w_3) \\
& \vdots
\end{array}$$

en las cuales una pila de índices es compartida por F_1, F_2, F_3, ..., perdiéndose de ese modo la linealidad.

Por otra parte, las transiciones de la tabla 8.31 no aseguran que las pilas de índices construidas durante la fase de llamada y de retorno contengan los mismos valores. Por ello en esta sección nos centraremos en el desarrollo de una técnica de tabulación para aquellos autómatas lógicos a pila restringidos con estrategia $*$-Earley cuyas derivaciones de retorno construyen las mismas pilas de índices que las derivaciones de llamada y mediante las mismas operaciones de apilamiento y de extracción, aunque aplicadas en orden inverso. En particular, este es el caso

$$\frac{[i,C,j,\gamma \mid p,E,q]}{[j,F,k,- \mid -,-,-]} \quad C[\circ\circ] \stackrel{a}{\longmapsto} C[\circ\circ]\ F[\ ]$$

$$\frac{[i,C,j,\gamma \mid p,E,q]}{[i,F,k,\gamma \mid p,E,q]} \quad C[\circ\circ] \stackrel{a}{\longmapsto} F[\circ\circ]$$

$$\frac{[i,C,j,\gamma \mid p,E,q]}{[i,F,k,\gamma' \mid i,C,j]} \quad C[\circ\circ] \stackrel{a}{\longmapsto} F[\circ\circ\gamma']$$

$$\frac{\begin{array}{c}[i,C,j,\gamma \mid p,E,q]\\ [p,E,q,\gamma' \mid u,P,v]\end{array}}{[i,F,k,\gamma' \mid u,P,v]} \quad C[\circ\circ\gamma] \stackrel{a}{\longmapsto} F[\circ\circ]$$

$$\frac{\begin{array}{c}[i,C,j,- \mid -,-,-]\\ [m,B,i,\gamma \mid p,E,q]\end{array}}{[m,F,k,\gamma \mid p,E,q]} \quad B[\circ\circ]\ C[\ ] \stackrel{a}{\longmapsto} F[\circ\circ]$$

$$\frac{\begin{array}{c}[i,C,j,- \mid -,-,-]\\ [m,B,i,\gamma \mid p,E,q]\end{array}}{[m,F,k,\gamma' \mid m,B,i]} \quad B[\circ\circ]\ C[\ ] \stackrel{a}{\longmapsto} F[\circ\circ\gamma']$$

$$\frac{\begin{array}{c}[i,C,j,- \mid -,-,-]\\ [m,B,i,\gamma \mid p,E,q]\\ [p,E,q,\gamma' \mid u,P,v]\end{array}}{[m,F,k,\gamma' \mid u,P,v]} \quad B[\circ\circ\gamma]\ C[\ ] \stackrel{a}{\longmapsto} F[\circ\circ]$$

$$\frac{\begin{array}{c}[i,C,j,\gamma \mid p,E,q]\\ [m,B,i,- \mid -,-,-]\end{array}}{[m,F,k,\gamma \mid p,E,q]} \quad B[\ ]\ C[\circ\circ] \stackrel{a}{\longmapsto} F[\circ\circ]$$

$$\frac{\begin{array}{c}[i,C,j,\gamma \mid p,E,q]\\ [m,B,i,- \mid -,-,-]\end{array}}{[m,F,k,\gamma' \mid i,C,j]} \quad B[\ ]\ C[\circ\circ] \stackrel{a}{\longmapsto} F[\circ\circ\gamma']$$

$$\frac{\begin{array}{c}[i,C,j,\gamma \mid p,E,q]\\ [m,B,i,- \mid -,-,-]\\ [p,E,q,\gamma' \mid u,P,v]\end{array}}{[m,F,k,\gamma' \mid u,P,v]} \quad B[\ ]\ C[\circ\circ\gamma] \stackrel{a}{\longmapsto} F[\circ\circ]$$

Tabla 8.30: Combinación de ítems en las estrategia ascendente-ascendente

Transición	Compilación de LIG	Compilación de TAG
$C[\circ\circ] \longmapsto C[\circ\circ]\ F[\,]$	**[INIT][CALL]**	**[INIT][CALL]**
$C[\circ\circ\gamma] \longmapsto C[\circ\circ\gamma]\ F[\circ\circ\gamma']$	**[SCALL]**	**[SCALL][ACALL][FCALL]**
$C[\circ\circ] \longmapsto F[\circ\circ]$	**[SEL][PUB]**	**[SEL][PUB]**
$C[\circ\circ]\ F[\,] \longmapsto G[\circ\circ]$	**[RET]**	**[RET]**
$C[\circ\circ_1\gamma]\ F[\circ\circ_2\gamma'] \longmapsto G[\circ\circ_2\gamma]$	**[SRET]**	**[SRET][ARET][FRET]**
$C[\,] \overset{a}{\longmapsto} F[\,]$	**[SCAN]**	**[SCAN]**

Tabla 8.31: Tipos de transiciones en las estrategias $*$-Earley

de los RLPDA obtenidos partir de los esquemas de compilación de LIG y TAG que incorporan estrategias $*$-Earley.

Teniendo en cuenta las restricciones que acabamos de imponer, toda derivación debe ser de uno de los tres tipos que se muestran a continuación.

Derivaciones de llamada. Corresponden a la transmisión de una pila de índices en la fase descendente de la estrategia de análisis, e implican la existencia de la siguiente secuencia de subderivaciones:

$$\begin{aligned}(\xi\ A[\delta], a_{h+1} \ldots a_n) &\overset{*}{\vdash}_{d_1} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma'], a_{i+1} \ldots a_n)\\ &\overset{*}{\vdash}_{d_2} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']\ C[\delta\gamma], a_{j+1} \ldots a_n)\end{aligned}$$

donde $\gamma, \gamma' \in V_I$, $\delta, \delta' \in V_I^*$ y bien $\delta = \delta'\gamma'$, bien $\delta'\gamma' = \delta\gamma$ o bien $\delta' = \delta\gamma$. La pila $\xi\ A[\delta]$ no debe ser alterada en ningún momento, aunque se permite la consulta de A. La pila $\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']$ no debe ser alterada durante la subderivación d_2, aunque se permite la consulta de B y de γ'. $A[\delta]$ se corresponde con el elemento de la pila del autómata más próximo a la cima tal que su pila asociada es δ. Evidentemente, si $\delta'\gamma'{=}\delta$, entonces $A[\delta]$ y $B[\delta'\gamma']$ son el mismo elemento. En la figura 8.3 se muestra una representación gráfica de este tipo de transiciones.

Para cualquier $\xi' \in (P[V_I^*])^*$ y $\alpha \in V_I*$ tal que las restricciones previamente enunciadas para δ y δ' son aplicadas a α y α', se cumple que:

$$\begin{aligned}(\xi'\ A[\alpha], a_{h+1} \ldots a_n) &\overset{*}{\vdash}_{d_1} (\xi'\ A[\alpha]\ \xi_1'\ B[\alpha'\gamma'], a_{i+1} \ldots a_n)\\ &\overset{*}{\vdash}_{d_2} (\xi'\ A[\alpha]\ \xi_1'\ B[\alpha'\gamma']\ C[\alpha\gamma], a_{j+1} \ldots a_n)\end{aligned}$$

por lo que este tipo de derivaciones se puede representar de modo condensado mediante ítems de la forma

$$[A, h \mid B, i, \gamma', C, j, \gamma \mid -, -, -, -]$$

Derivaciones de retorno. Corresponden a la transmisión de una pila de índices durante la fase ascendente de la estrategia de análisis e implican la existencia de la siguiente secuencia de subderivaciones:

$$\begin{array}{rl}(\xi\ A[\delta], a_{h+1}\ldots a_n) & \overset{*}{\vdash}_{d_1} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma'], a_{i+1}\ldots a_n) \\ & \overset{*}{\vdash}_{d_2} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']\ \xi_2\ D[\delta\gamma], a_{p+1}\ldots a_n) \\ & \overset{*}{\vdash}_{d_3} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']\ \xi_2\ D[\delta\gamma]\ E[\beta], a_{q+1}\ldots a_n) \\ & \overset{*}{\vdash}_{d_4} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']\ C[\beta\gamma], a_{j+1}\ldots a_n)\end{array}$$

donde $\gamma, \gamma', \eta \in V_I$, $\delta, \delta' \in V_I^*$, y bien $\delta = \delta'\gamma'$, bien $\delta'\gamma' = \delta\gamma$ o bien $\delta' = \delta\gamma$. Si $\delta = \delta'\gamma'$ entonces $A[\delta]$ y $B[\delta'\gamma']$ se refieren al mismo elemento de la pila del autómata, en otro caso $A[\delta]$ es el elemento de la pila del autómata más próximo a la cima tal que su pila asociada es δ. La pila $\xi\ A[\delta]$ no debe ser alterada en ningún momento, aunque se permite la consulta de A en la subderivación d_1. La pila $\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']$ no debe ser alterada durante las subderivaciones d_2, d_3 y d_4, aunque se permite la consulta de B y de γ' en la subderivación d_2. La pila $\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']\ \xi_2\ D[\delta''\eta]$ no debe ser alterada durante la subderivación d_3, aunque se permite la consulta de D y de η. La pila de índices β no debe ser modificada en la derivación d_4. Sean $\delta = \delta_1 \ldots \delta_z$ y $\beta = \beta_1 \ldots \beta_z$, donde $\delta_i, \beta_i \in V_I$. Entonces debe cumplirse que $\forall i, \delta_i = \beta_i$, esto es, el contenido de δ y β es el mismo pero ambas pilas son *distintas* pues mientras δ ha sido construida (y posteriormente vaciada) en la fase descendente, la pila β ha sido construida en la fase ascendente a partir de una pila y en lo que resta de la fase ascendente será vaciada. En la figura 8.4 se muestra una representación gráfica de este tipo de transiciones

Para cualquier pila $\xi' \in (P[V_I^*])^*$ y pilas de índices $\alpha, \beta' \in V_I^*$ tal que existe una derivación $(D[\alpha\gamma], a_{p+1}\ldots a_n) \overset{*}{\vdash} (D[\alpha\gamma]\ E[\beta'], a_{q+1}\ldots a_n)$ se cumple

$$\begin{array}{rl}(\xi\ A[\alpha], a_{h+1}\ldots a_n) & \overset{*}{\vdash}_{d_1} (\xi\ A[\alpha]\ \xi_1\ B[\alpha'\gamma'], a_{i+1}\ldots a_n) \\ & \overset{*}{\vdash}_{d_2} (\xi\ A[\alpha]\ \xi_1\ B[\alpha'\gamma']\ \xi_2\ D[\alpha\gamma], a_{p+1}\ldots a_n) \\ & \overset{*}{\vdash}_{d_3} (\xi\ A[\alpha]\ \xi_1\ B[\alpha'\gamma']\ \xi_2\ D[\alpha\gamma]\ E[\beta'], a_{q+1}\ldots a_n) \\ & \overset{*}{\vdash}_{d_4} (\xi\ A[\alpha]\ \xi_1\ B[\alpha'\gamma']\ C[\beta'\gamma], a_{j+1}\ldots a_n)\end{array}$$

por lo que este tipo de derivaciones puede representarse de forma condensada mediante ítems de la forma

$$[A, h \mid B, i, \gamma', C, j, \gamma \mid D, p, E, q]$$

donde los componentes (A, h) y (D, p, E, q) garantizan que estamos considerando el mismo δ en toda la derivación.

Derivaciones de puntos especiales. Corresponden a la creación de una nueva pila de índices o bien a la terminación de una pila de índices e implican la existencia de la siguiente secuencia de subderivaciones.

$$(\xi\ B[\delta'\gamma'], a_{i+1}\ldots a_n) \ \overset{*}{\vdash}_d (\xi\ B[\delta'\gamma']\ C[\,], a_{j+1}\ldots a_n)$$

La pila $\xi\ B[\delta'\gamma']$ no debe ser alterada en ningún momento, aunque se permite la consulta de B. En la figura 8.5 se muestra una representación gráfica de este tipo de transiciones.

Para cualquier $\xi' \in (P[V_I^*])^*$ y $\alpha' \in V_I^*$ se cumple

$$(\xi'\ B[\alpha'\gamma'], a_{i+1}\ldots a_n) \ \overset{*}{\vdash}_d (\xi'\ B[\alpha'\gamma']\ C[\,], a_{j+1}\ldots a_n)$$

por lo que se pueden utilizar ítems de la forma

$$[-,- \mid B,i,\gamma',C,j,- \mid -,-,-,-]$$

para representar este tipo de derivaciones.

En la tabla 8.32 se muestran las reglas de combinaciones de ítems para las transiciones de la tabla 8.31, siempre considerando la restricción de que las pilas de índices deben ser construidas durante la fase descendente mediante la aplicación en orden inverso de las operaciones de apilamiento y extracción realizadas durante la fase de llamada. El ítem inicial es de la forma

$$[-,- \mid -,0,-,\$_0,0,- \mid -,-,-,-]$$

mientras que los ítems finales son de la forma

$$[-,- \mid \$_0,0,-,\$_f,n,- \mid -,-,-,-]$$

En las figuras 8.6 y 8.7 se muestra una representación gráfica de los antecedentes involucrados en las reglas de compilación correspondiente a las transiciones $B[\circ\circ_1]\ C[\circ\circ_2\gamma'] \longmapsto F[\circ\circ_2]$ y $B[\circ\circ_1\gamma]\ C[\circ\circ_2] \longmapsto F[\circ\circ_2\gamma]$, respectivamente.

Teorema 8.3 *La manipulación de configuraciones mediante la aplicación de transiciones en los autómatas a pila restringidos con estrategias $*$-Earley es equivalente a la manipulación de ítems mediante las reglas de combinación de la tabla 8.32.*

Demostración:

Puesto que un ítem representa una derivación y toda derivación debe ser representada por algún ítem, es suficiente con demostrar que la combinación de los ítems produce ítems que se corresponden con derivaciones válidas y que para toda derivación que se pueda producir como resultado de la aplicación de una transición, existe una regla de combinación de ítems que produce un ítem que representa a dicha derivación. A continuación se muestra una lista de todos los casos posibles de derivación que se puedan dar junto con la correspondiente regla de combinación de ítems.

- Derivaciones que son el resultado de aplicar una transición $C[\circ\circ] \longmapsto C[\circ\circ]\ F[\,]$
  - a una derivación de llamada:

$$\begin{array}{ll}(\xi\ A[\delta], a_{h+1}\ldots a_n) & \overset{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma'], a_{i+1}\ldots a_n) \\ & \overset{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']\ C[\delta\gamma], a_{j+1}\ldots a_n) \\ & \vdash (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']\ C[\delta\gamma]\ F[\,], a_{j+1}\ldots a_n)\end{array}$$

$$\frac{[A,h \mid B,i,\gamma',C,j,\gamma \mid -,-,-,-]}{[-,- \mid C,j,\gamma,F,j,- \mid -,-,-,-]}\ C[\circ\circ] \longmapsto C[\circ\circ]\ F[\,]$$

  - a una derivación de retorno:

$$\begin{array}{ll}(\xi\ A[\delta], a_{h+1}\ldots a_n) & \overset{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma'], a_{i+1}\ldots a_n) \\ & \overset{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']\ \xi_2\ D[\delta\gamma], a_{p+1}\ldots a_n) \\ & \overset{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']\ \xi_2\ D[\delta\gamma]\ E[\beta], a_{q+1}\ldots a_n) \\ & \overset{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']\ C[\beta\gamma], a_{j+1}\ldots a_n) \\ & \vdash (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']\ C[\beta\gamma]\ F[\,], a_{j+1}\ldots a_n)\end{array}$$

$$\frac{[A,h \mid B,i,\gamma',C,j,\gamma \mid D,p,E,q]}{[-,- \mid C,j,\gamma,F,j,- \mid -,-,-,-]}\ C[\circ\circ] \longmapsto C[\circ\circ]\ F[\,]$$

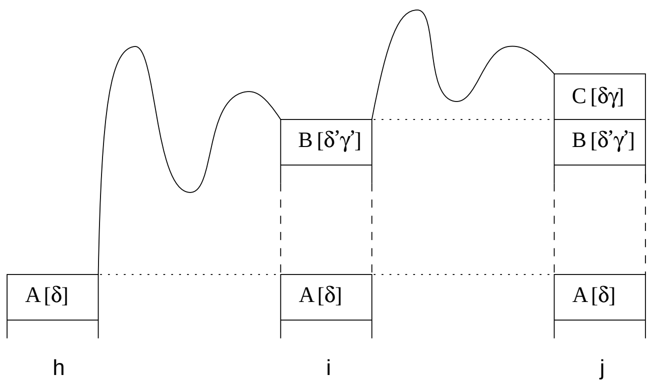

Figura 8.3: Derivaciones de llamada en estrategias $*$-Earley

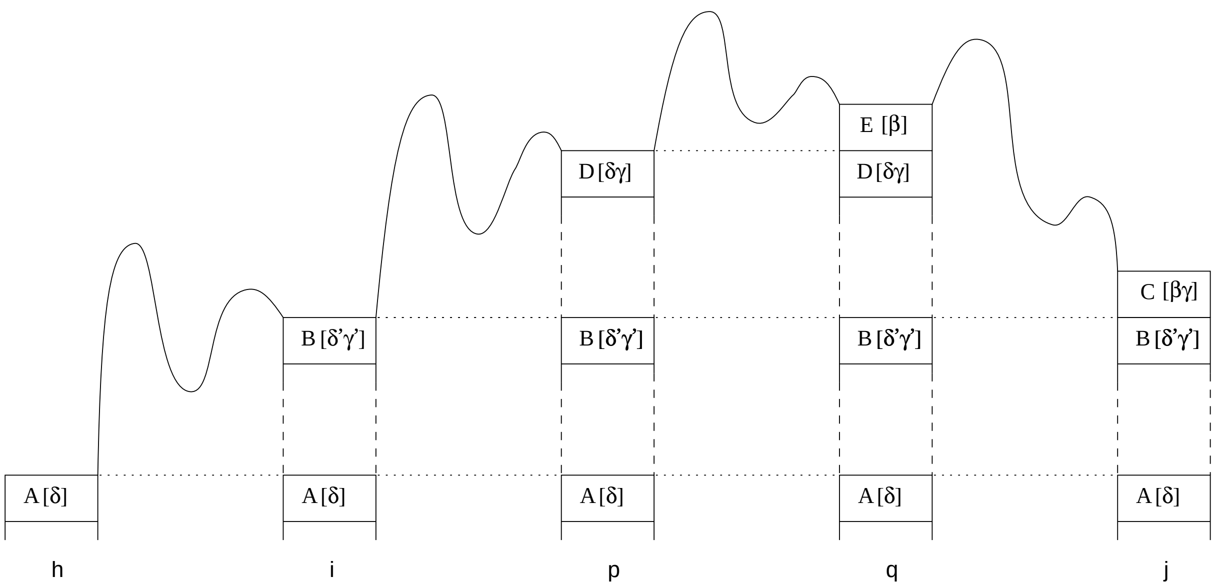

Figura 8.4: Derivaciones de retorno en estrategias $*$-Earley

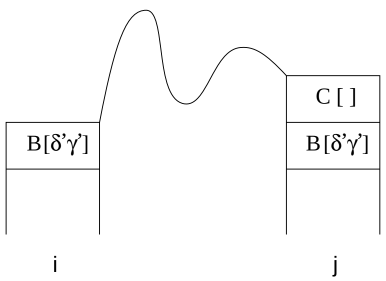

Figura 8.5: Derivaciones de puntos especiales en estrategias $*$-Earley

$$\frac{[A,h \mid B,i,\gamma',C,j,\gamma \mid D,p,E,q]}{[-,- \mid C,j,\gamma,F,j,- \mid -,-,-,-]} \quad C[\circ\circ] \longmapsto C[\circ\circ]\ F[\,]$$

$$\frac{[A,h \mid B,i,\gamma',C,j,\gamma \mid -,-,-,-]}{[A,h \mid C,j,\gamma,F,j,\gamma \mid -,-,-,-]} \quad C[\circ\circ] \longmapsto C[\circ\circ]\ F[\circ\circ]$$

$$\frac{[A,h \mid B,i,\gamma'',C,j,\gamma \mid -,-,-,-]}{[C,j \mid C,j,\gamma,F,j,\gamma' \mid -,-,-,-]} \quad C[\circ\circ] \longmapsto C[\circ\circ]\ F[\circ\circ\gamma']$$

$$\frac{\begin{array}{c}[A,h \mid B,i,\gamma'',C,j,\gamma \mid -,-,-,-]\\ [M,m \mid N,t,\gamma''',A,h,\gamma' \mid -,-,-,-]\end{array}}{[M,m \mid C,j,\gamma,F,j,\gamma' \mid -,-,-,-]} \quad C[\circ\circ\gamma] \longmapsto C[\circ\circ\gamma]\ F[\circ\circ]$$

$$\frac{[A,h \mid B,i,\gamma',C,j,\gamma \mid D,p,E,q]}{[A,h \mid B,i,\gamma',F,j,\gamma \mid D,p,E,q]} \quad C[\circ\circ] \longmapsto F[\circ\circ]$$

$$\frac{\begin{array}{c}[-,- \mid C,j,\gamma,F,k,- \mid -,-,-,-]\\ [A,h \mid B,i,\gamma',C,j,\gamma \mid D,p,E,q]\end{array}}{[A,h \mid B,i,\gamma',G,k,\gamma \mid D,p,E,q]} \quad C[\circ\circ]\ F[\,] \longmapsto G[\circ\circ]$$

$$\frac{\begin{array}{c}[A,h \mid C,j,\gamma,F,k,\gamma \mid D,p,E,q]\\ [A,h \mid B,i,\gamma',C,j,\gamma \mid -,-,-,-]\end{array}}{[A,h \mid B,i,\gamma',G,k,\gamma \mid D,p,E,q]} \quad C[\circ\circ_1]\ F[\circ\circ_2] \longmapsto G[\circ\circ_2]$$

$$\frac{\begin{array}{c}[C,i \mid C,i,\gamma,F,k,\gamma' \mid D,p,E,q]\\ [A,h \mid, B,i,\gamma'',C,i,\gamma \mid -,-,-,-]\\ [A,h \mid D,p,\gamma',E,q,\gamma \mid O,u,P,v]\end{array}}{[A,h \mid B,i,\gamma'',G,k,\gamma \mid O,u,P,v]} \quad C[\circ\circ_1]\ F[\circ\circ_2\gamma'] \longmapsto G[\circ\circ_2]$$

$$\frac{\begin{array}{c}[M,m \mid C,j,\gamma,F,k,\gamma' \mid D,p,E,q]\\ [A,h \mid B,i,\gamma'',C,j,\gamma \mid -,-,-,-]\\ [M,m \mid N,t,\gamma''',A,h,\gamma' \mid -,-,-,-]\end{array}}{[A,h \mid B,i,\gamma'',G,k,\gamma \mid C,j,F,k]} \quad C[\circ\circ_1\gamma]\ F[\circ\circ_2] \longmapsto G[\circ\circ_2\gamma]$$

$$\frac{[-,- \mid B,i,\gamma',C,j,- \mid -,-,-,-]}{[-,- \mid B,i,\gamma',F,k,- \mid -,-,-,-]} \quad C[\,] \overset{a}{\longmapsto} F[\,]$$

Tabla 8.32: Combinación de ítems en las estrategias $*$-Earley

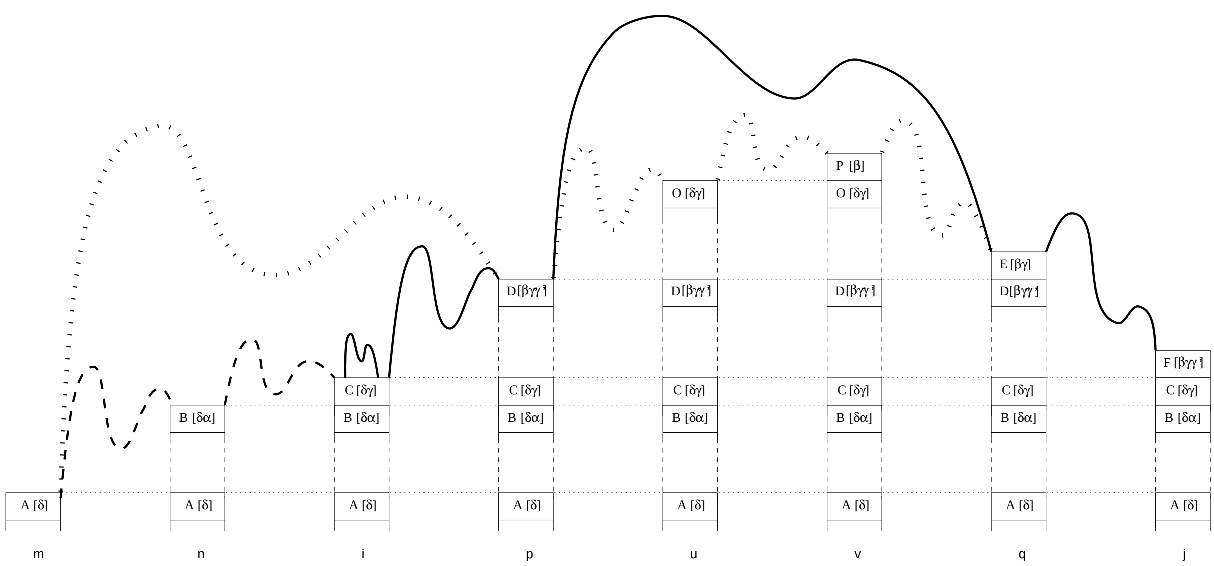

Figura 8.6: Regla de combinación para las transiciones $B[\circ\circ_1]\ C[\circ\circ_2\gamma'] \longmapsto F[\circ\circ_2]$

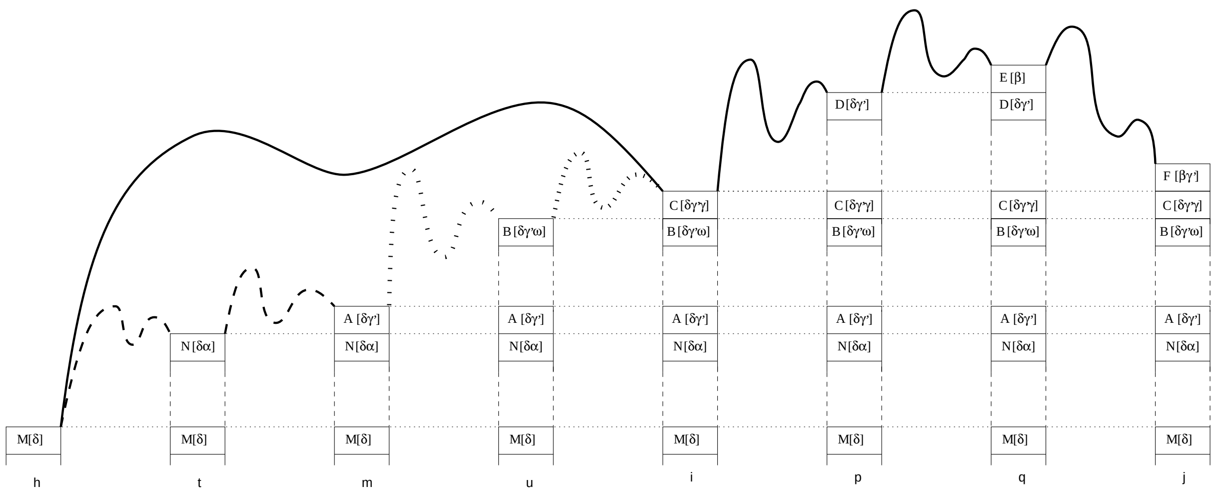

Figura 8.7: Regla de combinación para las transiciones $B[\circ\circ_1\gamma]\ C[\circ\circ_2] \longmapsto F[\circ\circ_2\gamma]$

– a una derivación de puntos especiales:

$$\begin{array}{ll}(\xi\ B[\delta'\gamma'], a_{i+1}\dots a_n) & \overset{*}{\vdash} (\xi\ B[\delta'\gamma']\ C[\,], a_{j+1}\dots a_n)\\ & \overset{*}{\vdash} (\xi\ B[\delta'\gamma']\ C[\,]\ F[\,], a_{j+1}\dots a_n)\end{array}$$

$$\frac{[-,-\mid B,i,\gamma',C,j,-\mid -,-,-,-]}{[-,-\mid C,j,\gamma',F,j,-\mid -,-,-,-]}\quad C[\circ\circ]\longmapsto C[\circ\circ]\ F[\,]$$

- Derivaciones que son el resultado de aplicar una transición $C[\circ\circ]\longmapsto C[\circ\circ]\ F[\circ\circ]$
  - a una derivación de llamada:

$$\begin{array}{ll}(\xi\ A[\delta], a_{h+1}\dots a_n) & \overset{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma'], a_{i+1}\dots a_n)\\ & \overset{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']\ C[\delta\gamma], a_{j+1}\dots a_n)\\ & \vdash (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']\ C[\delta\gamma]\ F[\delta\gamma], a_{j+1}\dots a_n)\end{array}$$

$$\frac{[A,h\mid B,i,\gamma',C,j,\gamma\mid -,-,-,-]}{[A,h\mid C,j,\gamma,F,j,\gamma\mid -,-,-,-]}\quad C[\circ\circ]\longmapsto C[\circ\circ]\ F[\circ\circ]$$

  - a una derivación de puntos especiales:

$$\begin{array}{ll}(\xi\ B[\delta'\gamma'], a_{i+1}\dots a_n) & \overset{*}{\vdash} (\xi\ B[\delta'\gamma']\ C[\,], a_{j+1}\dots a_n)\\ & \overset{*}{\vdash} (\xi\ B[\delta'\gamma']\ C[\,]\ F[\,], a_{j+1}\dots a_n)\end{array}$$

$$\frac{[-,-\mid B,i,\gamma',C,j,-\mid -,-,-,-]}{[-,-\mid C,j,-,F,j,-\mid -,-,-,-]}\quad C[\circ\circ]\longmapsto C[\circ\circ]\ F[\circ\circ]$$

- Derivaciones que son el resultado de aplicar una transición $C[\circ\circ]\longmapsto C[\circ\circ]\ F[\circ\circ\gamma']$
  - a una derivación de llamada:

$$\begin{array}{ll}(\xi\ A[\delta], a_{h+1}\dots a_n) & \overset{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma''], a_{i+1}\dots a_n)\\ & \overset{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma'']\ C[\delta\gamma], a_{j+1}\dots a_n)\\ & \vdash (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma'']\ C[\delta\gamma]\ F[\delta\gamma\gamma'], a_{j+1}\dots a_n)\end{array}$$

$$\frac{[A,h\mid B,i,\delta'',C,j,\gamma\mid -,-,-,-]}{[C,j\mid C,j,\gamma,F,j,\gamma'\mid -,-,-,-]}\quad C[\circ\circ]\longmapsto C[\circ\circ]\ F[\circ\circ\gamma']$$

  - a una derivación de puntos especiales:

$$\begin{array}{ll}(\xi\ B[\delta'\gamma''], a_{i+1}\dots a_n) & \overset{*}{\vdash} (\xi\ B[\delta'\gamma'']\ C[\,], a_{j+1}\dots a_n)\\ & \overset{*}{\vdash} (\xi\ B[\delta'\gamma'']\ C[\,]\ F[\gamma'], a_{j+1}\dots a_n)\end{array}$$

$$\frac{[-,-\mid B,i,\gamma'',C,j,-\mid -,-,-,-]}{[C,j\mid C,j,-,F,j,\gamma'\mid -,-,-,-]}\quad C[\circ\circ]\longmapsto C[\circ\circ]\ F[\circ\circ\gamma']$$

- Derivaciones que son el resultado de aplicar una transición $C[\circ\circ\gamma]\longmapsto C[\circ\circ\gamma]\ F[\circ\circ]$ a una derivación de llamada:

$$\begin{array}{ll}(\xi\ M[\delta], a_{h+1}\dots a_n) & \overset{*}{\vdash} (\xi\ M[\delta]\ \xi_1\ N[\delta''\gamma'''], a_{h+1}\dots a_n)\\ & \overset{*}{\vdash} (\xi\ M[\delta]\ \xi_1\ N[\delta''\gamma''']\ A[\delta\gamma'], a_{h+1}\dots a_n)\\ & \overset{*}{\vdash} (\xi\ M[\delta]\ \xi_1\ N[\delta''\gamma''']\ A[\delta\gamma']\ \xi_2\ B[\delta'\gamma''], a_{i+1}\dots a_n)\\ & \overset{*}{\vdash} (\xi\ M[\delta]\ \xi_1\ N[\delta''\gamma''']\ A[\delta\gamma']\ \xi_2\ B[\delta'\gamma'']\ C[\delta\gamma'\gamma], a_{j+1}\dots a_n)\\ & \vdash (\xi\ M[\delta]\ \xi_1\ N[\delta''\gamma''']\ A[\delta\gamma']\ \xi_1\ B[\delta'\gamma'']\ C[\delta\gamma'\gamma]\ F[\delta\gamma'], a_{j+1}\dots a_n)\end{array}$$

$$\frac{\begin{array}{c}[A,h\mid B,i,\gamma'',C,j,\gamma\mid -,-,-,-]\\ [M,m\mid N,t,\gamma''',A,h,\gamma'\mid -,-,-,-]\end{array}}{[M,m\mid C,j,\gamma,F,j,\gamma'\mid -,-,-,-]}\quad C[\circ\circ\gamma]\longmapsto C[\circ\circ\gamma]\ F[\circ\circ]$$

- Derivaciones que son el resultado de aplicar una transición $C[\circ\circ] \longmapsto F[\circ\circ]$
  - a una derivación de llamada:

$$\begin{array}{ll}(\xi\ A[\delta], a_{h+1}\dots a_n) & \overset{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma'], a_{i+1}\dots a_n) \\ & \overset{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']\ C[\delta\gamma], a_{j+1}\dots a_n) \\ & \vdash (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']\ F[\delta\gamma], a_{j+1}\dots a_n)\end{array}$$

$$\frac{[A,h \mid B,i,\gamma',C,j,\gamma \mid -,-,-,-]}{[A,h \mid B,i,\gamma',F,j,\gamma \mid -,-,-,-]}\ C[\circ\circ] \longmapsto F[\circ\circ]$$

  - a una derivación de retorno:

$$\begin{array}{ll}(\xi\ A[\delta], a_{h+1}\dots a_n) & \overset{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma'], a_{i+1}\dots a_n) \\ & \overset{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']\ \xi_2\ D[\delta\gamma], a_{p+1}\dots a_n) \\ & \overset{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']\ \xi_2\ D[\delta\gamma]\ E[\beta], a_{q+1}\dots a_n) \\ & \overset{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']\ C[\beta\gamma], a_{j+1}\dots a_n) \\ & \vdash (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']\ F[\beta\gamma], a_{j+1}\dots a_n)\end{array}$$

$$\frac{[A,h \mid B,i,\gamma',C,j,\gamma \mid D,p,E,q]}{[A,h \mid B,i,\gamma',F,j,\gamma \mid D,p,E,q]}\ C[\circ\circ] \longmapsto F[\circ\circ]$$

  - a una derivación de puntos especiales:

$$\begin{array}{ll}(\xi\ B[\delta'\gamma'], a_{i+1}\dots a_n) & \overset{*}{\vdash} (\xi\ B[\delta'\gamma']\ C[\,], a_{j+1}\dots a_n) \\ & \overset{*}{\vdash} (\xi\ B[\delta'\gamma']\ F[\,], a_{j+1}\dots a_n)\end{array}$$

$$\frac{[-,- \mid B,i,\gamma',C,j,- \mid -,-,-,-]}{[-,- \mid B,i,\gamma',F,j,- \mid -,-,-,-]}\ C[\circ\circ] \longmapsto F[\circ\circ]$$

- Derivaciones que son el resultado de aplicar una transición $C[\circ\circ]\ F[\,] \longmapsto G[\circ\circ]$ obtenida tras aplicar una transición $C[\circ\circ] \longmapsto C[\,]\ F'[\circ\circ]$
  - a una derivación de llamada:

$$\begin{array}{ll}(\xi\ A[\delta], a_{h+1}\dots a_n) & \overset{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma'], a_{i+1}\dots a_n) \\ & \overset{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']\ C[\delta\gamma], a_{j+1}\dots a_n) \\ & \vdash (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']\ C[\delta\gamma]\ F'[\,], a_{j+1}\dots a_n) \\ & \overset{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']\ C[\delta\gamma]\ F[\,], a_{k+1}\dots a_n) \\ & \vdash (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']\ G[\delta\gamma], a_{k+1}\dots a_n)\end{array}$$

$$\frac{\begin{array}{c}[-,- \mid C,j,\gamma,F,k,- \mid -,-,-,-] \\ [A,h \mid B,i,\gamma',C,j,\gamma \mid -,-,-,-]\end{array}}{[A,h \mid B,i,\gamma',G,k,\gamma \mid -,-,-,-]}\ C[\circ\circ]\ F[\,] \longmapsto G[\circ\circ]$$

  - a una derivación de retorno:

$$\begin{array}{ll}(\xi\ A[\delta], a_{h+1}\dots a_n) & \overset{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma'], a_{i+1}\dots a_n) \\ & \overset{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']\ \xi_2\ D[\delta\gamma], a_{p+1}\dots a_n) \\ & \overset{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']\ \xi_2\ D[\delta\gamma]\ E[\beta], a_{q+1}\dots a_n) \\ & \overset{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']\ C[\beta\gamma], a_{j+1}\dots a_n) \\ & \vdash (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']\ C[\beta\gamma]\ F'[\,], a_{j+1}\dots a_n) \\ & \overset{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']\ C[\beta\gamma]\ F[\,], a_{k+1}\dots a_n) \\ & \vdash (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']\ G[\beta\gamma], a_{j+1}\dots a_n)\end{array}$$

$$\frac{\begin{array}{c}[-,- \mid C,j,\gamma,F,k,- \mid -,-,-,-] \\ [A,h \mid B,i,\gamma',C,j,\gamma \mid D,p,E,q]\end{array}}{[A,h \mid B,i,\gamma',G,k,\gamma \mid D,p,E,q]}\ C[\circ\circ]\ F[\,] \longmapsto G[\circ\circ]$$

– a una derivación de puntos especiales:

$$\begin{array}{ll}(\xi\ B[\delta'\gamma'], a_{i+1}\dots a_n) & \overset{*}{\vdash} (\xi\ B[\delta'\gamma']\ C[\,], a_{j+1}\dots a_n)\\ & \vdash (\xi\ B[\delta'\gamma']\ C[\,]\ F'[\,], a_{j+1}\dots a_n)\\ & \overset{*}{\vdash} (\xi\ B[\delta'\gamma']\ C[\,]\ F[\,], a_{k+1}\dots a_n)\\ & \vdash (\xi\ B[\delta'\gamma']\ G[\,], a_{k+1}\dots a_n)\end{array}$$

$$\frac{\begin{array}{c}[-,-\mid C,j,\gamma,F,k,-\mid -,-,-,-]\\ [-,-\mid B,i,\gamma',C,j,-\mid -,-,-,-]\end{array}}{[-,-\mid B,i,\gamma',G,k,-\mid -,-,-,-]}\quad C[\circ\circ]\ F[\,]\longmapsto G[\circ\circ]$$

- Derivaciones que son el resultado de aplicar una transición $C[\circ\circ_1]\ F[\circ\circ_2]\longmapsto G[\circ\circ_2]$
  - a una derivación de retorno obtenida tras aplicar una transición $C[\circ\circ]\longmapsto C[\circ\circ]\ F'[\circ\circ]$ a una derivación de llamada:

$$\begin{array}{ll}(\xi\ A[\delta], a_{h+1}\dots a_n) & \overset{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma'], a_{i+1}\dots a_n)\\ & \overset{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']\ C[\delta\gamma], a_{j+1}\dots a_n)\\ & \vdash (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']\ C[\delta\gamma]\ F'[\delta\gamma], a_{j+1}\dots a_n)\\ & \overset{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']\ C[\delta\gamma]\ F'[\delta\gamma]\ \xi_2\ D[\delta\gamma], a_{p+1}\dots a_n)\\ & \overset{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']\ C[\delta\gamma]\ F'[\delta\gamma]\ \xi_2\ D[\delta\gamma]\ E[\beta], a_{q+1}\dots a_n)\\ & \overset{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']\ C[\delta\gamma]\ F[\beta\gamma], a_{k+1}\dots a_n)\\ & \vdash (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']\ G[\beta\gamma], a_{j+1}\dots a_n)\end{array}$$

$$\frac{\begin{array}{c}[A,h\mid C,j,\gamma,F,k,\gamma\mid D,p,E,q]\\ [A,h\mid B,i,\gamma',C,j,\gamma\mid -,-,-,-]\end{array}}{[A,h\mid B,i,\gamma',G,k,\gamma\mid D,p,E,q]}\quad C[\circ\circ_1]\ F[\circ\circ_2]\longmapsto G[\circ\circ_2]$$

  - a una derivación de puntos especiales obtenida tras aplicar una transición $C[\circ\circ]\longmapsto C[\circ\circ]\ F'[\circ\circ]$ a una derivación de puntos especiales:

$$\begin{array}{ll}(\xi\ B[\delta'\gamma'], a_{i+1}\dots a_n) & \overset{*}{\vdash} (\xi\ B[\delta'\gamma']\ C[\,], a_{j+1}\dots a_n)\\ & \vdash (\xi\ B[\delta'\gamma']\ C[\,]\ F'[\,], a_{j+1}\dots a_n)\\ & \overset{*}{\vdash} (\xi\ B[\delta'\gamma']\ C[\,]\ F[\,], a_{k+1}\dots a_n)\\ & \vdash (\xi\ B[\delta'\gamma']\ G[\,], a_{k+1}\dots a_n)\end{array}$$

$$\frac{\begin{array}{c}[-,-\mid C,j,\gamma,F,k,-\mid -,-,-,-]\\ [-,-\mid B,i,\gamma',C,j,-\mid -,-,-,-]\end{array}}{[-,-\mid B,i,\gamma',G,k,\gamma\mid -,-,-,-]}\quad C[\circ\circ_1]\ F[\circ\circ_2]\longmapsto G[\circ\circ_2]$$

- Derivaciones que son el resultado de aplicar una transición $C[\circ\circ_1]\ F[\circ\circ_2\gamma']\longmapsto G[\circ\circ_2]$ a una derivación de retorno obtenida tras aplicar una transición $C[\circ\circ\gamma]\longmapsto C[\circ\circ]\ F'[\circ\circ\gamma']$
  - a una derivación de llamada:

$$\begin{array}{l}(\xi\ A[\delta], a_{h+1}\dots a_n)\\ \overset{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma''], a_{i+1}\dots a_n)\\ \overset{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma'']\ C[\delta\gamma], a_{j+1}\dots a_n)\\ \vdash (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma'']\ C[\delta\gamma]\ F'[\delta\gamma\gamma'], a_{j+1}\dots a_n)\\ \overset{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma'']\ C[\delta\gamma]\ F'[\delta\gamma\gamma']\ \xi_1\ D[\delta\gamma\gamma'], a_{p+1}\dots a_n)\\ \overset{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma'']\ C[\delta\gamma]\ F'[\delta\gamma\gamma']\ \xi_1\ D[\delta\gamma\gamma']\ \xi_2\ O[\delta\gamma], a_{u+1}\dots a_n)\\ \overset{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma'']\ C[\delta\gamma]\ F'[\delta\gamma\gamma']\ \xi_1\ D[\delta\gamma\gamma']\ \xi_2\ O[\delta\gamma]\ P[\beta], a_{v+1}\dots a_n)\\ \overset{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma'']\ C[\delta\gamma]\ F'[\delta\gamma\gamma']\ \xi_1\ D[\delta\gamma\gamma']\ E[\beta\gamma], a_{q+1}\dots a_n)\\ \overset{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma'']\ C[\delta\gamma]\ F[\beta\gamma\gamma'], a_{k+1}\dots a_n)\\ \vdash (\xi\ A[\delta]\ \xi_1\ B[\delta\gamma'']\ G[\beta\gamma], a_{k+1}\dots a_n)\end{array}$$

$$\frac{\begin{array}{c}[C,i \mid C,i,\gamma,F,k,\gamma' \mid D,p,E,q] \\ [A,h \mid, B,i,\gamma'',C,i,\gamma \mid -,-,-,-] \\ [A,h \mid D,p,\gamma',E,q,\gamma \mid O,u,P,v]\end{array}}{[A,h \mid B,i,\gamma'',G,k,\gamma \mid O,u,P,v]} \quad C[\circ\circ_1]\ F[\circ\circ_2\gamma'] \longmapsto G[\circ\circ_2]$$

– a una derivación de puntos especiales:

$$\begin{array}{ll}(\xi\ B[\delta'\gamma''], a_{i+1}\ldots a_n) & \overset{*}{\vdash} (\xi\ B[\delta'\gamma'']\ C[\,], a_{j+1}\ldots a_n) \\ & \overset{*}{\vdash} (\xi\ B[\delta'\gamma'']\ C[\,]\ F'[\gamma'], a_{j+1}\ldots a_n) \\ & \overset{*}{\vdash} (\xi\ B[\delta'\gamma'']\ C[\,]\ F'[\gamma']\ \xi_1\ D[\gamma'], a_{p+1}\ldots a_n) \\ & \overset{*}{\vdash} (\xi\ B[\delta'\gamma'']\ C[\,]\ F'[\gamma']\ \xi_1\ D[\gamma']\ E[\,], a_{q+1}\ldots a_n) \\ & \overset{*}{\vdash} (\xi\ B[\delta'\gamma'']\ C[\,]\ F[\gamma'], a_{k+1}\ldots a_n) \\ & \overset{*}{\vdash} (\xi\ B[\delta'\gamma'']\ G[\,], a_{k+1}\ldots a_n)\end{array}$$

$$\frac{\begin{array}{c}[C,i \mid C,i,-,F,k,\gamma' \mid D,p,E,q] \\ [-,- \mid, B,i,\gamma'',C,i,- \mid -,-,-,-] \\ [-,- \mid D,p,\gamma',E,q,- \mid -,-,-,-]\end{array}}{[-,- \mid B,i,\gamma'',G,k,- \mid -,-,-,-]} \quad C[\circ\circ_1]\ F[\circ\circ_2\gamma'] \longmapsto G[\circ\circ_2]$$

- Derivaciones que son el resultado de aplicar una transición $C[\circ\circ_1\gamma]\ F[\circ\circ_2] \longmapsto G[\circ\circ_2\gamma]$ a una derivación de retorno obtenida tras aplicar una transición $C[\circ\circ\gamma] \longmapsto C[\circ\circ\gamma]\ F'[\circ\circ]$ a una derivación de llamada:

$$\begin{array}{l}(\xi\ M[\delta''\gamma'''], a_{h+1}\ldots a_n) \\ \overset{*}{\vdash} (\xi\ M[\delta''\gamma''']\ \xi_1\ N[\delta\alpha_1], a_{h+1}\ldots a_n) \\ \overset{*}{\vdash} (\xi\ M[\delta''\gamma''']\ \xi_1\ N[\delta\alpha_1]\ A[\delta\gamma'], a_{h+1}\ldots a_n) \\ \overset{*}{\vdash} (\xi\ M[\delta''\gamma''']\ \xi_1\ N[\delta\alpha_1]\ A[\delta\gamma']\ \xi_2\ B[\delta'\gamma''], a_{i+1}\ldots a_n) \\ \overset{*}{\vdash} (\xi\ M[\delta''\gamma''']\ \xi_1\ N[\delta\alpha_1]\ A[\delta\gamma']\ \xi_2\ B[\delta'\gamma'']\ C[\delta\gamma'\gamma], a_{j+1}\ldots a_n) \\ \vdash (\xi\ M[\delta''\gamma''']\ \xi_1\ N[\delta\alpha_1]\ A[\delta\gamma']\ \xi_1\ B[\delta'\gamma'']\ C[\delta\gamma'\gamma]\ F'[\delta\gamma'], a_{j+1}\ldots a_n) \\ \overset{*}{\vdash} (\xi\ M[\delta''\gamma''']\ \xi_1\ N[\delta\alpha_1]\ A[\delta\gamma']\ \xi_1\ B[\delta'\gamma'']\ C[\delta\gamma'\gamma]\ F'[\delta\gamma']\ \xi_2\ D[\delta\gamma'], a_{p+1}\ldots a_n) \\ \overset{*}{\vdash} (\xi\ M[\delta''\gamma''']\ \xi_1\ N[\delta\alpha_1]\ A[\delta\gamma']\ \xi_1\ B[\delta'\gamma'']\ C[\delta\gamma'\gamma]\ F'[\delta\gamma']\ \xi_2\ D[\delta\gamma']\ E[\beta], a_{q+1}\ldots a_n) \\ \overset{*}{\vdash} (\xi\ M[\delta''\gamma''']\ \xi_1\ N[\delta\alpha_1]\ A[\delta\gamma']\ \xi_1\ B[\delta'\gamma'']\ C[\delta\gamma'\gamma]\ F[\beta\gamma'], a_{k+1}\ldots a_n) \\ \vdash (\xi\ M[\delta''\gamma''']\ \xi_1\ N[\delta\alpha_1]\ A[\delta\gamma']\ \xi_1\ B[\delta'\gamma'']\ G[\beta\gamma'\gamma], a_{k+1}\ldots a_n)\end{array}$$

$$\frac{\begin{array}{c}[M,m \mid C,j,\gamma,F,k,\gamma' \mid D,p,E,q] \\ [A,h \mid B,i,\gamma'',C,j,\gamma \mid -,-,-,-] \\ [M,m \mid N,t,\gamma'',A,h,\gamma' \mid -,-,-,-]\end{array}}{[A,h \mid B,i,\gamma'',G,k,\gamma \mid C,j,F,k]} \quad C[\circ\circ_1\gamma]\ F[\circ\circ_2] \longmapsto G[\circ\circ_2\gamma]$$

- Derivaciones que son el resultado de aplicar una transición $C[\,] \overset{a}{\longmapsto} F[\,]$ a una derivación de puntos especiales:

$$\begin{array}{ll}(\xi\ B[\delta'\gamma'], a_{i+1}\ldots a_n) & \overset{*}{\vdash} (\xi\ B[\delta'\gamma']\ C[\,], a_{j+1}\ldots a_n) \\ & \overset{*}{\vdash} (\xi\ B[\delta'\gamma']\ F[\,], a_{k+1}\ldots a_n)\end{array}$$

$$\frac{[-,- \mid B,i,\gamma',C,j,- \mid -,-,-,-]}{[-,- \mid B,i,\gamma',F,k,- \mid -,-,-,-]} \quad C[\,] \overset{a}{\longmapsto} F[\,]$$

donde $k = j$ si $a = \epsilon$ y $k = j + 1$ si $a = a_{j+1}$.

Si aplicamos inducción en la longitud de una derivación, a partir de la lista observamos que para cualquier derivación obtenida mediante la aplicación de una transición, existe una regla de combinación que busca los ítems correspondientes a las subderivaciones relevantes y que produce el ítem correspondiente a la derivación resultante. También podemos observar que dada una regla de combinación de ítems, existen subderivaciones en el autómata que se corresponden con cada uno de los ítems antecedentes y que combinadas entre sí producen la derivación correspondiente al ítem consecuente. □

La complejidad temporal de esta técnica de tabulación, con respecto a la longitud n de la cadena de entrada, es $\mathcal{O}(n^7)$. Este incremento de complejidad con respecto a las técnicas de tabulación para estrategias $*$-ascendentes es debido a la regla de combinación

$$\frac{\begin{array}{l}[B,i \mid B,i,C,j,\gamma' \mid D,p,E,q]\\ [M,m \mid, N,t,B,i,\gamma \mid -,-,-,-]\\ [M,m \mid D,p,E,q,\gamma \mid O,u,P,v]\end{array}}{[M,m \mid N,t,F,k,\gamma \mid O,u,P,v]} \quad B[\circ\circ_1]\ C[\circ\circ_2\gamma'] \longmapsto F[\circ\circ_2]$$

que involucra la combinación de 8 posiciones de la cadena de entrada, aunque sólo es preciso utilizar 7 de ellas conjuntamente. Para reducir la complejidad podemos utilizar la técnica propuesta en [53, 125], consistente en dividir la regla mencionada en dos reglas de menor complejidad, de tal modo que la primera genere un pseudo-ítem intermedio que proporcione la información relevante para la segunda. En este caso en particular, se trata de repartir la información proporcionada por el ítem $[M,m \mid N,t,F,k,\gamma \mid O,u,P,v]$ entre las dos nuevas reglas:

$$\frac{\begin{array}{l}[B,i \mid B,i,C,j,\gamma' \mid D,p,E,q]\\ [M,m \mid D,p,E,q,\gamma \mid O,u,P,v]\end{array}}{[[B,i,C,j,\gamma' \mid O,u,P,v]]} \quad B[\circ\circ_1]\ C[\circ\circ_2\gamma'] \overset{F}{\longmapsto} [\circ\circ_2]$$

$$\frac{\begin{array}{l}[[B,i,C,j,\gamma' \mid O,u,P,v]]\\ [M,m \mid, N,t,B,i,\gamma \mid -,-,-,-]\\ [M,m \mid D,p,E,q,\gamma \mid O,u,P,v]\end{array}}{[M,m \mid N,t,F,j,\gamma \mid O,u,P,v]} \quad B[\circ\circ_1]\ C[\circ\circ_2\gamma'] \longmapsto F[\circ\circ_2]$$

donde $[[B,i,C,j,\gamma' \mid O,u,P,v]]$ es el pseudo-ítem. La diferencia con una aplicación parcial es que la primera regla ignora los elementos M y m en el pseudo-ítem generado, puesto que tales elementos son posteriormente recuperados del segundo y tercer ítem que intervienen en la segunda regla, que junto con el pseudo-ítem son suficientes para garantizar la existencia del ítem $[B,i \mid B,i,C,j,\gamma' \mid D,p,E,q]$ por la definición de derivaciones de llamada y retorno. La primera regla presenta una complejidad temporal $\mathcal{O}(n^6)$ (la posición m no interviene) y la segunda presenta también una complejidad $\mathcal{O}(n^6)$ (las posiciones p y q no intervienen) por lo que hemos logrado rebajar la complejidad temporal final de la técnica de tabulación a $\mathcal{O}(n^6)$.

La complejidad espacial con respecto a la longitud de la cadena de entrada es $\mathcal{O}(n^5)$ puesto que cada ítem almacena cinco posiciones con respecto a la cadena de entrada.

8.5.4 Tabulación de estrategias $*$-descendentes

Las estrategias de tipo $*$-descendentes se caracterizan porque las pilas de índices se construyen durante la fase de llamada y no son propagadas durante la fase de retorno. Esto quiere decir que todas las restricciones dependientes del contexto impuestas por los índices son comprobadas en la fase de llamada. Como consecuencia, las técnicas tabulares son más complejas que en el caso de estrategias $*$-ascendentes, puesto que debemos comprobar ciertas operaciones realizadas en la fase de llamada para garantizar que la fase de retorno actúa correctamente.

Las transiciones mostradas en la tabla 8.33 se corresponden con los tipos de transiciones utilizadas en los esquemas de compilación de LIG y TAG que incorporan estrategias $*$-descendentes,

Transición	Compilación de LIG	Compilación de TAG
$C[\circ\circ] \longmapsto C[\circ\circ]\ F[\,]$	**[INIT][CALL]**	**[INIT][CALL]**
$C[\circ\circ\gamma] \longmapsto C[\circ\circ\gamma]\ F[\circ\circ\gamma']$	**[SCALL]**	**[SCALL][ACALL][FCALL]**
$C[\circ\circ] \overset{a}{\longmapsto} F[\circ\circ]$	**[SEL][PUB][SCAN]**	**[SEL][PUB][SCAN]**
$C[\circ\circ]\ F[\,] \longmapsto G[\circ\circ]$	**[RET]**	**[RET]**
$C[\circ\circ]\ F[\,] \longmapsto G[\,]$	**[SRET]**	**[SRET][ARET][FRET]**

Tabla 8.33: Tipos de transiciones en las estrategias $*$-descendente

con la salvedad de las transiciones de la forma $B[\circ\circ]\ C[\,] \longmapsto F[\circ\circ]$, que constituyen una generalización de las transiciones $B[\circ\circ\gamma]\ C[\,] \longmapsto F[\circ\circ\gamma]$ utilizadas en tales esquemas. Con las transiciones de la tabla 8.33 es posible construir autómatas a pila restringidos que aceptan lenguajes que no pertenecen a la clase de los lenguajes de adjunción de árboles. Ello se debe a que la utilización combinada de transiciones $C[\circ\circ\gamma] \longmapsto C[\circ\circ\gamma]\ F[\circ\circ\gamma']$ y $C[\circ\circ]\ F[\,] \longmapsto G[\circ\circ]$ permitiría crear derivaciones de la forma

$$\begin{aligned}
(\xi C_1[\delta], w_1) &\vdash (\xi C[\delta] F_1[\delta], w_1) \\
&\overset{*}{\vdash} (\xi C[\delta] F_1'[\,], w_2) \\
&\vdash (\xi C_2[\delta], w_2) \\
&\vdash (\xi C_2[\delta] F_2[\,], w_2) \\
&\overset{*}{\vdash} (\xi C_2[\delta] F_2'[\delta], w_3) \\
&\vdash (\xi C_3[\delta], w_3) \\
&\vdash (\xi C_3[\delta] F_3[\,], w_3) \\
&\vdots
\end{aligned}$$

en las cuales una pila de índices es compartida por F_1, F_2, F_3, $\ldots$, perdiéndose de ese modo la linealidad. Para evitar este tipo de derivaciones no deseadas debemos establecer la restricción de que en toda derivación, una transición $C[\circ\circ]\ F[\,] \longmapsto G[\circ\circ]$ sólo puede ser utilizada para eliminar de la pila un elemento $F[\,]$ que se encuentra a la misma altura que un elemento $F'[\,]$ apilado mediante una transición $C[\circ\circ\gamma] \longmapsto C[\circ\circ\gamma]\ F'[\,]$. En el caso de que el elemento $F'[\,]$ hubiese sido apilado utilizando otro tipo de transición, el elemento $F[\,]$ deberá ser sacado de la pila utilizando una transición $C[\circ\circ]\ F[\,] \overset{a}{\longmapsto} G[\,]$.

Teorema 8.4 *Los autómatas lógicos a pila restringidos que utilizan el juego de transiciones de la tabla 8.33 y que únicamente emparejan transiciones $C[\circ\circ]\ F[\,] \overset{a}{\longmapsto} G[\circ\circ]$ con transiciones $C[\circ\circ\gamma] \overset{a}{\longmapsto} C[\circ\circ\gamma]\ F'[\,]$, aceptan la clase de los lenguajes de adjunción de árboles.*

Demostración:

Por los esquemas de compilación de LIG y TAG en autómatas lógicos a pila restringidos sabemos que los lenguajes de adjunción de árboles son aceptados por RLPDA utilizando las transiciones de la tabla 8.33 con las restricciones mencionadas.

Para demostrar que todo lenguaje aceptado por un RLPDA que utilice las transiciones de la tabla 8.33 con las restricciones mencionadas es un lenguaje de adjunción de árboles, definiremos un procedimiento para crear una gramática lineal de índices a partir de tales autómatas.

Sea $\mathcal{A} = (V_T, P, V_I, \mathcal{X}, \$_0, \$_f, \Theta)$ un autómata lógico a pila restrigido. Construiremos una gramática lineal de índices $\mathcal{L} = (V_T, V_N, V_I, S, Prod)$, donde el conjunto V_N de no-terminales estará formado por pares $\langle E, B \rangle$ tal que $A, B \in P$. Para que $\mathcal{L}$ reconozca el lenguaje aceptado por $\mathcal{A}$ el conjunto de producciones en $Prod$ ha de construirse a partir de las transiciones en Θ de la siguiente manera:

- Para toda transición $C[\circ\circ] \overset{a}{\longmapsto} F[\circ\circ]$ y para todo $E \in P$ creamos una producción

$$\langle C, E \rangle[\circ\circ] \rightarrow a \; \langle F, E \rangle[\circ\circ]$$

- Para todo par de transiciones $C[\circ\circ]\; F[\,] \longmapsto G[\circ\circ]$ y $C[\circ\circ] \longmapsto C[\circ\circ]\; F'[\,]$, y para todo $E \in P$ creamos una producción

$$\langle C, E \rangle[\circ\circ] \rightarrow \langle F', F \rangle[\,] \;\; \langle G, E \rangle[\circ\circ]$$

- Para todo par de transiciones $C[\circ\circ]\; F[\,] \longmapsto G[\,]$ y $C[\circ\circ\gamma] \longmapsto C[\circ\circ\gamma] F'[\circ\circ\gamma']$, y para todo $E \in P$ creamos una producción

$$\langle C, E \rangle[\circ\circ\gamma] \rightarrow \langle F', F \rangle[\circ\circ\gamma'] \;\; \langle G, E \rangle[\,]$$

- Para todo $E \in P$ creamos una producción

$$\langle E, E \rangle[\,] \rightarrow \epsilon$$

- Para toda transición $\$_0[\circ\circ] \longmapsto \$_0[\circ\circ]\;\; F[\;]$ o $\$_0[\circ\circ] \longmapsto \$_0[\circ\circ]\;\; F[\circ\circ]$, donde $F \in P - \{\$_0\}$, creamos una producción

$$\langle \$_0, \$_0 \rangle[\circ\circ] \rightarrow \langle F, \$_f \rangle[\circ\circ]$$

- Para toda transición $C[\circ\circ] \overset{a}{\longmapsto} \$_f[\circ\circ]$ creamos una transición

$$\langle C, \$_f \rangle[\,] \rightarrow a$$

Con respecto al axioma de la gramática, tenemos que $S = \langle \$_0, \$_0 \rangle$.

Mediante inducción en la longitud de las derivaciones, es posible mostrar que $\langle B, E \rangle[\alpha] \overset{*}{\Rightarrow} w$ si y sólo si $(B[\alpha], w) \overset{*}{\vdash} (E[\,], \epsilon)$, puesto que

- Si una derivación $(C[\alpha], w) \overset{*}{\vdash} (E[\,], \epsilon)$ es el resultado de aplicar la secuencia $t_1, \ldots, t_m$ de transiciones en Θ, entonces existe una secuencia $p_1, \ldots, p'_m$ de producciones en $Prod$ tal que la derivación $\langle C, E \rangle[\alpha] \overset{*}{\Rightarrow} w$ resultado de aplicar $p_1, \ldots, p'_m$ reconoce w. La demostración se realiza por inducción en la longitud de la derivación del autómata. El caso base lo constituye la derivación $(E[\;], \epsilon) \overset{0}{\vdash} (E[\;], \epsilon)$, para la que existe una transición $\langle E, E \rangle[\;] \rightarrow \epsilon$. Por hipótesis de inducción suponemos que la proposición se cumple para cualquier derivación del autómata de longitud m. En tal caso, durante el paso de inducción verificamos que se cumple para cualquier posible derivación de longitud mayor que m:
  - Si $(C[\alpha], aw) \vdash (F[\alpha], w) \overset{m}{\vdash} (E[\;], \epsilon)$, existe una producción $\langle C, E \rangle[\circ\circ] \rightarrow a \; \langle F, E \rangle[\circ\circ]$, por hipótesis de inducción $\langle F, E \rangle[\alpha] \overset{*}{\Rightarrow} w$ y en consecuencia $\langle C, E \rangle[\alpha] \overset{*}{\Rightarrow} aw$.
  - Si $(C[\alpha], w_1 w_2) \vdash (C[\alpha]\;\; F'[\;], w_1 w_2) \overset{m_1}{\vdash} (C[\alpha]\;\; F[\;], w_2) \vdash (G[\alpha], w_2) \overset{m_2}{\vdash} (E[\;], \epsilon)$, existe una producción $\langle C, E \rangle[\circ\circ] \rightarrow \langle F', F \rangle[\,]\; \langle G, E \rangle[\circ\circ]$, por hipótesis de inducción $\langle F', F \rangle[\,] \overset{*}{\Rightarrow} w_1$ y $\langle G, E \rangle[\alpha] \overset{*}{\Rightarrow} w_2$ y en consecuencia $\langle B, E \rangle[\alpha] \overset{*}{\Rightarrow} w_1 w_2$.

- Si $(C[\alpha\gamma], w_1 w_2) \vdash (C[\alpha\gamma]F'[\alpha\gamma'], w_1 w_2) \overset{m_1}{\vdash} (C[\alpha\gamma]\ F[\ ], w_2) \vdash (G[\ ], w_2) \overset{m_2}{\vdash} (E[\ ], \epsilon)$, existe una producción $\langle C, E\rangle[\circ\circ\gamma] \to \langle F', F\rangle[\circ\circ\gamma']\ \langle G, E\rangle[\ ]$, por hipótesis de inducción $\langle F', F\rangle[\alpha\gamma'] \overset{*}{\Rightarrow} w_1$ y $\langle G, E\rangle[\ ] \overset{*}{\Rightarrow} w_2$ y en consecuencia $\langle C, E\rangle[\alpha\gamma] \overset{*}{\Rightarrow} w_1 w_2$.

- Si una derivación izquierda $\langle C, E\rangle[\alpha] \overset{*}{\Rightarrow} w$ reconoce la cadena w como resultado de aplicar la secuencia $p_1, \ldots, p_m$ de producciones en $Prod$, entonces existe una secuencia de transiciones $t_1, \ldots, t'_m$ tal que la derivación $(C[\alpha], w) \overset{*}{\vdash} (E[\ ], \epsilon)$ es el resultado de aplicar la secuencia de transiciones $t_1, \ldots, t'_m$. La demostración se realiza por inducción en la longitud de la derivación de la gramática. El caso base lo constituye la derivación $\langle E, E\rangle[\ ] \Rightarrow \epsilon$, para la que existe una derivación $(E[\ ], \epsilon) \overset{0}{\vdash} (E[\ ], \epsilon)$ en el autómata. Por hipótesis de inducción suponemos que la proposición se cumple para cualquier derivación de la gramática de longitud m. En tal caso, durante el paso de inducción verificamos que se cumple para cualquier posible derivación de longitud mayor que m:
  - Si $\langle C, E\rangle[\alpha] \Rightarrow a\ \langle F, E\rangle[\alpha] \overset{m}{\Rightarrow} aw$, existe una transición $C[\circ\circ] \overset{a}{\longmapsto} F[\circ\circ]$, por hipótesis de inducción $(F[\alpha], w) \overset{*}{\vdash} (E[\,], \epsilon)$ y en consecuencia $(C[\alpha], aw) \overset{*}{\vdash} (E[\,], \epsilon)$.
  - Si $\langle C, E\rangle[\alpha] \Rightarrow \langle F', F\rangle[\ ]\ \langle G, E\rangle[\alpha] \overset{m_1}{\Rightarrow} w_1\ \langle G, E\rangle[\alpha] \overset{m_2}{\Rightarrow} w_1 w_2$, existe un par de transiciones $C[\circ\circ] \longmapsto C[\circ\circ]\ F'[\ ]$ y $C[\circ\circ]\ F[\ ] \longmapsto G[\circ\circ]$, por hipótesis de inducción $(F'[\,], w_1) \overset{*}{\vdash} (F[\,], \epsilon)$ y $(G[\alpha], w_2) \overset{*}{\vdash} (E[\ ], \epsilon)$ y en consecuencia $(C[\alpha], w_1 w_2) \overset{*}{\vdash} (E[\,], \epsilon)$.
  - Si $\langle C, E\rangle[\alpha\gamma] \Rightarrow \langle F', F\rangle[\alpha\gamma']\ \langle G, E\rangle[\ ] \overset{m_1}{\Rightarrow} w_1\ \langle G, E\rangle[\ ] \overset{m_2}{\Rightarrow} w_1 w_2$, existe un par de transiciones $C[\circ\circ\gamma] \longmapsto C[\circ\circ\gamma]\ F'[\circ\circ\gamma']$ y $C[\circ\circ]\ F[\ ] \longmapsto G[\ ]$, por hipótesis de inducción $(F'[\alpha\gamma'], w_1) \overset{*}{\vdash} (F[\,], \epsilon)$ y $(G[\ ], w_2) \overset{*}{\vdash} (E[\ ], \epsilon)$ y en consecuencia $(C[\alpha\gamma], w_1 w_2) \overset{*}{\vdash} (E[\,], \epsilon)$.

□

A continuación diseñaremos una técnica de tabulación general para los autómatas lógicos a pila restringidos que utilizan las transiciones de la tabla 8.33 con la restricción mencionada anteriormente. Debemos considerar los siguientes tipos de derivaciones:

Derivaciones de llamada. Corresponden a la transmisión de una pila de índices en la fase de llamada de la estrategia de análisis e implican la existencia de la siguiente secuencia de subderivaciones:

$$\begin{aligned}(\xi\ A[\delta], a_{h+1} \ldots a_n) &\overset{*}{\vdash}_{d_1} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma'], a_{i+1} \ldots a_n) \\ &\overset{*}{\vdash}_{d_2} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']\ C[\delta\gamma], a_{j+1} \ldots a_n)\end{aligned}$$

donde $\gamma, \gamma' \in V_I,\ \ \delta, \delta' \in V_I^*$ y bien $\delta = \delta'\gamma'$, bien $\delta'\gamma' = \delta\gamma$ o bien $\delta' = \delta\gamma$. La pila $\xi\ A[\delta]$ no debe ser alterada en ningún momento, aunque se permite la consulta de A. La pila $\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']$ no debe ser alterada durante la subderivación d_2, aunque se permite la consulta de B y de γ'. $A[\delta]$ se corresponde con el elemento de la pila del autómata más próximo a la cima tal que su pila asociada es δ. Evidentemente, si $\delta'\gamma' = \delta$, entonces $A[\delta]$ y $B[\delta'\gamma']$ son el mismo elemento. La figura 8.8 muestra una representación gráfica de este tipo de derivaciones.

Para cualquier $\xi' \in (P[V_I^*])^*$ y $\alpha \in V_I*$ tal que las restricciones previamente enunciadas para δ y δ' son aplicadas a α y α', se cumple que:

$$\begin{aligned}(\xi'\ A[\alpha], a_{h+1} \ldots a_n) &\overset{*}{\vdash}_{d_1} (\xi'\ A[\alpha]\ \xi'_1\ B[\alpha'\gamma'], a_{i+1} \ldots a_n) \\ &\overset{*}{\vdash}_{d_2} (\xi'\ A[\alpha]\ \xi'_1\ B[\alpha'\gamma']\ C[\alpha\gamma], a_{j+1} \ldots a_n)\end{aligned}$$

En consecuencia, podremos utilizar ítems de la forma

$$[A, h \mid B, i, \gamma', C, j, \gamma \mid -, -, -, -, -]$$

para representar este tipo de derivaciones.

Derivaciones de retorno. Corresponden a la fase de retorno de la estrategia de análisis e implican la existencia de la siguiente secuencia de subderivaciones:

$$\begin{array}{ll} (\xi\ A[\delta], a_{h+1} \ldots a_n) & \overset{*}{\vdash}_{d_1} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma'], a_{i+1} \ldots a_n) \\ & \overset{*}{\vdash}_{d_2} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']\ \xi_2\ D[\delta\eta], a_{p+1} \ldots a_n) \\ & \overset{*}{\vdash}_{d_3} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']\ \xi_2\ D[\delta\eta]\ E[\,], a_{q+1} \ldots a_n) \\ & \overset{*}{\vdash}_{d_4} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']\ C[\,], a_{j+1} \ldots a_n) \end{array}$$

donde $\gamma, \gamma', \eta \in V_I$, $\delta, \delta' \in V_I^*$, y bien $\delta = \delta'\gamma'$, bien $\delta'\gamma' = \delta\gamma$ o bien $\delta' = \delta\gamma$. Si $\delta = \delta'\gamma'$ entonces $A[\delta]$ y $B[\delta'\gamma']$ se refieren al mismo elemento de la pila del autómata, en otro caso $A[\delta]$ es el elemento de la pila del autómata más próximo a la cima tal que su pila asociada es δ. La pila $\xi\ A[\delta]$ no debe ser alterada en ningún momento, aunque se permite la consulta de A en la subderivación d_1. La pila $\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']$ no debe ser alterada durante las subderivaciones d_2, d_3 y d_4, aunque se permite la consulta de B y de γ' en la subderivación d_2. La pila $\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']\ \xi_2\ D[\delta\eta]$ no debe ser alterada durante la subderivación d_3, aunque se permite la consulta de D y de η. La figura 8.9 muestra una representación gráfica de este tipo de derivaciones.

Para cualquier pila $\xi' \in (P[V_I^*])^*$ y pila de índices $\alpha \in V_I^*$ tal que existe una derivación $(D[\alpha\eta], a_{p+1} \ldots a_n) \overset{*}{\vdash} (D[\alpha\eta]\ E[\,], a_{q+1} \ldots a_n)$ se cumple

$$\begin{array}{ll} (\xi'\ A[\alpha], a_{h+1} \ldots a_n) & \overset{*}{\vdash}_{d_1} (\xi'\ A[\alpha]\ \xi_1\ B[\alpha'\gamma'], a_{i+1} \ldots a_n) \\ & \overset{*}{\vdash}_{d_2} (\xi'\ A[\alpha]\ \xi_1\ B[\alpha'\gamma']\ \xi_2\ D[\alpha\eta], a_{p+1} \ldots a_n) \\ & \overset{*}{\vdash}_{d_3} (\xi'\ A[\alpha]\ \xi_1\ B[\alpha'\gamma']\ \xi_2\ D[\alpha\eta]\ E[\,], a_{q+1} \ldots a_n) \\ & \overset{*}{\vdash}_{d_4} (\xi'\ A[\alpha]\ \xi_1\ B[\alpha'\gamma']\ C[\,], a_{j+1} \ldots a_n) \end{array}$$

Los ítems que representan este tipo de derivaciones son de la forma

$$[A, h \mid B, i, \gamma', C, j, - \mid D, p, \eta, E, q]$$

donde los componentes (A, h) y (D, p, η, E, q) garantizan que se está utilizando la misma δ en toda la derivación.

Derivaciones de puntos especiales. Corresponden a la creación de una nueva pila de índices o a la terminación de una de tales pilas:

$$\begin{array}{ll} (\xi\ B[\delta'\gamma'], a_{i+1} \ldots a_n) & \vdash (\xi\ B[\delta'\gamma']\ D[\,], a_{i+1} \ldots a_n) \\ & \overset{*}{\vdash}_{d} (\xi\ B[\delta'\gamma']\ C[\,], a_{j+1} \ldots a_n) \end{array}$$

donde $\delta' \in V_I^*$ y $\gamma' \in V_I$. La pila $\xi\ B[\delta'\gamma']$ no debe ser alterada en ningún momento, aunque se permite la consulta de B. La figura 8.10 muestra una representación gráfica de este tipo de derivaciones.

Para cualquier $\xi' \in (P[V_I^*])^*$ y $\alpha'\gamma' \in V_I^*$ se cumple

$$\begin{array}{ll}(\xi'\ B[\alpha'\gamma'], a_{i+1}\dots a_n) & \vdash (\xi\ B[\alpha'\gamma']\ D[\,], a_{i+1}\dots a_n) \\ & \overset{*}{\vdash}_d (\xi'\ B[\alpha'\gamma']\ C[\,], a_{j+1}\dots a_n)\end{array}$$

La forma de los ítems que representan derivaciones de puntos especiales es la siguiente:

$$[-,- \mid B,i,\gamma',C,j,- \mid -,-,-,-,-]$$

En la tablas 8.34 y 8.35 se muestran las reglas de combinación de ítems para los diferentes tipos de transiciones. El ítem inicial es $[-,- \mid -,0,-,\$_0,0,- \mid -,-,-,-,-]$ y los ítems finales son de la forma $[-,- \mid \$_0,0,-,S,n,- \mid -,-,-,-,-]$.

$$\frac{[A,h \mid B,i,\gamma',C,j,\gamma \mid D,p,\eta,E,q]}{[-,- \mid C,j,\gamma,F,j,- \mid -,-,-,-,-]}\quad C[\circ\circ] \longmapsto C[\circ\circ]\ F[\,]$$

$$\frac{[A,h \mid B,i,\gamma',C,j,\gamma \mid -,-,-,-,-]}{[A,h \mid C,j,\gamma,F,j,\gamma \mid -,-,-,-,-]}\quad C[\circ\circ] \longmapsto C[\circ\circ]\ F[\circ\circ]$$

$$\frac{[A,h \mid B,i,\gamma',C,j,- \mid D,p,\eta,E,q]}{[-,- \mid C,j,-,F,j,- \mid -,-,-,-,-]}\quad C[\circ\circ] \longmapsto C[\circ\circ]\ F[\circ\circ]$$

$$\frac{[A,h \mid B,i,\gamma'',C,j,\gamma \mid -,-,-,-,-]}{[C,j \mid C,j,\gamma,F,j,\gamma' \mid -,-,-,-,-]}\quad C[\circ\circ] \longmapsto C[\circ\circ]\ F[\circ\circ\gamma']$$

$$\frac{[A,h \mid B,i,\gamma'',C,j,- \mid D,p,\eta,E,q]}{[C,j \mid C,j,-,F,j,\gamma' \mid -,-,-,-,-]}\quad C[\circ\circ] \longmapsto C[\circ\circ]\ F[\circ\circ\gamma']$$

$$\frac{\begin{array}{c}[A,h \mid B,i,\gamma'',C,j,\gamma \mid -,-,-,-,-] \\ [M,m \mid N,t,\gamma''',A,h,\gamma' \mid -,-,-,-,-]\end{array}}{[M,m \mid C,j,\gamma,F,j,\gamma' \mid -,-,-,-,-]}\quad C[\circ\circ\gamma] \longmapsto C[\circ\circ\gamma]\ F[\circ\circ]$$

$$\frac{\begin{array}{c}[A,h \mid B,i,\gamma'',C,j,\gamma \mid -,-,-,-,-] \\ [M,m \mid N,t,\gamma'''A,h,- \mid D,p,\eta,E,q]\end{array}}{[-,- \mid C,j,\gamma,F,j,- \mid -,-,-,-,-]}\quad C[\circ\circ\gamma] \longmapsto C[\circ\circ\gamma]\ F[\circ\circ]$$

$$\frac{[A,h \mid B,i,\gamma',C,j,\gamma \mid D,p,\eta,E,q]}{[A,h \mid B,i,\gamma',F,k,\gamma \mid D,p,\eta,E,q]}\quad C[\circ\circ] \overset{a}{\longmapsto} F[\circ\circ],\ \ k=j \text{ si } a=\epsilon,\ \ k=j+1 \text{ si } a \in V_T$$

Tabla 8.34: Combinación de ítems en las estrategias $*$-descendentes (fase de llamada)

Teorema 8.5 *La manipulación de configuraciones mediante la aplicación de transiciones en los autómatas a pila restringidos con estrategias $*$-descendentes es equivalente a la manipulación de ítems mediante las reglas de combinación de las tablas 8.34 y 8.35.*

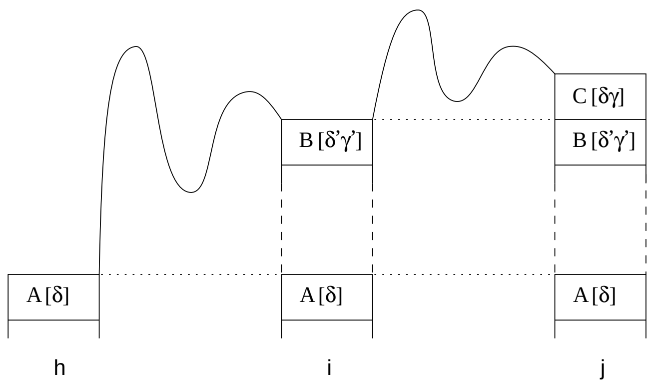

Figura 8.8: Derivaciones de llamada en estrategias ∗-descendentes

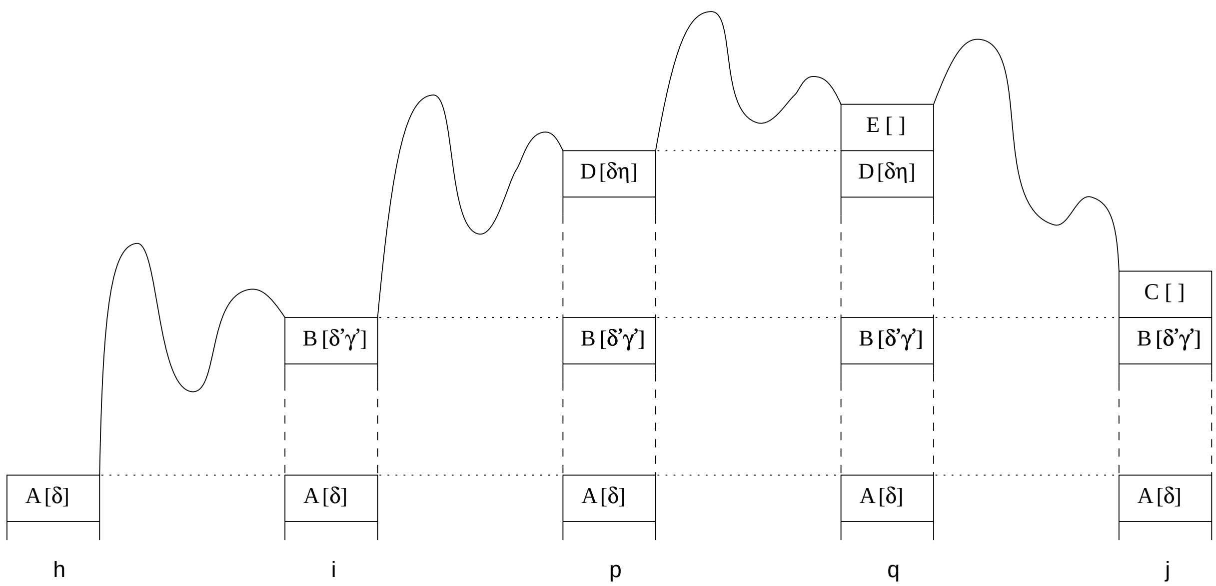

Figura 8.9: Derivaciones de retorno en estrategias ∗-descendentes

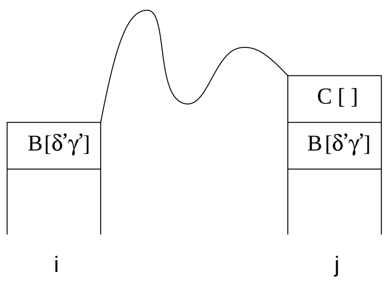

Figura 8.10: Derivaciones de puntos especiales en estrategias ∗-descendentes

$$\frac{\begin{array}{c}[-,- \mid C,j,\gamma,F,k,- \mid -,-,-,-,-]\\ [A,h \mid B,i,\gamma',C,j,\gamma \mid D,p,\eta,E,q]\end{array}}{[A,h \mid B,i,\gamma',G,k,\gamma \mid D,p,\eta,E,q]} \quad \begin{array}{l} C[\circ\circ] \longmapsto C[\circ\circ]\ F'[\,]\\ C[\circ\circ]\ F[\,] \longmapsto G[\circ\circ]\end{array}$$

$$\frac{\begin{array}{c}[A,h \mid C,j,\gamma,F,k,- \mid D,p,\gamma,E,q]\\ [A,h \mid B,i,\gamma',C,j,\gamma \mid -,-,-,-,-]\end{array}}{[A,h \mid B,i,\gamma',G,k,- \mid D,p,\gamma,E,q]} \quad \begin{array}{l} C[\circ\circ] \longmapsto C[\circ\circ]\ F'[\circ\circ]\\ C[\circ\circ]\ F[\,] \longmapsto G[\,]\end{array}$$

$$\frac{\begin{array}{c}[-,- \mid C,j,-,F,k,- \mid -,-,-,-,-]\\ [A,h \mid B,i,\gamma',C,j,- \mid D,p,\eta,E,q]\end{array}}{[A,h \mid B,i,\gamma',G,k,- \mid D,p,\eta,E,q]} \quad \begin{array}{l} C[\circ\circ] \longmapsto C[\circ\circ]\ F'[\circ\circ]\\ C[\circ\circ]\ F[\,] \longmapsto G[\,]\end{array}$$

$$\frac{\begin{array}{c}[C,j \mid C,j,\gamma,F,k,- \mid D,p,\gamma',E,q]\\ [A,h \mid B,i,\gamma'',C,j,\gamma \mid -,-,-,-,-]\\ [A,h \mid D,p,\gamma',E,q,- \mid O,u,\gamma,P,v]\end{array}}{[A,h \mid B,i,\gamma'',G,k,- \mid O,u,\gamma,P,v]} \quad \begin{array}{l} C[\circ\circ] \longmapsto C[\circ\circ]\ F'[\circ\circ\gamma']\\ C[\circ\circ]\ F[\,] \longmapsto G[\,]\end{array}$$

$$\frac{\begin{array}{c}[C,j \mid C,j,-,F,k,- \mid O,u,\gamma',P,v]\\ [A,h \mid B,i,\gamma'',C,j,- \mid D,p,\eta,E,q]\\ [-,- \mid O,u,\gamma',P,v,- \mid -,-,-,-,-]\end{array}}{[A,h \mid B,i,\gamma'',G,k,- \mid D,p,\eta,E,q]} \quad \begin{array}{l} C[\circ\circ] \longmapsto C[\circ\circ]\ F'[\circ\circ\gamma']\\ C[\circ\circ]\ F[\,] \longmapsto G[\,]\end{array}$$

$$\frac{\begin{array}{c}[M,m \mid C,j,\gamma,F,k,- \mid D,p,\gamma',E,q]\\ [A,h \mid B,i,\gamma'',C,j,\gamma \mid -,-,-,-,-]\\ [M,m \mid N,t,\gamma''',A,h,\gamma' \mid -,-,-,-,-]\end{array}}{[A,h \mid B,i,\gamma'',G,k,- \mid C,j,\gamma,F,k]} \quad \begin{array}{l} C[\circ\circ\gamma] \longmapsto C[\circ\circ\gamma]\ F'[\circ\circ]\\ C[\circ\circ]\ F[\,] \longmapsto G[\,]\end{array}$$

$$\frac{\begin{array}{c}[-,- \mid C,j,\gamma,F,k,- \mid -,-,-,-,-]\\ [A,h \mid B,i,\gamma'',C,j,\gamma \mid -,-,-,-,-]\\ [M,m \mid N,t,\gamma''' A,h,- \mid D,p,\eta,E,q]\end{array}}{[A,h \mid B,i,\gamma'',G,k,- \mid C,j,\gamma,F,k} \quad \begin{array}{l} C[\circ\circ\gamma] \longmapsto C[\circ\circ\gamma]\ F'[\circ\circ]\\ C[\circ\circ]\ F[\,] \longmapsto G[\,]\end{array}$$

Tabla 8.35: Combinación de ítems en las estrategias $*$-descendentes (fase de retorno)

<u>Demostración</u>:

Puesto que un ítem representa una derivación y toda derivación debe ser representada por algún ítem, es suficiente con demostrar que la combinación de los ítems produce ítems que se corresponden con derivaciones válidas y que para toda derivación que se pueda producir como resultado de la aplicación de una transición, existe una regla de combinación de ítems que produce un ítem que representa a dicha derivación. A continuación se muestra una lista de todos los casos posibles de derivación que se pueden dar junto con la correspondiente regla de combinación de ítems.

- Derivaciones que son el resultado de aplicar una transición $C[\circ\circ] \longmapsto C[\circ\circ]\ F[\,]$
  - a una derivación de llamada:

$$\begin{array}{ll}(\xi\ A[\delta], a_{h+1} \ldots a_n) & \overset{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma'], a_{i+1} \ldots a_n) \\ & \overset{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']\ C[\delta\gamma], a_{j+1} \ldots a_n) \\ & \vdash (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']\ C[\delta\gamma]\ F[\,], a_{j+1} \ldots a_n)\end{array}$$

$$\frac{[A,h \mid B,i,\gamma',C,j,\gamma \mid -,-,-,-,-]}{[-,- \mid C,j,\gamma,F,j,- \mid -,-,-,-,-]}\ C[\circ\circ] \longmapsto C[\circ\circ]\ F[\,]$$

  - a una derivación de retorno:

$$\begin{array}{ll}(\xi\ A[\delta], a_{h+1} \ldots a_n) & \overset{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma'], a_{i+1} \ldots a_n) \\ & \overset{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']\ \xi_2\ D[\delta\eta], a_{p+1} \ldots a_n) \\ & \overset{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']\ \xi_2\ D[\delta\eta]\ E[\,], a_{q+1} \ldots a_n) \\ & \overset{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']\ C[\,], a_{j+1} \ldots a_n) \\ & \vdash (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']\ C[\,]\ F[\,], a_{j+1} \ldots a_n)\end{array}$$

$$\frac{[A,h \mid B,i,\gamma',C,j,- \mid D,p,\eta,E,q]}{[-,- \mid C,j,-,F,j,- \mid -,-,-,-,-]}\ C[\circ\circ] \longmapsto C[\circ\circ]\ F[\,]$$

  - a una derivación de puntos especiales:

$$\begin{array}{ll}(\xi\ B[\delta'\gamma'], a_{i+1} \ldots a_n) & \vdash (\xi\ B[\delta'\gamma']\ D[\,], a_{i+1} \ldots a_n) \\ & \overset{*}{\vdash} (\xi\ B[\delta'\gamma']\ C[\,], a_{j+1} \ldots a_n) \\ & \vdash (\xi\ B[\delta'\gamma']\ C[\,]\ F[\,], a_{j+1} \ldots a_n)\end{array}$$

$$\frac{[-,- \mid B,i,\gamma',C,j,- \mid -,-,-,-,-]}{[-,- \mid C,j,-,F,j,- \mid -,-,-,-,-]}\ C[\circ\circ] \longmapsto C[\circ\circ]\ F[\,]$$

- Derivaciones que son el resultado de aplicar una transición $C[\circ\circ] \longmapsto C[\circ\circ]\ F[\circ\circ]$
  - a una derivación de llamada:

$$\begin{array}{ll}(\xi\ A[\delta], a_{h+1} \ldots a_n) & \overset{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma'], a_{i+1} \ldots a_n) \\ & \overset{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']\ C[\delta\gamma], a_{j+1} \ldots a_n) \\ & \vdash (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']\ C[\delta\gamma]\ F[\delta\gamma], a_{j+1} \ldots a_n)\end{array}$$

$$\frac{[A,h \mid B,i,\gamma',C,j,\gamma \mid -,-,-,-,-]}{[A,h \mid C,j,\gamma,F,j,\gamma \mid -,-,-,-,-]}\ C[\circ\circ] \longmapsto C[\circ\circ]\ F[\circ\circ]$$

  - a una derivación de retorno:

$$\begin{array}{ll}(\xi\ A[\delta], a_{h+1} \ldots a_n) & \overset{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma'], a_{i+1} \ldots a_n) \\ & \overset{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']\ \xi_2\ D[\delta\eta], a_{p+1} \ldots a_n) \\ & \overset{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']\ \xi_2\ D[\delta\eta]\ E[\,], a_{q+1} \ldots a_n) \\ & \overset{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']\ C[\,], a_{j+1} \ldots a_n) \\ & \vdash (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']\ C[\,]\ F[\,], a_{j+1} \ldots a_n)\end{array}$$

$$\frac{[A,h \mid B,i,\gamma',C,j,- \mid D,p,\eta,E,q]}{[-,- \mid C,j,-,F,j,- \mid -,-,-,-,-]} \quad C[\circ\circ] \longmapsto C[\circ\circ]\ F[\circ\circ]$$

– a una derivación de puntos especiales:

$$\begin{array}{ll}(\xi\ B[\delta'\gamma'], a_{i+1}\dots a_n) & \vdash (\xi\ B[\delta'\gamma']\ D[\,], a_{i+1}\dots a_n) \\ & \overset{*}{\vdash} (\xi\ B[\delta'\gamma']\ C[\,], a_{j+1}\dots a_n) \\ & \vdash (\xi\ B[\delta'\gamma']\ C[\,]\ F[\,], a_{j+1}\dots a_n)\end{array}$$

$$\frac{[-,- \mid B,i,\gamma',C,j,- \mid -,-,-,-,-]}{[-,- \mid C,j,-,F,j,- \mid -,-,-,-,-]} \quad C[\circ\circ] \longmapsto C[\circ\circ]\ F[\circ\circ]$$

- Derivaciones que son el resultado de aplicar una transición $C[\circ\circ] \longmapsto C[\circ\circ]\ F[\circ\circ\gamma']$
  – a una derivación de llamada:

$$\begin{array}{ll}(\xi\ A[\delta], a_{h+1}\dots a_n) & \overset{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma''], a_{i+1}\dots a_n) \\ & \overset{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma'']\ C[\delta\gamma], a_{j+1}\dots a_n) \\ & \vdash (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma'']\ C[\delta\gamma]\ F[\delta\gamma\gamma'], a_{j+1}\dots a_n)\end{array}$$

$$\frac{[A,h \mid B,i,\gamma'',C,j,\gamma \mid -,-,-,-,-]}{[C,j \mid C,j,\gamma,F,j,\gamma' \mid -,-,-,-,-]} \quad C[\circ\circ] \longmapsto C[\circ\circ]\ F[\circ\circ\gamma']$$

  – a una derivación de retorno:

$$\begin{array}{ll}(\xi\ A[\delta], a_{h+1}\dots a_n) & \overset{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma''], a_{i+1}\dots a_n) \\ & \overset{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma'']\ \xi_2\ D[\delta\eta], a_{p+1}\dots a_n) \\ & \overset{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma'']\ \xi_2\ D[\delta\eta]\ E[\,], a_{q+1}\dots a_n) \\ & \overset{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']\ C[\,], a_{j+1}\dots a_n) \\ & \vdash (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']\ C[\,]\ F[\gamma'], a_{j+1}\dots a_n)\end{array}$$

$$\frac{[A,h \mid B,i,\gamma'',C,j,- \mid D,p,\eta,E,q]}{[C,j \mid C,j,-,F,j,\gamma' \mid -,-,-,-,-]} \quad C[\circ\circ] \longmapsto C[\circ\circ]\ F[\circ\circ\gamma']$$

  – a una derivación de puntos especiales:

$$\begin{array}{ll}(\xi\ B[\delta'\gamma''], a_{i+1}\dots a_n) & \vdash (\xi\ B[\delta'\gamma']\ D[\,], a_{i+1}\dots a_n) \\ & \overset{*}{\vdash} (\xi\ B[\delta'\gamma'']\ C[\,], a_{j+1}\dots a_n) \\ & \vdash (\xi\ B[\delta'\gamma'']\ C[\,]\ F[\gamma'], a_{j+1}\dots a_n)\end{array}$$

$$\frac{[-,- \mid B,i,\gamma',C,j,- \mid -,-,-,-,-]}{[C,j \mid C,j,-,F,j,\gamma' \mid -,-,-,-,-]} \quad C[\circ\circ] \longmapsto C[\circ\circ]\ F[\circ\circ]$$

- Derivaciones que son el resultado de aplicar una transición $C[\circ\circ\gamma] \longmapsto C[\circ\circ\gamma]\ F[\circ\circ]$ a una derivación de llamada[8], con los tres casos siguientes
  – la derivación de llamada es a su vez derivada a partir de una derivación de llamada:

$$\begin{array}{ll}(\xi\ M[\delta], a_{m+1}\dots a_n) & \overset{*}{\vdash} (\xi\ M[\delta]\ N[\delta''\gamma'''], a_{t+1}\dots a_n) \\ & \overset{*}{\vdash} (\xi\ M[\delta]\ N[\delta''\gamma''']\ A[\delta\gamma'], a_{h+1}\dots a_n) \\ & \overset{*}{\vdash} (\xi\ M[\delta]\ N[\delta''\gamma''']\ A[\delta\gamma']\ \xi_1\ B[\delta'\gamma''], a_{i+1}\dots a_n) \\ & \overset{*}{\vdash} (\xi\ M[\delta]\ N[\delta''\gamma''']\ A[\delta\gamma']\ \xi_1\ B[\delta'\gamma'']\ C[\delta\gamma'\gamma], a_{j+1}\dots a_n) \\ & \vdash (\xi\ M[\delta]\ N[\delta''\gamma''']\ A[\delta\gamma']\ \xi_1\ B[\delta'\gamma'']\ C[\delta\gamma'\gamma]\ F[\delta\gamma'], a_{j+1}\dots a_n)\end{array}$$

$$\frac{\begin{array}{c}[A,h \mid B,i,\gamma'',C,j,\gamma \mid -,-,-,-,-] \\ [M,m \mid N,t,\gamma''',A,h,\gamma' \mid -,-,-,-,-]\end{array}}{[M,m \mid C,j,\gamma,F,j,\gamma' \mid -,-,-,-,-]} \quad C[\circ\circ\gamma] \longmapsto C[\circ\circ\gamma]\ F[\circ\circ]$$

[8] En este caso no se puede aplicar una transición $C[\circ\circ\gamma] \longmapsto C[\circ\circ\gamma]\ F[\circ\circ]$ a una derivación de retorno o de puntos especiales puesto que en ambas la pila de índices de la cima está vacía.

– la derivación de llamada es a su vez derivada a partir de una derivación de retorno:

$$\begin{array}{ll}(\xi\ M[\delta], a_{m+1} \ldots a_n) & \stackrel{*}{\vdash} (\xi\ M[\delta]\ N[\delta''\gamma'''], a_{t+1} \ldots a_n) \\ & \stackrel{*}{\vdash} (\xi\ M[\delta]\ N[\delta''\gamma''']\ \xi_1\ D[\delta\eta], a_{p+1} \ldots a_n) \\ & \stackrel{*}{\vdash} (\xi\ M[\delta]\ N[\delta''\gamma''']\ \xi_1\ D[\delta\eta]\ E[\,], a_{q+1} \ldots a_n) \\ & \stackrel{*}{\vdash} (\xi\ M[\delta]\ N[\delta''\gamma''']\ A[\,], a_{h+1} \ldots a_n) \\ & \stackrel{*}{\vdash} (\xi\ M[\delta]\ N[\delta''\gamma''']\ A[\,]\ \xi_2\ B[\delta'\gamma''], a_{i+1} \ldots a_n) \\ & \stackrel{*}{\vdash} (\xi\ M[\delta]\ N[\delta''\gamma''']\ A[\,]\ \xi_2\ B[\delta'\gamma'']\ C[\gamma], a_{j+1} \ldots a_n) \\ & \vdash (\xi\ M[\delta]\ N[\delta''\gamma''']\ A[\,]\ \xi_2\ B[\delta'\gamma'']\ C[\gamma]\ F[\,], a_{j+1} \ldots a_n)\end{array}$$

$$\frac{\begin{array}{c}[A,h \mid B,i,\gamma'',C,j,\gamma \mid -,-,-,-,-] \\ [M,m \mid N,t,\gamma'''A,h,- \mid D,p,\eta,E,q]\end{array}}{[-,- \mid C,j,\gamma,F,j,- \mid -,-,-,-,-]}\ C[\circ\circ\gamma] \longmapsto C[\circ\circ\gamma]\ F[\circ\circ]$$

– la derivación de llamada es a su vez derivada a partir de una derivación de puntos especiales:

$$\begin{array}{ll}(\xi\ N[\delta''\gamma'''], a_{t+1} \ldots a_n) & \vdash (\xi\ N[\delta'\gamma''']\ D[\,], a_{t+1} \ldots a_n) \\ & \stackrel{*}{\vdash} (\xi\ N[\delta''\gamma''']\ A[\,], a_{h+1} \ldots a_n) \\ & \stackrel{*}{\vdash} (\xi\ N[\delta''\gamma''']\ A[\,]\ \xi_2\ B[\delta'\gamma''], a_{i+1} \ldots a_n) \\ & \stackrel{*}{\vdash} (\xi\ N[\delta''\gamma''']\ A[\,]\ \xi_2\ B[\delta'\gamma'']\ C[\gamma], a_{j+1} \ldots a_n) \\ & \vdash (\xi\ N[\delta''\gamma''']\ A[\,]\ \xi_2\ B[\delta'\gamma'']\ C[\gamma]\ F[\,], a_{j+1} \ldots a_n)\end{array}$$

$$\frac{\begin{array}{c}[A,h \mid B,i,\gamma'',C,j,\gamma \mid -,-,-,-,-] \\ [-,- \mid N,t,\gamma'''A,h,- \mid -,-,-,-,-]\end{array}}{[-,- \mid C,j,\gamma,F,j,- \mid -,-,-,-,-]}\ C[\circ\circ\gamma] \longmapsto C[\circ\circ\gamma]\ F[\circ\circ]$$

- Derivaciones que son el resultado de aplicar una transición $C[\circ\circ] \stackrel{a}{\longmapsto} F[\circ\circ]$
  – a una derivación de llamada:

$$\begin{array}{ll}(\xi\ A[\delta], a_{h+1} \ldots a_n) & \stackrel{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma''], a_{i+1} \ldots a_n) \\ & \stackrel{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma'']\ C[\delta\gamma], a_{j+1} \ldots a_n) \\ & \vdash (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma'']\ F[\delta\gamma], a_{k1} \ldots a_n)\end{array}$$

$$\frac{[A,h \mid B,i,\gamma',C,j,\gamma \mid -,-,-,-,-]}{[A,h \mid B,i,\gamma'F,k,\gamma \mid -,-,-,-,-]}\ C[\circ\circ] \stackrel{a}{\longmapsto} F[\circ\circ]$$

  donde $k = j$ si $a = \epsilon$ y $k = j + 1$ si $a \in V_T$.

  – a una derivación de retorno:

$$\begin{array}{ll}(\xi\ A[\delta], a_{h+1} \ldots a_n) & \stackrel{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma''], a_{i+1} \ldots a_n) \\ & \stackrel{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma'']\ \xi_2\ D[\delta\eta], a_{p+1} \ldots a_n) \\ & \stackrel{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma'']\ \xi_2\ D[\delta\eta]\ E[\,], a_{q+1} \ldots a_n) \\ & \stackrel{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']\ C[\,], a_{j+1} \ldots a_n) \\ & \vdash (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']\ F[\,], a_{k+1} \ldots a_n)\end{array}$$

$$\frac{[A,h \mid B,i,\gamma',C,j,\gamma \mid D,p,\eta,E,q]}{[A,h \mid B,i,\gamma',F,k,\gamma \mid D,p,\eta,E,q]}\ C[\circ\circ] \stackrel{a}{\longmapsto} F[\circ\circ]$$

  donde $k = j$ si $a = \epsilon$ y $k = j + 1$ si $a \in V_T$.

– a una derivación de puntos especiales:

$$\begin{array}{ll}(\xi\ B[\delta'\gamma''], a_{i+1}\dots a_n) & \vdash (\xi\ B[\delta'\gamma'']\ D[\,], a_{i+1}\dots a_n)\\ & \overset{*}{\vdash} (\xi\ B[\delta'\gamma'']\ C[\,], a_{j+1}\dots a_n)\\ & \vdash (\xi\ B[\delta'\gamma'']\ F[\,], a_{k+1}\dots a_n)\end{array}$$

$$\frac{[-,- \mid B,i,\gamma',C,j,\gamma \mid -,-,-,-,-]}{[-,- \mid B,i,\gamma' F,k,\gamma \mid -,-,-,-,-]}\quad C[\circ\circ] \overset{a}{\longmapsto} F[\circ\circ]$$

donde $k=j$ si $a=\epsilon$ y $k=j+1$ si $a \in V_T$.

- Derivaciones que son el resultado de aplicar una transición $C[\circ\circ]\ F[\,] \longmapsto G[\circ\circ]$ a una derivación obtenida tras aplicar la transición $C[\circ\circ] \longmapsto C[\circ\circ]\ F'[\,]$

  – a una derivación de llamada:

$$\begin{array}{ll}(\xi\ A[\delta], a_{h+1}\dots a_n) & \overset{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma'], a_{i+1}\dots a_n)\\ & \overset{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']\ C[\delta\gamma], a_{j+1}\dots a_n)\\ & \vdash (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']\ C[\delta\gamma]\ F'[\,], a_{j+1}\dots a_n)\\ & \overset{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']\ C[\delta\gamma]\ F[\,], a_{k+1}\dots a_n)\\ & \vdash (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']\ G[\delta\gamma], a_{k+1}\dots a_n)\end{array}$$

$$\frac{\begin{array}{c}[-,- \mid C,j,\gamma,F,k,- \mid -,-,-,-,-]\\ [A,h \mid B,i,\gamma',C,j,\gamma \mid -,-,-,-,-]\end{array}}{[A,h \mid B,i,\gamma',G,k,\gamma \mid -,-,-,-,-]}\quad \begin{array}{l}C[\circ\circ] \longmapsto C[\circ\circ]\ F'[\,]\\ C[\circ\circ]\ F[\,] \longmapsto G[\circ\circ]\end{array}$$

  – a una derivación de retorno:

$$\begin{array}{ll}(\xi\ A[\delta], a_{h+1}\dots a_n) & \overset{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma'], a_{i+1}\dots a_n)\\ & \overset{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']\ \xi_2\ D[\delta\eta], a_{p+1}\dots a_n)\\ & \overset{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']\ \xi_2\ D[\delta\eta]\ E[\,], a_{q+1}\dots a_n)\\ & \overset{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']\ C[\,], a_{j+1}\dots a_n)\\ & \vdash (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']\ C[\,]\ F'[\,], a_{j+1}\dots a_n)\\ & \overset{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']\ C[\,]\ F[\,], a_{k+1}\dots a_n)\\ & \vdash (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']\ G[\,], a_{k+1}\dots a_n)\end{array}$$

$$\frac{\begin{array}{c}[-,- \mid C,j,\gamma,F,k,- \mid -,-,-,-,-]\\ [A,h \mid B,i,\gamma',C,j,- \mid D,p,\eta,E,q]\end{array}}{[A,h \mid B,i,\gamma',G,k,- \mid D,p,\eta,E,q]}\quad \begin{array}{l}C[\circ\circ] \longmapsto C[\circ\circ]\ F'[\,]\\ C[\circ\circ]\ F[\,] \longmapsto G[\circ\circ]\end{array}$$

  – a una derivación de puntos especiales:

$$\begin{array}{ll}(\xi\ B[\delta'\gamma'], a_{i+1}\dots a_n) & \vdash (\xi\ B[\delta'\gamma']\ D[\,], a_{i+1}\dots a_n)\\ & \overset{*}{\vdash} (\xi\ B[\delta'\gamma']\ C[\,], a_{j+1}\dots a_n)\\ & \vdash (\xi\ B[\delta'\gamma']\ C[\,]\ F'[\,], a_{j+1}\dots a_n)\\ & \overset{*}{\vdash} (\xi\ B[\delta'\gamma']\ C[\,]\ F[\,], a_{k+1}\dots a_n)\\ & \vdash (\xi\ B[\delta'\gamma']\ G[\,], a_{k+1}\dots a_n)\end{array}$$

$$\frac{\begin{array}{c}[-,- \mid C,j,-,F,k,- \mid -,-,-,-,-]\\ [-,- \mid B,i,\gamma',C,j,- \mid -,-,-,-,-]\end{array}}{[-,- \mid B,i,\gamma',G,k,- \mid -,-,-,-,-]}\quad \begin{array}{l}C[\circ\circ] \longmapsto C[\circ\circ]\ F'[\,]\\ C[\circ\circ]\ F[\,] \longmapsto G[\circ\circ]\end{array}$$

- Derivaciones que son el resultado de aplicar una transición $C[\circ\circ]\ F[\,] \longmapsto G[\circ\circ]$ a una derivación obtenida tras aplicar la transición $C[\circ\circ] \longmapsto C[\circ\circ]\ F'[\circ\circ]$

– a una derivación de llamada:

$$\begin{array}{ll}(\xi\ A[\delta], a_{h+1}\dots a_n) & \overset{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma'], a_{i+1}\dots a_n)\\ & \overset{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']\ C[\delta\gamma], a_{j+1}\dots a_n)\\ & \vdash (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']\ C[\delta\gamma]\ F'[\delta\gamma], a_{j+1}\dots a_n)\\ & \overset{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']\ C[\delta\gamma]\ F'[\delta\gamma]\ \xi_2\ D[\delta\gamma], a_{p+1}\dots a_n)\\ & \overset{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']\ C[\delta\gamma]\ F'[\delta\gamma]\ \xi_2\ D[\delta\gamma]\ E[\,], a_{q+1}\dots a_n)\\ & \overset{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']\ C[\delta\gamma]\ F[\,], a_{k+1}\dots a_n)\\ & \vdash (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']\ G[\,], a_{k+1}\dots a_n)\end{array}$$

$$\begin{array}{c}[A,h \mid C,j,\gamma,F,k,- \mid D,p,\gamma,E,q]\\ [A,h \mid B,i,\gamma',C,j,\gamma \mid -,-,-,-,-]\\ \hline [A,h \mid B,i,\gamma',G,k,- \mid D,p,\gamma,E,q]\end{array} \quad \begin{array}{l} C[\circ\circ] \longmapsto C[\circ\circ]\ F'[\circ\circ]\\ C[\circ\circ]\ F[\,] \longmapsto G[\,]\end{array}$$

– a una derivación de retorno:

$$\begin{array}{ll}(\xi\ A[\delta], a_{h+1}\dots a_n) & \overset{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma'], a_{i+1}\dots a_n)\\ & \overset{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']\ \xi_2\ D[\delta\eta], a_{p+1}\dots a_n)\\ & \overset{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']\ \xi_2\ D[\delta\eta]\ E[\,], a_{q+1}\dots a_n)\\ & \overset{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']\ C[\,], a_{j+1}\dots a_n)\\ & \vdash (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']\ C[\,]\ F'[\,], a_{j+1}\dots a_n)\\ & \overset{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']\ C[\,]\ F[\,], a_{k+1}\dots a_n)\\ & \vdash (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']\ G[\,], a_{k+1}\dots a_n)\end{array}$$

$$\begin{array}{c}[-,- \mid C,j,-,F,k,- \mid -,-,-,-,-]\\ [A,h \mid B,i,\gamma',C,j,- \mid D,p,\eta,E,q]\\ \hline [A,h \mid B,i,\gamma',G,k,- \mid D,p,\eta,E,q]\end{array} \quad \begin{array}{l} C[\circ\circ] \longmapsto C[\circ\circ]\ F'[\circ\circ]\\ C[\circ\circ]\ F[\,] \longmapsto G[\,]\end{array}$$

– a una derivación de puntos especiales:

$$\begin{array}{ll}(\xi\ B[\delta'\gamma'], a_{i+1}\dots a_n) & \vdash (\xi\ B[\delta'\gamma']\ D[\,], a_{i+1}\dots a_n)\\ & \overset{*}{\vdash} (\xi\ B[\delta'\gamma']\ C[\,], a_{j+1}\dots a_n)\\ & \vdash (\xi\ B[\delta'\gamma']\ C[\,]\ F'[\,], a_{j+1}\dots a_n)\\ & \overset{*}{\vdash} (\xi\ B[\delta'\gamma']\ C[\,]\ F[\,], a_{k+1}\dots a_n)\\ & \vdash (\xi\ B[\delta'\gamma']\ G[\,], a_{k+1}\dots a_n)\end{array}$$

$$\begin{array}{c}[-,- \mid C,j,-,F,k,- \mid -,-,-,-,-]\\ [-,- \mid B,i,\gamma',C,j,- \mid -,-,-,-,-]\\ \hline [-,- \mid B,i,\gamma',G,k,- \mid -,-,-,-,-]\end{array} \quad \begin{array}{l} C[\circ\circ] \longmapsto C[\circ\circ]\ F'[\circ\circ]\\ C[\circ\circ]\ F[\,] \longmapsto G[\,]\end{array}$$

- Derivaciones que son el resultado de aplicar una transición $C[\circ\circ]\ F[\,] \longmapsto G[\circ\circ]$ a una derivación obtenida tras aplicar la transición $C[\circ\circ] \longmapsto C[\circ\circ]\ F'[\circ\circ\gamma']$

– a una derivación de llamada:

$$\begin{array}{ll}(\xi\ A[\delta], a_{h+1}\dots a_n) & \\ & \overset{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma''], a_{i+1}\dots a_n)\\ & \overset{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma'']\ C[\delta\gamma], a_{j+1}\dots a_n)\\ & \vdash (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma'']\ C[\delta\gamma]\ F'[\delta\gamma\gamma'], a_{j+1}\dots a_n)\\ & \overset{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma'']\ C[\delta\gamma]\ F'[\delta\gamma\gamma']\ \xi_2\ D[\delta\gamma\gamma'], a_{p+1}\dots a_n)\\ & \overset{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma'']\ C[\delta\gamma]\ F'[\delta\gamma\gamma']\ \xi_2\ D[\delta\gamma\gamma']\ \xi_3\ O[\delta\gamma], a_{u+1}\dots a_n)\\ & \overset{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma'']\ C[\delta\gamma]\ F'[\delta\gamma\gamma']\ \xi_2\ D[\delta\gamma\gamma']\ \xi_3\ O[\delta\gamma]\ P[\,], a_{v+1}\dots a_n)\\ & \overset{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma'']\ C[\delta\gamma]\ F'[\delta\gamma\gamma']\ \xi_2\ D[\delta\gamma\gamma']\ E[\,], a_{q+1}\dots a_n)\\ & \overset{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma'']\ C[\delta\gamma]\ F[\,], a_{k+1}\dots a_n)\\ & \vdash (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma'']\ G[\,], a_{k+1}\dots a_n)\end{array}$$

$$\frac{\begin{array}{l}[C,j \mid C,j,\gamma,F,k,- \mid D,p,\gamma',E,q] \\ [A,h \mid B,i,\gamma'',C,j,\gamma \mid -,-,-,-,-] \\ [A,h \mid D,p,\gamma',E,q,- \mid O,u,\gamma,P,v]\end{array}}{[A,h \mid B,i,\gamma'',G,k,- \mid O,u,\gamma,P,v]} \quad \begin{array}{l} C[\circ\circ] \longmapsto C[\circ\circ]\ F'[\circ\circ\gamma'] \\ C[\circ\circ]\ F[\,] \longmapsto G[\,]\end{array}$$

– a una derivación de retorno:

$$\begin{array}{ll}(\xi\ A[\delta], a_{h+1}\dots a_n) & \overset{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma''], a_{i+1}\dots a_n) \\ & \overset{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma'']\ \xi_2\ D[\delta\eta], a_{p+1}\dots a_n) \\ & \overset{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma'']\ \xi_2\ D[\delta\eta]\ E[\,], a_{q+1}\dots a_n) \\ & \overset{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']\ C[\,], a_{j+1}\dots a_n) \\ & \vdash (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']\ C[\,]\ F'[\gamma'], a_{j+1}\dots a_n) \\ & \overset{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']\ C[\,]\ F'[\gamma']\ \xi_2\ O[\gamma'], a_{u+1}\dots a_n) \\ & \overset{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']\ C[\,]\ F'[\gamma']\ \xi_2\ O[\gamma']\ P[\,], a_{v+1}\dots a_n) \\ & \overset{*}{\vdash} (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']\ C[\,]\ F[\,], a_{k+1}\dots a_n) \\ & \vdash (\xi\ A[\delta]\ \xi_1\ B[\delta'\gamma']\ G[\,], a_{k+1}\dots a_n)\end{array}$$

$$\frac{\begin{array}{l}[C,j \mid C,j,-,F,k,- \mid O,u,\gamma',P,v] \\ [A,h \mid B,i,\gamma'',C,j,- \mid D,p,\eta,E,q] \\ [-,- \mid O,u,\gamma',P,v,- \mid -,-,-,-,-]\end{array}}{[A,h \mid B,i,\gamma'',G,k,- \mid D,p,\eta,E,q]} \quad \begin{array}{l} C[\circ\circ] \longmapsto C[\circ\circ]\ F'[\circ\circ\gamma'] \\ C[\circ\circ]\ F[\,] \longmapsto G[\,]\end{array}$$

– a una derivación de puntos especiales:

$$\begin{array}{ll}(\xi\ B[\delta'\gamma''], a_{i+1}\dots a_n) & \vdash (\xi\ B[\delta'\gamma'']\ D[\,], a_{i+1}\dots a_n) \\ & \overset{*}{\vdash} (\xi\ B[\delta'\gamma'']\ C[\,], a_{j+1}\dots a_n) \\ & \vdash (\xi\ B[\delta'\gamma'']\ C[\,]\ F'[\gamma'], a_{j+1}\dots a_n) \\ & \overset{*}{\vdash} (\xi\ B[\delta'\gamma'']\ C[\,]\ F'[\gamma']\ \xi_1\ O[\gamma'], a_{u+1}\dots a_n) \\ & \overset{*}{\vdash} (\xi\ B[\delta'\gamma'']\ C[\,]\ F'[\gamma']\ \xi_1\ O[\gamma']\ P[\,], a_{v+1}\dots a_n) \\ & \overset{*}{\vdash} (\xi\ B[\delta'\gamma'']\ C[\,]\ F[\,], a_{k+1}\dots a_n) \\ & \vdash (\xi\ B[\delta'\gamma'']\ G[\,], a_{k+1}\dots a_n)\end{array}$$

$$\frac{\begin{array}{l}[C,j \mid C,j,-,F,k,- \mid O,u,\gamma',P,v] \\ [-,- \mid B,i,\gamma'',C,j,- \mid -,-,-,-,-] \\ [-,- \mid O,u,\gamma',P,v,- \mid -,-,-,-,-]\end{array}}{[A,h \mid B,i,\gamma'',G,k,- \mid D,p,\eta,E,q]} \quad \begin{array}{l} C[\circ\circ] \longmapsto C[\circ\circ]\ F'[\circ\circ\gamma'] \\ C[\circ\circ]\ F[\,] \longmapsto G[\,]\end{array}$$

- Derivaciones que son el resultado de aplicar una transición $C[\circ\circ]\ F[\,] \longmapsto G[\circ\circ]$ a una derivación obtenida tras aplicar la transición $C[\circ\circ\gamma] \longmapsto C[\circ\circ\gamma]\ F'[\circ\circ]$ a una derivación de llamada, con los tres casos siguientes:
  – la derivación de llamada es a su vez derivada a partir de una derivación de llamada:

$$\begin{array}{l}(\xi\ M[\delta], a_{m+1}\dots a_n) \\ \overset{*}{\vdash} (\xi\ M[\delta]\ N[\delta''\gamma'''], a_{t+1}\dots a_n) \\ \overset{*}{\vdash} (\xi\ M[\delta]\ N[\delta''\gamma''']\ A[\delta\gamma'], a_{h+1}\dots a_n) \\ \overset{*}{\vdash} (\xi\ M[\delta]\ N[\delta''\gamma''']\ A[\delta\gamma']\ \xi_1\ B[\delta'\gamma''], a_{i+1}\dots a_n) \\ \overset{*}{\vdash} (\xi\ M[\delta]\ N[\delta''\gamma''']\ A[\delta\gamma']\ \xi_1\ B[\delta'\gamma'']\ C[\delta\gamma'\gamma], a_{j+1}\dots a_n) \\ \vdash (\xi\ M[\delta]\ N[\delta''\gamma''']\ A[\delta\gamma']\ \xi_1\ B[\delta'\gamma'']\ C[\delta\gamma'\gamma]\ F'[\delta\gamma'], a_{j+1}\dots a_n) \\ \overset{*}{\vdash} (\xi\ M[\delta]\ N[\delta''\gamma''']\ A[\delta\gamma']\ \xi_1\ B[\delta'\gamma'']\ C[\delta\gamma'\gamma]\ F'[\delta\gamma']\ \xi_2\ D[\delta\gamma'], a_{p+1}\dots a_n) \\ \overset{*}{\vdash} (\xi\ M[\delta]\ N[\delta''\gamma''']\ A[\delta\gamma']\ \xi_1\ B[\delta'\gamma'']\ C[\delta\gamma'\gamma]\ F'[\delta\gamma']\ \xi_2\ D[\delta\gamma']\ E[\,], a_{q+1}\dots a_n) \\ \overset{*}{\vdash} (\xi\ M[\delta]\ N[\delta''\gamma''']\ A[\delta\gamma']\ \xi_1\ B[\delta'\gamma'']\ C[\delta\gamma'\gamma]\ F[\,], a_{k+1}\dots a_n) \\ \vdash (\xi\ M[\delta]\ N[\delta''\gamma''']\ A[\delta\gamma']\ \xi_1\ B[\delta'\gamma'']\ G[\,], a_{k+1}\dots a_n)\end{array}$$

$$\frac{\begin{array}{l}[M,m \mid C,j,\gamma,F,k,- \mid D,p,\gamma',E,q] \\ [A,h \mid B,i,\gamma'',C,j,\gamma \mid -,-,-,-,-] \\ [M,m \mid N,t,\gamma''',A,h,\gamma' \mid -,-,-,-,-]\end{array}}{[A,h \mid B,i,\gamma'',G,k,- \mid C,j,\gamma,F,k]} \quad \begin{array}{l} C[\circ\circ\gamma] \longmapsto C[\circ\circ\gamma]\ F'[\circ\circ] \\ C[\circ\circ]\ F[\,] \longmapsto G[\,]\end{array}$$

– la derivación de llamada es a su vez derivada a partir de una derivación de retorno:

$$\begin{array}{ll}(\xi\ M[\delta], a_{m+1}\ldots a_n) & \overset{*}{\vdash} (\xi\ M[\delta]\ N[\delta''\gamma'''], a_{t+1}\ldots a_n) \\ & \overset{*}{\vdash} (\xi\ M[\delta]\ N[\delta''\gamma''']\ \xi_1\ D[\delta\eta], a_{p+1}\ldots a_n) \\ & \overset{*}{\vdash} (\xi\ M[\delta]\ N[\delta''\gamma''']\ \xi_1\ D[\delta\eta]\ E[\,], a_{q+1}\ldots a_n) \\ & \overset{*}{\vdash} (\xi\ M[\delta]\ N[\delta''\gamma''']\ A[\,], a_{h+1}\ldots a_n) \\ & \overset{*}{\vdash} (\xi\ M[\delta]\ N[\delta''\gamma''']\ A[\,]\ \xi_2\ B[\delta'\gamma''], a_{i+1}\ldots a_n) \\ & \overset{*}{\vdash} (\xi\ M[\delta]\ N[\delta''\gamma''']\ A[\,]\ \xi_2\ B[\delta'\gamma'']\ C[\gamma], a_{j+1}\ldots a_n) \\ & \vdash (\xi\ M[\delta]\ N[\delta''\gamma''']\ A[\,]\ \xi_2\ B[\delta'\gamma'']\ C[\gamma]\ F'[\,], a_{j+1}\ldots a_n) \\ & \overset{*}{\vdash} (\xi\ M[\delta]\ N[\delta''\gamma''']\ A[\,]\ \xi_2\ B[\delta'\gamma'']\ C[\gamma]\ F[\,], a_{k+1}\ldots a_n) \\ & \vdash (\xi\ M[\delta]\ N[\delta''\gamma''']\ A[\,]\ \xi_2\ B[\delta'\gamma'']\ G[\,], a_{k+1}\ldots a_n)\end{array}$$

$$\frac{\begin{array}{l}[-,- \mid C,j,\gamma,F,k,- \mid -,-,-,-,-] \\ [A,h \mid B,i,\gamma'',C,j,\gamma \mid -,-,-,-,-] \\ [M,m \mid N,t,\gamma'''A,h,- \mid D,p,\eta,E,q]\end{array}}{[A,h \mid B,i,\gamma'',G,k,- \mid C,j,\gamma,F,k} \quad \begin{array}{l} C[\circ\circ\gamma] \longmapsto C[\circ\circ\gamma]\ F'[\circ\circ] \\ C[\circ\circ]\ F[\,] \longmapsto G[\,]\end{array}$$

– la derivación de llamada es a su vez derivada a partir de una derivación de puntos especiales:

$$\begin{array}{ll}(\xi\ N[\delta''\gamma'''], a_{t+1}\ldots a_n) & \vdash (\xi\ N[\delta''\gamma''']\ D[\,], a_{t+1}\ldots a_n) \\ & \overset{*}{\vdash} (\xi\ N[\delta''\gamma''']\ A[\,], a_{h+1}\ldots a_n) \\ & \overset{*}{\vdash} (\xi\ N[\delta''\gamma''']\ A[\,]\ \xi_2\ B[\delta'\gamma''], a_{i+1}\ldots a_n) \\ & \overset{*}{\vdash} (\xi\ N[\delta''\gamma''']\ A[\,]\ \xi_2\ B[\delta'\gamma'']\ C[\gamma], a_{j+1}\ldots a_n) \\ & \vdash (\xi\ N[\delta''\gamma''']\ A[\,]\ \xi_2\ B[\delta'\gamma'']\ C[\gamma]\ F'[\,], a_{j+1}\ldots a_n) \\ & \overset{*}{\vdash} (\xi\ N[\delta''\gamma''']\ A[\,]\ \xi_2\ B[\delta'\gamma'']\ C[\gamma]\ F[\,], a_{k+1}\ldots a_n) \\ & \vdash (\xi\ N[\delta''\gamma''']\ A[\,]\ \xi_2\ B[\delta'\gamma'']\ G[\,], a_{k+1}\ldots a_n)\end{array}$$

$$\frac{\begin{array}{l}[-,- \mid C,j,\gamma,F,k,- \mid -,-,-,-,-] \\ [A,h \mid B,i,\gamma'',C,j,\gamma \mid -,-,-,-,-] \\ [-,- \mid N,t,\gamma'''A,h,- \mid -,-,-,-,-]\end{array}}{[-,- \mid B,i,\gamma'',G,k,- \mid -,-,-,-,-]} \quad \begin{array}{l} C[\circ\circ\gamma] \longmapsto C[\circ\circ\gamma]\ F'[\circ\circ] \\ C[\circ\circ]\ F[\,] \longmapsto G[\,]\end{array}$$

Si aplicamos inducción en la longitud de una derivación, a partir de la lista observamos que para cualquier derivación obtenida mediante la aplicación de una transición, existe una regla de combinación que busca los ítems correspondientes a las subderivaciones relevantes y que produce el ítem correspondiente a la derivación resultante. También podemos observar que dada una regla de combinación de ítems, existen subderivaciones en el autómata que se corresponden con cada uno de los ítems antecedentes y que combinadas entre sí producen la derivación correspondiente al ítem consecuente. □

La complejidad espacial de la técnica de tabulación con respecto a la longitud n de la cadena de entrada es $\mathcal{O}(n^5)$ puesto que cada ítem almacena 5 posiciones de la cadena de entrada. La complejidad temporal es inicialmente $\mathcal{O}(n^7)$. Esta complejidad viene dada por la regla

$$\frac{\begin{array}{l}[C,j \mid C,j,\gamma,F,k,- \mid D,p,\gamma',E,q] \\ [A,h \mid B,i,\gamma'',C,j,\gamma \mid -,-,-,-,-] \\ [A,h \mid D,p,\gamma',E,q,- \mid O,u,\gamma,P,v]\end{array}}{[A,h \mid B,i,\gamma'',G,k,- \mid O,u,\gamma,P,v]} \quad \begin{array}{l} C[\circ\circ] \longmapsto C[\circ\circ]\ F'[\circ\circ\gamma'] \\ C[\circ\circ]\ F[\,] \longmapsto G[\,]\end{array}$$

Utilizando la misma técnica que para las estrategias $*$-Earley podemos dividir dicha regla en las dos reglas de combinación de ítems siguientes:

$$\frac{\begin{array}{c}[C,j \mid C,j,\gamma,F,k,- \mid D,p,\gamma',E,q] \\ [A,h \mid D,p,\gamma',E,q,- \mid O,u,\gamma,P,v]\end{array}}{[[C,j,\gamma,F,k,- \mid O,u,\gamma,P,v]]} \quad \begin{array}{l} C[\circ\circ] \longmapsto C[\circ\circ]\ F'[\circ\circ\gamma'] \\ C[\circ\circ]\ F[\,] \longmapsto G[\,]\end{array}$$

$$\frac{\begin{array}{l}[[C,j,\gamma,F,k,- \mid O,u,\gamma,P,v]] \\ [A,h \mid B,i,\gamma'',C,j,\gamma \mid -,-,-,-,-] \\ [A,h \mid D,p,\gamma',E,q,- \mid O,u,\gamma,P,v]\end{array}}{[A,h \mid B,i,\gamma'',G,k,- \mid O,u,\gamma,P,v]} \quad \begin{array}{l} C[\circ\circ] \longmapsto C[\circ\circ]\ F'[\circ\circ\gamma'] \\ C[\circ\circ]\ F[\,] \longmapsto G[\,]\end{array}$$

donde $[[C,j,\gamma,F,k,- \mid O,u,\gamma,P,v]]$ es un pseudo-ítem encargado de trasmitir cierta información de la primera a la segunda regla. La diferencia con una aplicación parcial es que la primera regla ignora los elementos A y h en el pseudo-ítem generado, puesto que tales elementos son posteriormente recuperados del segundo y tercer ítem que intervienen en la segunda regla, que junto con el pseudo-ítem son suficientes para garantizar la existencia del ítem $[C,j \mid C,j,\gamma,F,k,- \mid D,p,\gamma',E,q]$ por la definición de derivaciones de llamada y retorno. La primera regla presenta una complejidad temporal $\mathcal{O}(n^6)$ (la posición h no interviene) y la segunda presenta también una complejidad $\mathcal{O}(n^6)$ (las posiciones p y q no intervienen) por lo que hemos logrado rebajar la complejidad temporal final de la técnica de tabulación a $\mathcal{O}(n^6)$.

Capítulo 9

Autómatas lineales de índices

En este capítulo se describen dos tipos de autómata para los lenguajes de adjunción de árboles, los autómatas lineales de índices derechos e izquierdos, junto con sus correspondientes técnicas de tabulación, y se define la clase de los autómatas lineales de índices fuertemente dirigidos, que permiten definir estrategias de diferentes tipos para gramáticas lineales de índices y para gramáticas de adjunción de árboles, incluyendo estrategias de tipo Earley. Este nuevo modelo de autómata, junto con su técnica de tabulación, constituye la principal aportación de este capítulo. Las ideas de este capítulo han dado lugar a [20]

9.1 Introducción

Los autómatas lineales de índices (*Linear Indexed Automata*, LIA) [124, 126] son una extensión de los autómatas a pila en la cual cada símbolo de pila tiene asociado una pila de índices. Formalmente, un autómata lineal de índices es una tupla $(V_T, V_S, \$_0, \$_f, V_I, \mathcal{T})$, donde:

- V_T es un conjunto finito de símbolos terminales.
- V_S es un conjunto finito de símbolos de pila.
- $\$_0 \in V_S$ es el símbolo inicial de la pila.
- $\$_f \in V_S$ es el símbolos final de pila.
- V_I es un conjunto finito de índices.
- $\mathcal{T}$ es un conjunto finito de transiciones, que pueden ser de alguno de los siguientes tipos:
  - $C[\circ\circ\gamma] \stackrel{a}{\longmapsto} F[\circ\circ\gamma']$
  - $C[\circ\circ] \stackrel{a}{\longmapsto} F[\circ\circ]\ G[\,]$
  - $C[\circ\circ\gamma] \stackrel{a}{\longmapsto} F[\,]\ G[\circ\circ\gamma']$
  - $F[\circ\circ\gamma]\ G[\,] \stackrel{a}{\longmapsto} C[\circ\circ\gamma']$
  - $F[\,]\ G[\circ\circ\gamma] \stackrel{a}{\longmapsto} C[\circ\circ\gamma']$

  donde $C, F, G \in V_S$, $a \in V_T \cup \{\epsilon\}$, $\gamma, \gamma' \in V_I \cup \{\epsilon\}$ y en cada transición, bien $\gamma = \epsilon$, bien $\gamma' = \epsilon$ o bien $\gamma = \gamma' = \epsilon$.

Dada una transición, decimos que el elemento de la parte derecha que comparte la pila de índices con el elemento de la parte izquierda es su *hijo dependiente*. La relación de *descendiente dependiente* es el cierre reflexivo y transitivo de la relación de hijo dependiente.

Una *configuración* de un autómata lineal de índices es un par (Υ, w), donde $\Upsilon \in (V_S[V_I^*])^*$ y $w \in V_T^*$, tal que Υ representa el contenido de la pila del autómata y w representa la parte de la cadena de entrada que resta por leer. Una configuración (Υ_1, aw) deriva una configuración (Υ_2, w), denotado mediante $(\Upsilon_1, aw) \vdash (\Upsilon_2, w)$ si y sólo si existe una transición que transforma la pila Υ_1 en la pila Υ_2 leyendo $a \in V_T \cup \{\epsilon\}$ de la cadena de entrada. En caso de ser necesario identificar una derivación d concreta, utilizaremos la notación $\vdash_d$. Denotamos por $\overset{*}{\vdash}$ el cierre reflexivo y transitivo de $\vdash$. Decimos que una cadena de entrada w es aceptada por un autómata lineal de índices si $(\$_0[\,], w) \overset{*}{\vdash} (\$_0[\,]\ \$_f[\,], \epsilon)$.

Observamos una gran similitud entre la definición de los autómatas lineales de índices y la de los autómatas lógicos a pila restringidas del capítulo 8. Al igual que ocurría en el caso de los RLPDA, diferentes juegos de instrucciones nos van a permitir diferentes clases de autómatas, que nos permitirán definir determinadas estrategias de análisis en lo que respecta a la manipulación de las pilas de índices. En las secciones que siguen se describe la clase de los *autómatas lineales de índices orientados a la derecha*, que permiten definir estrategias en las cuales las pilas de índices se construyen de modo ascendente, de los *autómatas lineales de índices orientados a la izquierda*, en los cuales las pilas de índices se construyen de modo descendente, para finalizar con una clase de autómatas lineales de índices más general que permite la descripción de estrategias mixtas con ciertas restricciones.

9.2 Autómatas lineales de índices orientados a la derecha

Dada una gramática lineal de índices, una estrategia que construya de modo ascendente las pilas de índices ignorará el contenido de dichas pilas durante su contacto con la *izquierda* de cada espina en la fase descendente, mientras que durante la fase ascendente irá calculando el contenido de las pilas de índices cada vez que visite la *derecha* de una espina, tal y como se muestra en la figura 9.1. Este tipo de estrategias son precisamente las que se pueden implementar en los autómatas lineales de índices orientados a la derecha (*Right-oriented Linear Indexed Automata*, R–LIA) [124, 126], de ahí su nombre.

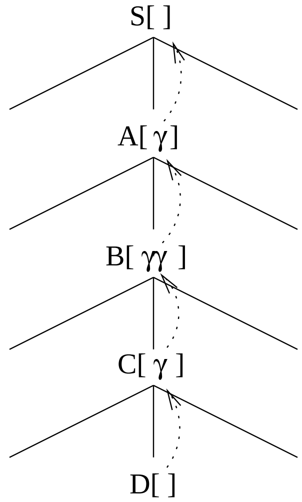

Figura 9.1: Construcción *orientada a la derecha* de las pilas de índices

Inicialmente, los autómatas lineales de índices orientados a la derecha fueron definidos por Nederhof en [124] como una clase de los autómatas lineales de índices en la cual el juego de transiciones se restringe de tal modo que sólo los siguientes tipos de transiciones están disponibles:

- $C[\circ\circ\gamma] \overset{a}{\longmapsto} F[\circ\circ\gamma']$
- $C[\circ\circ] \overset{a}{\longmapsto} C[\circ\circ]\ F[\,]$
- $C[\circ\circ\gamma]\ F[\,] \overset{a}{\longmapsto} G[\circ\circ\gamma']$
- $C[\,]\ F[\circ\circ\gamma] \overset{a}{\longmapsto} G[\circ\circ\gamma']$

Posteriormente, Nederhof redefine en [126] este tipo de autómatas, cambiando los tipos de transiciones permitidas, que pasan a ser los siguientes:

- $C[\circ\circ\gamma] \longmapsto F[\circ\circ\gamma']$
- $C[\circ\circ] \overset{a}{\longmapsto} F[\circ\circ]\ G[\,]$
- $F[\circ\circ]\ G[\,] \overset{a}{\longmapsto} C[\circ\circ]$
- $F[\,]\ G[\circ\circ] \longmapsto C[\circ\circ]$

El primer conjunto de transiciones está incluido en el segundo puesto que

- Una transición $C[\circ\circ\gamma] \overset{a}{\longmapsto} F[\circ\circ\gamma']$ puede ser emulada mediante la aplicación consecutiva de las tres transiciones $C[\circ\circ\gamma] \longmapsto F'[\circ\circ\gamma']$, $F'[\circ\circ] \overset{a}{\longmapsto} F'[\circ\circ]\ F''[\,]$ y $F'[\circ\circ]\ F''[\,] \longmapsto F[\circ\circ]$.
- Una transición $C[\circ\circ\gamma]\ F[\,] \overset{a}{\longmapsto} G[\circ\circ\gamma']$ puede emularse mediante la aplicación consecutiva de las transiciones $C[\circ\circ]\ F[\,] \overset{a}{\longmapsto} G'[\circ\circ]$ y $G'[\circ\circ\gamma] \longmapsto G[\circ\circ\gamma']$.
- Una transición $C[\,]\ F[\circ\circ\gamma] \overset{a}{\longmapsto} G[\circ\circ\gamma']$ puede emularse mediante la aplicación consecutiva de las transiciones $C[\,]\ F[\circ\circ] \longmapsto G'[\circ\circ]$, $G'[\circ\circ\gamma] \longmapsto G''[\circ\circ\gamma']$, $G''[\circ\circ] \overset{a}{\longmapsto} G''[\circ\circ]\ G'''[\,]$ y $G''[\circ\circ]\ G'''[\,] \longmapsto G[\circ\circ]$.

Por otra parte, el segundo conjunto de transiciones está incluido en el primero al poderse emular una transición $C[\circ\circ] \overset{a}{\longmapsto} F[\circ\circ]\ G[\,]$ mediante la aplicación consecutiva de las transiciones $C[\circ\circ] \overset{a}{\longmapsto} F'[\circ\circ]$ y $F'[\circ\circ] \longmapsto F'[\circ\circ]\ G[\,]$, la adición de una transición $F'[\circ\circ]\ G[\,] \overset{b}{\longmapsto} K[\circ\circ]$ por cada transición $F[\circ\circ]\ G[\,] \overset{b}{\longmapsto} K[\circ\circ]$ presente en el autómata y la adición de una transición $F'[\,]\ G[\circ\circ] \longmapsto K[\circ\circ]$ por cada transición $F[\,]\ G[\circ\circ] \longmapsto K[\circ\circ]$ presente en el autómata. En todos los casos, dado $X \in V_S$, se ha utilizado X', X'' y X''' para denotar nuevos símbolos de pila que no aparecen en ninguna otra transición del autómata.

En consecuencia, ambas definiciones de los autómatas lineales de índices orientados a la derecha son equivalentes. En este capítulo, utilizaremos la primera puesto que es la que mejor se adapta a la definición de esquemas de tabulación para LIG y TAG.

9.2.1 Esquemas de compilación

Las estrategias de análisis sintáctico que pueden ser implantadas en un autómata lineal de índices orientados a la derecha están limitadas por la forma de las transiciones permitidas, ya que sólo se puede apilar un elemento que tenga asociado una pila de índices vacía. Puesto que en un autómata a pila la información que se transmite en una operación de apilamiento representa la información propagada durante la fase descendente de un algoritmo de análisis, los autómatas lineales de índices orientados a la derecha no permiten definir estrategias de análisis descendentes con respecto a los contenidos de las pilas de índices. Esto es, las predicciones deben ser puramente independientes del contexto. Esta limitación se corresponde exactamente con la presente en los autómatas lógicos a pila restringidos que incorporan estrategias $*$-ascendentes, descritos en la sección 8.1. Si unimos este hecho a que el juego de transiciones de los RLPDA $*$-ascendentes coincide con el de los R–LIA, tenemos que los esquemas de compilación coincidirán en ambos casos, por lo que nos limitaremos a mostrar dos esquemas de compilación genéricos de LIG y TAG en R–LIA.

Esquema de compilación 9.1 El esquema de compilación genérico de una gramática lineal de índices en un autómata lineal de índices orientado a la derecha queda definido por el conjunto de reglas mostrado en la tabla 9.1 y por los elementos inicial $\$_0[\,]$ y final $\overleftarrow{S}[\,]$. §

Esquema de compilación 9.2 El esquema de compilación genérico de una gramática de adjunción de árboles en un autómata lógico a pila restringido queda definido por el conjunto de reglas mostrado en la tabla 9.2 y por los elementos inicial $\$_0[\,]$ y final $\overleftarrow{\top\alpha}[\,]$, con $\alpha \in \boldsymbol{I}$. §

9.2.2 Tabulación

Puesto que el juego de transiciones y la forma de las derivaciones es la misma en los RLPDA $*$-ascendentes y los R–LIA, la técnica de tabulación desarrollada para los primeros en la sección 8.5.1 es aplicable a los segundos.

9.3 Autómatas lineales de índices orientados a la izquierda

Dada una gramática lineal de índices, decimos que una estrategia construye las pilas de índices de modo descendente si calcula el contenido de las mismas cuando visita la izquierda de los elementos de las espinas, mientras que ignorará el contenido de dichas pilas cuando visite la derecha de dichos elementos, tal y como se muestra en la figura 9.2. Este tipo de estrategias son precisamente las que se pueden implementar en los autómatas lineales de índices orientados a la izquierda (*Left-oriented Linear Indexed Automata, L–LIA*) [126], de ahí su nombre.

Los autómatas lineales de índices orientados a la izquierda fueron introducidos por Nederhof en [126] como una clase de autómatas lineales de índices en la cual los tipos de transiciones disponibles se limitan a los siguientes:

- $C[\circ\circ\gamma] \longmapsto F[\circ\circ\gamma']$
- $C[\circ\circ] \overset{a}{\longmapsto} F[\circ\circ]\ G[\,]$
- $C[\circ\circ] \longmapsto F[\,]\ G[\circ\circ]$
- $F[\circ\circ]\ G[\,] \overset{a}{\longmapsto} C[\circ\circ]$

En este capítulo proponemos un juego de transiciones ligeramente más restrictivo, según el cual admitimos como válidas transiciones pertenecientes a los tipos siguientes:

- $C[\circ\circ] \overset{a}{\longmapsto} F[\circ\circ]$

[INIT]	$\$_0[\circ\circ] \longmapsto \$_0[\circ\circ]\ \nabla_{0,0}[\,]$	
[CALL]	$\nabla_{r,s}[\circ\circ] \longmapsto \nabla_{r,s}[\circ\circ]\ \overrightarrow{A_{r,s+1}}[\,]$	$A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\,]\Upsilon_2$
[SCALL]	$\nabla_{r,s}[\circ\circ] \longmapsto \nabla_{r,s}[\circ\circ]\ \overrightarrow{A_{r,s+1}}[\,]$	$A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\circ\circ\gamma']\Upsilon_2$
[SEL]	$\overrightarrow{A_{r,0}}[\circ\circ] \longmapsto \nabla_{r,0}[\circ\circ]$	$r \neq 0$
[PUB]	$\nabla_{r,n_r}[\circ\circ] \longmapsto \overleftarrow{A_{r,0}}[\circ\circ]$	
[RET]	$\nabla_{r,s}[\circ\circ]\ \overleftarrow{A_{r,s+1}}[\,] \longmapsto \nabla_{r,s+1}[\circ\circ]$	$A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\,]\Upsilon_2$
[SRET]	$\nabla_{r,s}[\,]\ \overleftarrow{A_{r,s+1}}[\circ\circ\gamma'] \longmapsto \nabla_{r,s+1}[\circ\circ\gamma]$	$A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\circ\circ\gamma']\Upsilon_2$
[SCAN]	$\overrightarrow{A_{r,0}}[\circ\circ] \overset{a}{\longmapsto} \overleftarrow{A_{r,0}}[\circ\circ]$	$A_{r,0}[\,] \to a$

Tabla 9.1: Reglas del esquema de compilación genérico de LIG en R–LIA

[INIT]	$\$_0[\circ\circ] \longmapsto \$_0[\circ\circ]\ \nabla^{\alpha}_{0,0}[\,]$	$\alpha \in \boldsymbol{I}$
[CALL]	$\nabla^{\gamma}_{r,s}[\circ\circ] \longmapsto \nabla^{\gamma}_{r,s}[\circ\circ]\ \overrightarrow{N^{\gamma}_{r,s+1}}[\,]$	$N^{\gamma}_{r,s+1} \notin \text{espina}(\gamma),\ \mathbf{nil} \in \text{adj}(N_{r,s+1})$
[SCALL]	$\nabla^{\gamma}_{r,s}[\circ\circ] \longmapsto \nabla^{\gamma}_{r,s}[\circ\circ]\ \overrightarrow{N^{\gamma}_{r,s+1}}[\,]$	$N^{\gamma}_{r,s+1} \in \text{espina}(\gamma),\ \mathbf{nil} \in \text{adj}(N_{r,s+1})$
[SEL]	$\overrightarrow{N^{\gamma}_{r,0}}[\circ\circ] \longmapsto \nabla^{\gamma}_{r,0}[\circ\circ]$	$r \neq 0$
[PUB]	$\nabla^{\gamma}_{r,n_r}[\circ\circ] \longmapsto \overleftarrow{N^{\gamma}_{r,0}}[\circ\circ]$	
[RET]	$\nabla^{\gamma}_{r,s}[\circ\circ]\ \overleftarrow{N^{\gamma}_{r,s+1}}[\,] \longmapsto \nabla^{\gamma}_{r,s+1}[\circ\circ]$	$N^{\gamma}_{r,s+1} \notin \text{espina}(\gamma),\ \mathbf{nil} \in \text{adj}(N_{r,s+1})$
[SRET]	$\nabla^{\gamma}_{r,s}[\circ\circ]\ \overleftarrow{N^{\gamma}_{r,s+1}}[\circ\circ] \longmapsto \nabla^{\gamma}_{r,s+1}[\circ\circ]$	$N^{\gamma}_{r,s+1} \in \text{espina}(\gamma),\ \mathbf{nil} \in \text{adj}(N_{r,s+1})$
[SCAN]	$\overrightarrow{N^{\gamma}_{r,0}}[\circ\circ] \overset{a}{\longmapsto} \overleftarrow{N^{\gamma}_{r,0}}[\circ\circ]$	$N^{\gamma}_{r,0}[\,] \to a$
[ACALL]	$\nabla^{\gamma}_{r,s}[\circ\circ] \longmapsto \nabla^{\gamma}_{r,s}[\circ\circ]\ \overrightarrow{\top^{\beta}}[\,]$	$\beta \in \text{adj}(N^{\gamma}_{r,s+1})$
[ARET]	$\nabla^{\gamma}_{r,s}[\,]\ \overleftarrow{\top^{\beta}}[\circ\circ N^{\gamma}_{r,s+1}] \longmapsto \nabla^{\gamma}_{r,s+1}[\circ\circ]$	$\beta \in \text{adj}(N^{\gamma}_{r,s+1})$
[FCALL]	$\nabla^{\beta}_{f,0}[\circ\circ] \longmapsto \nabla^{\beta}_{f,0}[\circ\circ]\ \overrightarrow{N^{\gamma}_{r,s+1}}[\,]$	$N^{\beta}_{f,0} = \mathbf{F}^{\beta},\ \beta \in \text{adj}(N^{\gamma}_{r,s+1})$
[FRET]	$\nabla^{\beta}_{f,0}[\,]\ \overleftarrow{N^{\gamma}_{r,s+1}}[\circ\circ] \longmapsto \nabla^{\beta}_{f,1}[\circ\circ N^{\gamma}_{r,s+1}]$	$N^{\beta}_{f,0} = \mathbf{F}^{\beta},\ \beta \in \text{adj}(N^{\gamma}_{r,s+1})$

Tabla 9.2: Reglas del esquema de compilación genérico de TAG en R–LIA

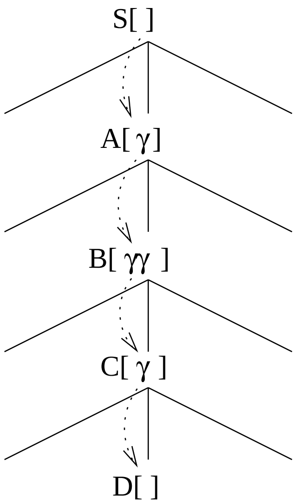

Figura 9.2: Construcción *orientada a la izquierda* de las pilas de índices

- $C[\circ\circ] \longmapsto C[\circ\circ]\ F[\,]$
- $C[\circ\circ\gamma] \longmapsto C[\,]\ F[\circ\circ\gamma']$
- $C[\circ\circ]\ F[\,] \longmapsto G[\circ\circ]$

El segundo juego de transiciones está incluido en el primero, puesto que:

- Una transición $C[\circ\circ] \overset{a}{\longmapsto} F[\circ\circ]$ puede ser emulada mediante la aplicación consecutiva de las transiciones $C[\circ\circ] \longmapsto F'[\circ\circ]$, $F'[\circ\circ] \overset{a}{\longmapsto} F'[\circ\circ]\ F''[\,]$ y $F'[\circ\circ]\ F''[\,] \longmapsto F[\circ\circ]$.
- Una transición $C[\circ\circ\gamma] \longmapsto F[\,]\ G[\circ\circ\gamma']$ puede ser emulada mediante la aplicación consecutiva de las transiciones $C[\circ\circ\gamma] \longmapsto C'[\circ\circ\gamma']$ y $C'[\circ\circ] \longmapsto C[\,]\ G[\circ\circ]$.

Sin embargo, una transición $C[\circ\circ\gamma] \longmapsto F[\circ\circ\gamma']$ en la que $\gamma \neq \epsilon$ o $\gamma' \neq \epsilon$ no puede ser emulada mediante las transiciones del segundo juego. La razón por la cual hemos decidido trabajar con el segundo juego es que, a pesar de ser menos potente, es más adecuado para definir los esquemas de compilación de LIG y TAG. Este hecho no es extraño si consideramos la estrecha relación existente entre las transiciones propuestas para L–LIA y las transiciones utilizadas por los RLP-DA $*$-descendentes del capítulo 8. A este respecto, las transiciones de tipo $C[\circ\circ]\ F[\,] \longmapsto G[\,]$ no son necesarias debido a que las transiciones $C[\circ\circ\gamma] \longmapsto C[\,]\ G[\circ\circ\gamma']$ permiten indicar de forma explícita que la pila de índices asociada a C está vacía.

9.3.1 Esquemas de compilación de gramáticas lineales de índices

Como se ha establecido anteriormente, los tipos de transiciones utilizados en los autómatas lineales de índices orientados a la izquierda sólo permiten implantar estrategias en las cuales las pilas de índices son construidas en la fase descendente. En cambio, no establecen limitaciones sobre la estrategia utilizada con respecto al esqueleto independiente del contexto. Sacando provecho de esta circunstancia, definiremos un esquema de compilación genérico, que contempla los parámetros

- $\overrightarrow{A}$, que se refiere a la predicción realizada sobre el no-terminal A durante la fase descendente de la estrategia de análisis.

L–LIA	**RLPDA $*$-descendentes**
$C[\circ\circ] \overset{a}{\longmapsto} F[\circ\circ]$	$C[\circ\circ] \overset{a}{\longmapsto} F[\circ\circ]$
$C[\circ\circ] \longmapsto C[\circ\circ]\ F[\,]$	$C[\circ\circ] \longmapsto C[\circ\circ]\ F[\,]$
$C[\circ\circ\gamma] \longmapsto C[\,]\ F[\circ\circ\gamma']$	$C[\circ\circ\gamma] \longmapsto C[\circ\circ\gamma]\ F[\circ\circ\gamma']$
$C[\circ\circ]\ F[\,] \longmapsto G[\circ\circ]$	$C[\circ\circ]\ F[\,] \longmapsto G[\circ\circ]$ $C[\circ\circ]\ F[\,] \longmapsto G[\,]$

Tabla 9.3: Transiciones de L–LIA y RLPDA $*$-descendentes

- $\overleftarrow{\overline{A}}$, que se refiere a la propagación de información respecto al no-terminal A durante la fase ascendente de la estrategia de análisis.

para a continuación definir los esquemas de compilación específicos para las estrategias ascendente-descendente, Earley-descendente y descendente-descendente.

Esquema de compilación 9.3 El esquema de compilación genérico de una gramática lineal de índices en un autómata lineal de índices orientado a la izquierda queda definido por el conjunto de reglas mostrado en la tabla 9.4 y por los elementos inicial $\$_0[\,]$ y final $\overleftarrow{\overline{S}}[\,]$. §

Estrategia ascendente-descendente

Esquema de compilación 9.4 El esquema de compilación ascendente-descendente de una gramática lineal de índices en un autómata lineal de índices orientado a la izquierda queda definido por el conjunto de reglas mostrado en la tabla 9.5 y por los elementos inicial $\$_0[\,]$ y final $S[\,]$. §

Estrategia Earley-descendente

Esquema de compilación 9.5 El esquema de compilación Earley-descendente de una gramática lineal de índices en un autómata lineal de índices orientado a la izquierda queda definido por el conjunto de reglas mostrado en la tabla 9.6 y por los elementos inicial $\$_0[\,]$ y final $\overline{\overline{S}}[\,]$. §

Estrategia descendente-descendente

Esquema de compilación 9.6 El esquema de compilación descendente-descendente de una gramática lineal de índices en un autómata lineal de índices orientado a la izquierda queda

[INIT]	$\$_0[\circ\circ] \longmapsto \$_0[\circ\circ]\ \nabla_{0,0}[\,]$	
[CALL]	$\nabla_{r,s}[\circ\circ] \longmapsto \nabla_{r,s}[\circ\circ]\ \overrightarrow{A_{r,s+1}}[\,]$	$A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\,]\Upsilon_2$
[SCALL]	$\nabla_{r,s}[\circ\circ\gamma] \longmapsto \nabla_{r,s}[\,]\ \overrightarrow{A_{r,s+1}}[\circ\circ\gamma']$	$A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\circ\circ\gamma']\Upsilon_2$
[SEL]	$\overrightarrow{A_{r,0}}[\circ\circ] \longmapsto \nabla_{r,0}[\circ\circ]$	$r \neq 0$
[PUB]	$\nabla_{r,n_r}[\circ\circ] \longmapsto \overleftarrow{A_{r,0}}[\circ\circ]$	
[RET]	$\nabla_{r,s}[\circ\circ]\ \overleftarrow{A_{r,s+1}}[\,] \longmapsto \nabla_{r,s+1}[\circ\circ]$	$A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\,]\Upsilon_2$
[SRET]	$\nabla_{r,s}[\circ\circ]\ \overleftarrow{A_{r,s+1}}[\,] \longmapsto \nabla_{r,s+1}[\circ\circ]$	$A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\circ\circ\gamma']\Upsilon_2$
[SCAN]	$\overrightarrow{A_{r,0}}[\,] \overset{a}{\longmapsto} \overleftarrow{A_{r,0}}[\,]$	$A_{r,0}[\,] \to a$

Tabla 9.4: Reglas del esquema de compilación genérico de LIG en L–LIA

[INIT]	$\$_0[\circ\circ] \longmapsto \$_0[\circ\circ]\ \nabla_{0,0}[\,]$	
[CALL]	$\nabla_{r,s}[\circ\circ] \longmapsto \nabla_{r,s}[\circ\circ]\ \Box[\,]$	$A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\,]\Upsilon_2$
[SCALL]	$\nabla_{r,s}[\circ\circ\gamma] \longmapsto \nabla_{r,s}[\,]\ \Box[\circ\circ\gamma']$	$A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\circ\circ\gamma']\Upsilon_2$
[SEL]	$\Box[\circ\circ] \longmapsto \nabla_{r,0}[\circ\circ]$	$r \neq 0$
[PUB]	$\nabla_{r,n_r}[\circ\circ] \longmapsto A_{r,0}[\circ\circ]$	
[RET]	$\nabla_{r,s}[\circ\circ]\ A_{r,s+1}[\,] \longmapsto \nabla_{r,s+1}[\circ\circ]$	$A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\,]\Upsilon_2$
[SRET]	$\nabla_{r,s}[\circ\circ]\ A_{r,s+1}[\,] \longmapsto \nabla_{r,s+1}[\circ\circ]$	$A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\circ\circ\gamma']\Upsilon_2$
[SCAN]	$\Box[\,] \overset{a}{\longmapsto} A_{r,0}[\,]$	$A_{r,0}[\,] \to a$

Tabla 9.5: Reglas del esquema de compilación ascendente-descendente de LIG en L–LIA

[INIT]	$\$_0[\circ\circ] \longmapsto \$_0[\circ\circ]\ \nabla_{0,0}[\,]$	
[CALL]	$\nabla_{r,s}[\circ\circ] \longmapsto \nabla_{r,s}[\circ\circ]\ \overline{A_{r,s+1}}[\,]$	$A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\,]\Upsilon_2$
[SCALL]	$\nabla_{r,s}[\circ\circ\gamma] \longmapsto \nabla_{r,s}[\,]\ \overline{A_{r,s+1}}[\circ\circ\gamma']$	$A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\circ\circ\gamma']\Upsilon_2$
[SEL]	$\overline{A_{r,0}}[\circ\circ] \longmapsto \nabla_{r,0}[\circ\circ]$	$r \neq 0$
[PUB]	$\nabla_{r,n_r}[\circ\circ] \longmapsto \overline{\overline{A_{r,0}}}[\circ\circ]$	
[RET]	$\nabla_{r,s}[\circ\circ]\ \overline{\overline{A_{r,s+1}}}[\,] \longmapsto \nabla_{r,s+1}[\circ\circ]$	$A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\,]\Upsilon_2$
[SRET]	$\nabla_{r,s}[\circ\circ]\ \overline{\overline{A_{r,s+1}}}[\,] \longmapsto \nabla_{r,s+1}[\circ\circ]$	$A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\circ\circ\gamma']\Upsilon_2$
[SCAN]	$\overline{A_{r,0}}[\,] \overset{a}{\longmapsto} \overline{\overline{A_{r,0}}}[\,]$	$A_{r,0}[\,] \to a$

Tabla 9.6: Reglas del esquema de compilación Earley-descendente de LIG en L–LIA

[INIT]	$\$_0[\circ\circ] \longmapsto \$_0[\circ\circ]\ \nabla_{0,0}[\,]$	
[CALL]	$\nabla_{r,s}[\circ\circ] \longmapsto \nabla_{r,s}[\circ\circ]\ \ A_{r,s+1}[\,]$	$A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\,]\Upsilon_2$
[SCALL]	$\nabla_{r,s}[\circ\circ\gamma] \longmapsto \nabla_{r,s}[\,]\ \ A_{r,s+1}[\circ\circ\gamma']$	$A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\circ\circ\gamma']\Upsilon_2$
[SEL]	$A_{r,0}[\circ\circ] \longmapsto \nabla_{r,0}[\circ\circ]$	$r \neq 0$
[PUB]	$\nabla_{r,n_r}[\circ\circ] \longmapsto \Box[\circ\circ]$	
[RET]	$\nabla_{r,s}[\circ\circ]\ \ \Box[\,] \longmapsto \nabla_{r,s+1}[\circ\circ]$	$A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\,]\Upsilon_2$
[SRET]	$\nabla_{r,s}[\circ\circ]\ \ \Box[\,] \longmapsto \nabla_{r,s+1}[\circ\circ]$	$A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\circ\circ\gamma']\Upsilon_2$
[SCAN]	$A_{r,0}[\,] \overset{a}{\longmapsto}\ \Box[\,]$	$A_{r,0}[\,] \to a$

Tabla 9.7: Reglas del esquema de compilación descendente-descendente de LIG en L–LIA

definido por el conjunto de reglas mostrado en la tabla 9.4 y por los elementos inicial $\$_0[\,]$ y final $\Box[\,]$. §

9.3.2 Esquemas de compilación de gramáticas de adjunción de árboles

Mediante los tipos de transiciones presentes en los L–LIA podemos implementar estrategias de análisis de gramáticas de adjunción de árboles en las cuales las adjunciones se resuelven en la fase descendente. Para ello, cuando se visita la raíz de un árbol auxiliar β se almacenará en la pila de índices asociada el nodo en el cual se realizó la adjunción de β. Esa pila de índices es transportada a lo largo de la espina hasta alcanzar el nodo pie, momento en el cual su cima nos indicará el nodo en el cual debe continuar el análisis. En cambio, durante la fase ascendente la pila de índices asociada a los nodos visitados estará vacía.

En lo que respecta al recorrido de cada uno de los árboles elementales no existe limitación alguna, lo cual nos permite definir un esquema de compilación genérico parametrizado precisamente en función de la información predicha y propagada con respecto a los nodos de los árboles elementales durante el recorrido de los mismos. Dicho esquema genérico será luego convertido en los esquemas específicos correspondientes a las estrategias ascendente-descendente, Earley-descendente y descendente-descendente.

Esquema de compilación 9.7 El esquema de compilación genérico de una gramática de adjunción de árboles en un autómata lineal de índices orientado a la izquierda queda definido por el conjunto de reglas mostrado en la tabla 9.8 y los elementos inicial $\$_0[\,]$ y final $\overleftarrow{\top\alpha}[\,]$, con $\alpha \in \boldsymbol{I}$. §

Estrategia ascendente-descendente

Esquema de compilación 9.8 El esquema de compilación ascendente-descendente de una gramática de adjunción de árboles en un autómata lineal de índices orientado a la izquierda queda definido por el conjunto de reglas mostrado en la tabla 9.9 y los elementos inicial $\$_0[\,]$ y final $\top\alpha[\,]$, con $\alpha \in \boldsymbol{I}$. §

[INIT]	$\$_0[\circ\circ] \longmapsto \$_0[\circ\circ]\ \nabla^{\alpha}_{0,0}[\,]$	$\alpha \in \boldsymbol{I}$
[CALL]	$\nabla^{\gamma}_{r,s}[\circ\circ] \longmapsto \nabla^{\gamma}_{r,s}[\circ\circ]\ \overrightarrow{N^{\gamma}_{r,s+1}}[\,]$	$N^{\gamma}_{r,s+1} \not\in \text{espina}(\gamma),\ \textbf{nil} \in \text{adj}(N_{r,s+1})$
[SCALL]	$\nabla^{\gamma}_{r,s}[\circ\circ] \longmapsto \nabla^{\gamma}_{r,s}[\,]\ \overrightarrow{N^{\gamma}_{r,s+1}}[\circ\circ]$	$N^{\gamma}_{r,s+1} \in \text{espina}(\gamma),\ \textbf{nil} \in \text{adj}(N_{r,s+1})$
[SEL]	$\overrightarrow{N^{\gamma}_{r,0}}[\circ\circ] \longmapsto \nabla^{\gamma}_{r,0}[\circ\circ]$	$r \neq 0$
[PUB]	$\nabla^{\gamma}_{r,n_r}[\circ\circ] \longmapsto \overleftarrow{N^{\gamma}_{r,0}}[\circ\circ]$	
[RET]	$\nabla^{\gamma}_{r,s}[\circ\circ]\ \overleftarrow{N^{\gamma}_{r,s+1}}[\,] \longmapsto \nabla^{\gamma}_{r,s+1}[\circ\circ]$	$N^{\gamma}_{r,s+1} \not\in \text{espina}(\gamma),\ \textbf{nil} \in \text{adj}(N_{r,s+1})$
[SRET]	$\nabla^{\gamma}_{r,s}[\circ\circ]\ \overleftarrow{N^{\gamma}_{r,s+1}}[\,] \longmapsto \nabla^{\gamma}_{r,s+1}[\circ\circ]$	$N^{\gamma}_{r,s+1} \in \text{espina}(\gamma),\ \textbf{nil} \in \text{adj}(N_{r,s+1})$
[SCAN]	$\overrightarrow{N^{\gamma}_{r,0}}[\circ\circ] \overset{a}{\longmapsto} \overleftarrow{N^{\gamma}_{r,0}}[\circ\circ]$	$N^{\gamma}_{r,0}[\,] \to a$
[ACALL]	$\nabla^{\gamma}_{r,s}[\circ\circ] \longmapsto \nabla^{\gamma}_{r,s}[\,]\ \overrightarrow{\top^{\beta}}[\circ\circ N^{\gamma}_{r,s+1}]$	$\beta \in \text{adj}(N^{\gamma}_{r,s+1})$
[ARET]	$\nabla^{\gamma}_{r,s}[\circ\circ]\ \overleftarrow{\top^{\beta}}[\,] \longmapsto \nabla^{\gamma}_{r,s+1}[\circ\circ]$	$\beta \in \text{adj}(N^{\gamma}_{r,s+1})$
[FCALL]	$\nabla^{\beta}_{f,0}[\circ\circ N^{\gamma}_{r,s+1}] \longmapsto \nabla^{\beta}_{f,0}[\,]\ \overrightarrow{N^{\gamma}_{r,s+1}}[\circ\circ]$	$N^{\beta}_{f,0} = \mathbf{F}^{\beta},\ \beta \in \text{adj}(N^{\gamma}_{r,s+1})$
[FRET]	$\nabla^{\beta}_{f,0}[\circ\circ]\ \overleftarrow{N^{\gamma}_{r,s+1}}[\,] \longmapsto \nabla^{\beta}_{f,1}[\circ\circ]$	$N^{\beta}_{f,0} = \mathbf{F}^{\beta},\ \beta \in \text{adj}(N^{\gamma}_{r,s+1})$

Tabla 9.8: Reglas del esquema de compilación genérico de TAG en L–LIA

Estrategia Earley-descendente

Esquema de compilación 9.9 El esquema de compilación Earley-descendente de una gramática de adjunción de árboles en un autómata lineal de índices orientado a la izquierda queda definido por el conjunto de reglas mostrado en la tabla 9.10 y los elementos inicial $\$_0[\,]$ y final $\overline{\overline{\top\alpha}}[\,]$, con $\alpha \in \boldsymbol{I}$. §

Estrategia descendente-descendente

Esquema de compilación 9.10 El esquema de compilación descendente-descendente de una gramática de adjunción de árboles en un autómata lineal de índices orientado a la izquierda queda definido por el conjunto de reglas mostrado en la tabla 9.11 y los elementos inicial $\$_0[\,]$ y final $\Box[\,]$. §

9.3.3 L–LIA y los lenguajes de adjunción de árboles

En la tabla 9.12 se muestra los tipos de transiciones y su relación con las reglas de los esquemas de compilación de LIG y TAG.

Teorema 9.1 *Los autómatas lineales de índices orientados a la izquierda que utilizan el juego de transiciones de la tabla 9.12 aceptan la clase de los lenguajes de adjunción de árboles.*

[INIT]	$\$_0[\circ\circ] \longmapsto \$_0[\circ\circ]\ \nabla^{\alpha}_{0,0}[\,]$	$\alpha \in \boldsymbol{I}$
[CALL]	$\nabla^{\gamma}_{r,s}[\circ\circ] \longmapsto \nabla^{\gamma}_{r,s}[\circ\circ]\ \Box[\,]$	$N^{\gamma}_{r,s+1} \notin \text{espina}(\gamma),\ \textbf{nil} \in \text{adj}(N_{r,s+1})$
[SCALL]	$\nabla^{\gamma}_{r,s}[\circ\circ] \longmapsto \nabla^{\gamma}_{r,s}[\,]\ \Box[\circ\circ]$	$N^{\gamma}_{r,s+1} \in \text{espina}(\gamma),\ \textbf{nil} \in \text{adj}(N_{r,s+1})$
[SEL]	$\Box[\circ\circ] \longmapsto \nabla^{\gamma}_{r,0}[\circ\circ]$	$r \neq 0$
[PUB]	$\nabla^{\gamma}_{r,n_r}[\circ\circ] \longmapsto N^{\gamma}_{r,0}[\circ\circ]$	
[RET]	$\nabla^{\gamma}_{r,s}[\circ\circ]\ N^{\gamma}_{r,s+1}[\,] \longmapsto \nabla^{\gamma}_{r,s+1}[\circ\circ]$	$N^{\gamma}_{r,s+1} \notin \text{espina}(\gamma),\ \textbf{nil} \in \text{adj}(N_{r,s+1})$
[SRET]	$\nabla^{\gamma}_{r,s}[\circ\circ]\ N^{\gamma}_{r,s+1}[\,] \longmapsto \nabla^{\gamma}_{r,s+1}[\circ\circ]$	$N^{\gamma}_{r,s+1} \in \text{espina}(\gamma),\ \textbf{nil} \in \text{adj}(N_{r,s+1})$
[SCAN]	$\Box[\circ\circ] \overset{a}{\longmapsto} N^{\gamma}_{r,0}[\circ\circ]$	$N^{\gamma}_{r,0}[\,] \to a$
[ACALL]	$\nabla^{\gamma}_{r,s}[\circ\circ] \longmapsto \nabla^{\gamma}_{r,s}[\,]\ \Box[\circ\circ N^{\gamma}_{r,s+1}]$	$\beta \in \text{adj}(N^{\gamma}_{r,s+1})$
[ARET]	$\nabla^{\gamma}_{r,s}[\circ\circ]\ \top^{\beta}[\,] \longmapsto \nabla^{\gamma}_{r,s+1}[\circ\circ]$	$\beta \in \text{adj}(N^{\gamma}_{r,s+1})$
[FCALL]	$\nabla^{\beta}_{f,0}[\circ\circ N^{\gamma}_{r,s+1}] \longmapsto \nabla^{\beta}_{f,0}[\,]\ \Box[\circ\circ]$	$N^{\beta}_{f,0} = \mathbf{F}^{\beta},\ \beta \in \text{adj}(N^{\gamma}_{r,s+1})$
[FRET]	$\nabla^{\beta}_{f,0}[\circ\circ]\ N^{\gamma}_{r,s+1}[\,] \longmapsto \nabla^{\beta}_{f,1}[\circ\circ]$	$N^{\beta}_{f,0} = \mathbf{F}^{\beta},\ \beta \in \text{adj}(N^{\gamma}_{r,s+1})$

Tabla 9.9: Reglas del esquema de compilación ascendente-descendente de TAG en L–LIA

Demostración:

Por los esquemas de compilación de LIG y TAG en L–LIA sabemos que los lenguajes de adjunción de árboles son aceptados por los L–LIA que utilizan las transiciones de la tabla 9.12.

Para demostrar que todo lenguaje aceptado por un L–LIA que utilice las transiciones de la tabla 9.12 es un lenguaje de adjunción de árboles, definiremos un procedimiento para crear una gramática lineal de índices a partir de tales autómatas.

Sea $\mathcal{A} = (V_T, V_S, \$_0, \$_f, V_I, \mathcal{T})$ un autómata lineal de índices orientado a la izquierda. Construiremos una gramática lineal de índices $\mathcal{L} = (V_T, V_N, V_I, S, P)$, donde el conjunto V_N de no-terminales estará formado por pares $\langle E, B\rangle$ tal que $A, B \in V_S$. Para que $\mathcal{L}$ reconozca el lenguaje aceptado por $\mathcal{A}$ el conjunto de producciones en P ha de construirse a partir de las transiciones en $\mathcal{T}$ de la siguiente manera:

- Para toda transición $C[\circ\circ] \overset{a}{\longmapsto} F[\circ\circ]$ y para todo $E \in V_S$ creamos una producción

$$\langle C, E\rangle[\circ\circ] \to a\ \langle F, E\rangle[\circ\circ]$$

- Para todo par de transiciones $C[\circ\circ]\ F[\,] \longmapsto G[\circ\circ]$ y $C[\circ\circ] \longmapsto C[\circ\circ]\ F'[\,]$, y para todo $E \in V_S$ creamos una producción

$$\langle C, E\rangle[\circ\circ] \to \langle F', F\rangle[\,]\ \ \langle G, E\rangle[\circ\circ]$$

- Para todo par de transiciones $C[\circ\circ]\ F[\,] \longmapsto G[\circ\circ]$ y $C[\circ\circ\gamma] \longmapsto C[\,]\ F'[\circ\circ\gamma']$, y para todo $E \in V_S$ creamos una producción

$$\langle C, E\rangle[\circ\circ\gamma] \to \langle F', F\rangle[\circ\circ\gamma']\ \ \langle G, E\rangle[\,]$$

- Para todo $E \in V_S$ creamos una producción

$$\langle E, E\rangle[\,] \to \epsilon$$

[INIT]	$\$_0[\circ\circ] \longmapsto \$_0[\circ\circ]\ \nabla^{\alpha}_{0,0}[\,]$	$\alpha \in \boldsymbol{I}$
[CALL]	$\nabla^{\gamma}_{r,s}[\circ\circ] \longmapsto \nabla^{\gamma}_{r,s}[\circ\circ]\ \overline{N^{\gamma}_{r,s+1}}[\,]$	$N^{\gamma}_{r,s+1} \notin \text{espina}(\gamma),\ \textbf{nil} \in \text{adj}(N_{r,s+1})$
[SCALL]	$\nabla^{\gamma}_{r,s}[\circ\circ] \longmapsto \nabla^{\gamma}_{r,s}[\,]\ \overline{N^{\gamma}_{r,s+1}}[\circ\circ]$	$N^{\gamma}_{r,s+1} \in \text{espina}(\gamma),\ \textbf{nil} \in \text{adj}(N_{r,s+1})$
[SEL]	$\overline{N^{\gamma}_{r,0}}[\circ\circ] \longmapsto \nabla^{\gamma}_{r,0}[\circ\circ]$	$r \neq 0$
[PUB]	$\nabla^{\gamma}_{r,n_r}[\circ\circ] \longmapsto \overline{\overline{\overline{N^{\gamma}_{r,0}}}}[\circ\circ]$	
[RET]	$\nabla^{\gamma}_{r,s}[\circ\circ]\ \overline{\overline{\overline{N^{\gamma}_{r,s+1}}}}[\,] \longmapsto \nabla^{\gamma}_{r,s+1}[\circ\circ]$	$N^{\gamma}_{r,s+1} \notin \text{espina}(\gamma),\ \textbf{nil} \in \text{adj}(N_{r,s+1})$
[SRET]	$\nabla^{\gamma}_{r,s}[\circ\circ]\ \overline{\overline{\overline{N^{\gamma}_{r,s+1}}}}[\,] \longmapsto \nabla^{\gamma}_{r,s+1}[\circ\circ]$	$N^{\gamma}_{r,s+1} \in \text{espina}(\gamma),\ \textbf{nil} \in \text{adj}(N_{r,s+1})$
[SCAN]	$\overline{N^{\gamma}_{r,0}}[\circ\circ] \overset{a}{\longmapsto} \overline{\overline{\overline{N^{\gamma}_{r,0}}}}[\circ\circ]$	$N^{\gamma}_{r,0}[\,] \to a$
[ACALL]	$\nabla^{\gamma}_{r,s}[\circ\circ] \longmapsto \nabla^{\gamma}_{r,s}[\,]\ \overline{\top^{\beta}}[\circ\circ N^{\gamma}_{r,s+1}]$	$\beta \in \text{adj}(N^{\gamma}_{r,s+1})$
[ARET]	$\nabla^{\gamma}_{r,s}[\circ\circ]\ \overline{\overline{\overline{\top^{\beta}}}}[\,] \longmapsto \nabla^{\gamma}_{r,s+1}[\circ\circ]$	$\beta \in \text{adj}(N^{\gamma}_{r,s+1})$
[FCALL]	$\nabla^{\beta}_{f,0}[\circ\circ N^{\gamma}_{r,s+1}] \longmapsto \nabla^{\beta}_{f,0}[\,]\ \overline{N^{\gamma}_{r,s+1}}[\circ\circ]$	$N^{\beta}_{f,0} = \mathbf{F}^{\beta},\ \beta \in \text{adj}(N^{\gamma}_{r,s+1})$
[FRET]	$\nabla^{\beta}_{f,0}[\circ\circ]\ \overline{\overline{\overline{N^{\gamma}_{r,s+1}}}}[\,] \longmapsto \nabla^{\beta}_{f,1}[\circ\circ]$	$N^{\beta}_{f,0} = \mathbf{F}^{\beta},\ \beta \in \text{adj}(N^{\gamma}_{r,s+1})$

Tabla 9.10: Reglas del esquema de compilación Earley-descendente de TAG en L–LIA

[INIT]	$\$_0[\circ\circ] \longmapsto \$_0[\circ\circ]\ \nabla^{\alpha}_{0,0}[\,]$	$\alpha \in \boldsymbol{I}$
[CALL]	$\nabla^{\gamma}_{r,s}[\circ\circ] \longmapsto \nabla^{\gamma}_{r,s}[\circ\circ]\ N^{\gamma}_{r,s+1}[\,]$	$N^{\gamma}_{r,s+1} \notin \text{espina}(\gamma),\ \textbf{nil} \in \text{adj}(N_{r,s+1})$
[SCALL]	$\nabla^{\gamma}_{r,s}[\circ\circ] \longmapsto \nabla^{\gamma}_{r,s}[\,]\ N^{\gamma}_{r,s+1}[\circ\circ]$	$N^{\gamma}_{r,s+1} \in \text{espina}(\gamma),\ \textbf{nil} \in \text{adj}(N_{r,s+1})$
[SEL]	$N^{\gamma}_{r,0}[\circ\circ] \longmapsto \nabla^{\gamma}_{r,0}[\circ\circ]$	$r \neq 0$
[PUB]	$\nabla^{\gamma}_{r,n_r}[\circ\circ] \longmapsto \Box[\circ\circ]$	
[RET]	$\nabla^{\gamma}_{r,s}[\circ\circ]\ \Box[\,] \longmapsto \nabla^{\gamma}_{r,s+1}[\circ\circ]$	$N^{\gamma}_{r,s+1} \notin \text{espina}(\gamma),\ \textbf{nil} \in \text{adj}(N_{r,s+1})$
[SRET]	$\nabla^{\gamma}_{r,s}[\circ\circ]\ \Box[\,] \longmapsto \nabla^{\gamma}_{r,s+1}[\circ\circ]$	$N^{\gamma}_{r,s+1} \in \text{espina}(\gamma),\ \textbf{nil} \in \text{adj}(N_{r,s+1})$
[SCAN]	$N^{\gamma}_{r,0}[\circ\circ] \overset{a}{\longmapsto} \Box[\circ\circ]$	$N^{\gamma}_{r,0}[\,] \to a$
[ACALL]	$\nabla^{\gamma}_{r,s}[\circ\circ] \longmapsto \nabla^{\gamma}_{r,s}[\,]\ \top^{\beta}[\circ\circ N^{\gamma}_{r,s+1}]$	$\beta \in \text{adj}(N^{\gamma}_{r,s+1})$
[ARET]	$\nabla^{\gamma}_{r,s}[\circ\circ]\ \Box[\,] \longmapsto \nabla^{\gamma}_{r,s+1}[\circ\circ]$	$\beta \in \text{adj}(N^{\gamma}_{r,s+1})$
[FCALL]	$\nabla^{\beta}_{f,0}[\circ\circ N^{\gamma}_{r,s+1}] \longmapsto \nabla^{\beta}_{f,0}[\,]\ N^{\gamma}_{r,s+1}[\circ\circ]$	$N^{\beta}_{f,0} = \mathbf{F}^{\beta},\ \beta \in \text{adj}(N^{\gamma}_{r,s+1})$
[FRET]	$\nabla^{\beta}_{f,0}[\circ\circ]\ \Box[\,] \longmapsto \nabla^{\beta}_{f,1}[\circ\circ]$	$N^{\beta}_{f,0} = \mathbf{F}^{\beta},\ \beta \in \text{adj}(N^{\gamma}_{r,s+1})$

Tabla 9.11: Reglas del esquema de compilación descendente-descendente de TAG en L–LIA

Transición	Compilación de LIG	Compilación de TAG
$C[\circ\circ] \overset{a}{\longmapsto} F[\circ\circ]$	**[SEL][PUB][SCAN]**	**[SEL][PUB][SCAN]**
$C[\circ\circ] \longmapsto C[\circ\circ]\ F[\,]$	**[INIT][CALL]**	**[INIT][CALL]**
$C[\circ\circ\gamma] \longmapsto C[\,]\ F[\circ\circ\gamma']$	**[SCALL]**	**[SCALL][ACALL][FCALL]**
$C[\circ\circ]\ F[\,] \longmapsto G[\circ\circ]$	**[RET][SRET]**	**[RET][SRET][ARET][FRET]**

Tabla 9.12: Tipos de transiciones L–LIA

- Para toda transición $\$_0[\circ\circ] \longmapsto \$_0[\circ\circ]\ F[\ ]$ o $\$_0[\circ\circ] \longmapsto \$_0[\circ\circ]\ F[\circ\circ]$, donde $F \in V_S - \{\$_0\}$, creamos una producción

$$\langle \$_0, \$_0 \rangle[\circ\circ] \to \langle F, \$_f \rangle[\circ\circ]$$

- Para toda transición $C[\circ\circ] \overset{a}{\longmapsto} \$_f[\circ\circ]$ creamos una transición

$$\langle C, \$_f \rangle[\,] \to a$$

Con respecto al axioma de la gramática, tenemos que $S = \langle \$_0, \$_0 \rangle$.

Mediante inducción en la longitud de las derivaciones, es posible mostrar que $\langle C, E\rangle[\alpha] \overset{*}{\Rightarrow} w$ si y sólo si $(C[\alpha], w) \overset{*}{\vdash} (E[\,], \epsilon)$, puesto que

- Si una derivación $(C[\alpha], w) \overset{*}{\vdash} (E[\,], \epsilon)$ es el resultado de aplicar la secuencia $t_1, \ldots, t_m$ de transiciones en $\mathcal{T}$, entonces existe una secuencia $p_1, \ldots, p_m$ de producciones en P tal que la derivación $\langle C, E\rangle[\alpha] \overset{*}{\Rightarrow} w$ resultado de aplicar $p_1, \ldots, p_m$ reconoce w. La demostración se realiza por inducción en la longitud de la derivación del autómata.

  El caso base lo constituye la derivación $(E[\ ], \epsilon) \overset{0}{\vdash} (E[\ ], \epsilon)$, para la que existe una producción $\langle E, E\rangle[\,] \to \epsilon$. Por hipótesis de inducción suponemos que la proposición se cumple para cualquier derivación del autómata de longitud m. En tal caso, durante el paso de inducción verificamos que se cumple para cualquier posible derivación de longitud mayor que m:

  - Si $(C[\alpha], aw) \vdash (F[\alpha], w) \overset{m}{\vdash} (E[\,], \epsilon)$, $\exists \langle C, E\rangle[\circ\circ] \to a\ \langle F, E\rangle[\circ\circ] \in P$, por hipótesis de inducción $\langle F, E\rangle[\alpha] \overset{*}{\Rightarrow} w$ y en consecuencia $\langle C, E\rangle[\alpha] \overset{*}{\Rightarrow} aw$.
  - Si $(C[\alpha], w_1w_2) \vdash (C[\alpha]\ F'[\,], w_1w_2) \overset{m_1}{\vdash} (C[\alpha]\ F[\,], w_2) \vdash (G[\alpha], w_2) \overset{m_2}{\vdash} (E[\,], \epsilon)$, existe una producción $\langle C, E\rangle[\circ\circ] \to \langle F', F\rangle[\,]\ \langle G, E\rangle[\circ\circ]$, por hipótesis de inducción $\langle F', F\rangle[\,] \overset{*}{\Rightarrow} w_1$ y $\langle G, E\rangle[\alpha] \overset{*}{\Rightarrow} w_2$ y en consecuencia $\langle B, E\rangle[\alpha] \overset{*}{\Rightarrow} w_1w_2$.
  - Si $(C[\alpha\gamma], w_1w_2) \vdash (C[\,]\ F'[\alpha\gamma'], w_1w_2) \overset{m_1}{\vdash} (C[\,]\ F[\,], w_2) \vdash (G[\,], w_2) \overset{m_2}{\vdash} (E[\,], \epsilon)$, existe una producción $\langle C, E\rangle[\circ\circ\gamma] \to \langle F', F\rangle[\circ\circ\gamma']\ \langle G, E\rangle[\,]$, por hipótesis de inducción $\langle F', F\rangle[\alpha\gamma'] \overset{*}{\Rightarrow} w_1$ y $\langle G, E\rangle[\,] \overset{*}{\Rightarrow} w_2$ y en consecuencia $\langle C, E\rangle[\alpha\gamma] \overset{*}{\Rightarrow} w_1w_2$.

- Si una derivación izquierda $\langle C, E\rangle[\alpha] \overset{*}{\Rightarrow} w$ reconoce la cadena w como resultado de aplicar la secuencia $p_1, \ldots, p_m$ de producciones en P, entonces existe una secuencia de transiciones $t_1, \ldots, t_m$ tal que la derivación $(C[\alpha], w) \overset{*}{\vdash} (E[\,], \epsilon)$ es el resultado de aplicar la secuencia de transiciones $t_1, \ldots, t_m$. La demostración se realiza por inducción en la longitud de la derivación de la gramática. El caso base lo constituye la derivación $\langle E, E\rangle[\ ] \Rightarrow \epsilon$, para la que existe una derivación $(E[\ ], \epsilon) \overset{0}{\vdash} (E[\ ], \epsilon)$ en el autómata. Por hipótesis de inducción suponemos que la proposición se cumple para cualquier derivación de la gramática de longitud m. En tal caso, durante el paso de inducción verificamos que se cumple para cualquier posible derivación de longitud mayor que m:

- Si $\langle C,E\rangle[\alpha] \Rightarrow a\ \langle F,E\rangle[\alpha] \overset{m}{\Rightarrow} aw$, existe una transición $C[\circ\circ] \overset{a}{\longmapsto} F[\circ\circ]$, por hipótesis de inducción $(F[\alpha],w) \overset{*}{\vdash} (E[\,],\epsilon)$ y en consecuencia $(C[\alpha],aw) \overset{*}{\vdash} (E[\,],\epsilon)$.
- Si $\langle C,E\rangle[\alpha] \Rightarrow \langle F',F\rangle[\ ]\ \langle G,E\rangle[\alpha] \overset{m_1}{\Rightarrow} w_1\ \langle G,E\rangle[\alpha] \overset{m_2}{\Rightarrow} w_1 w_2$, existe un par de transiciones $C[\circ\circ] \longmapsto C[\circ\circ]\ F'[\ ]$ y $C[\circ\circ]\ F[\ ] \longmapsto G[\circ\circ]$, por hipótesis de inducción $(F'[\,],w_1) \overset{*}{\vdash} (F[\,],\epsilon)$ y $(G[\alpha],w_2) \overset{*}{\vdash} (E[\ ],\epsilon)$ y en consecuencia $(C[\alpha],w_1w_2) \overset{*}{\vdash} (E[\,],\epsilon)$.
- Si $\langle C,E\rangle[\alpha\gamma] \Rightarrow \langle F',F\rangle[\alpha\gamma']\ \langle G,E\rangle[\ ] \overset{m_1}{\Rightarrow} w_1\ \langle G,E\rangle[\ ] \overset{m_2}{\Rightarrow} w_1 w_2$, existe un par de transiciones $C[\circ\circ\gamma] \longmapsto C[\ ]\ F'[\circ\circ\gamma']$ y $C[\circ\circ]\ F[\ ] \longmapsto G[\circ\circ]$, por hipótesis de inducción $(F'[\alpha\gamma'],w_1) \overset{*}{\vdash} (F[\,],\epsilon)$ y $(G[\ ],w_2) \overset{*}{\vdash} (E[\ ],\epsilon)$ y en consecuencia $(C[\alpha\gamma],w_1w_2) \overset{*}{\vdash} (E[\,],\epsilon)$.

□

9.3.4 Tabulación

Presentamos en esta sección una técnica de tabulación alternativa a la propuesta por Nederhof en [126] para los autómatas lineales de índices orientados a la izquierda. Una diferencia fundamental es que el conjunto de transiciones que se considera es distinto, como se ha comentado anteriormente. Para proceder al diseño de la técnica de tabulación debemos primero definir los distintos tipos de derivaciones observables en un L–LIA:

Derivaciones de llamada. Corresponden a la transmisión de una pila de índices en la fase de llamada de la estrategia de análisis y son de la forma

$$\begin{aligned}(\Upsilon\ A[\delta], a_{h+1}\dots a_n) &\overset{*}{\vdash} (\Upsilon\ A[\,]\ \Upsilon_1\ B[\delta\gamma], a_{i+1}\dots a_n)\\ &\overset{*}{\vdash} (\Upsilon\ A[\,]\ \Upsilon_1\ C[\delta\gamma], a_{j+1}\dots a_n)\end{aligned}$$

donde $\gamma \in V_I$, $\delta \in (V_S[V_I^*])^*$, tanto $B[\alpha\gamma]$ como $C[\alpha\gamma]$ son descendientes dependientes de $A[\alpha]$ y no existe un par $(F[\delta\gamma],f) \neq (B[\delta\gamma],j)$ tal que

$$\begin{aligned}(\Upsilon\ A[\delta], a_{h+1}\dots a_n) &\overset{*}{\vdash} (\Upsilon\ A[\,]\ \Upsilon_1\ F[\delta\gamma], a_{f+1}\dots a_n)\\ &\overset{*}{\vdash} (\Upsilon\ A[\,]\ \Upsilon_1\ B[\delta\gamma], a_{i+1}\dots a_n)\\ &\overset{*}{\vdash} (\Upsilon\ A[\,]\ \Upsilon_1\ C[\delta\gamma], a_{j+1}\dots a_n)\end{aligned}$$

La figura 9.3 muestra una representación gráfica de las derivaciones de llamada.

Para cualquier $\Upsilon' \in (V_S[V_I^*])^*$ y $\alpha \in V_I*$ se cumple que:

$$\begin{aligned}(\Upsilon'\ A[\alpha], a_{h+1}\dots a_n) &\overset{*}{\vdash} (\Upsilon'\ A[\,]\ \Upsilon_1\ B[\alpha\gamma], a_{i+1}\dots a_n)\\ &\overset{*}{\vdash} (\Upsilon'\ A[\,]\ \Upsilon_1\ C[\alpha\gamma], a_{j+1}\dots a_n)\end{aligned}$$

En consecuencia, este tipo de derivaciones se pueden representar mediante ítems de la forma

$$[A, h \mid B, i, \gamma, C, j, \gamma \mid -,-,-,-]$$

Derivaciones de retorno. Corresponden a la propagación de una pila durante la fase de retorno de la estrategia de análisis y son de la forma

$$\begin{array}{ll}(\Upsilon\ A[\delta], a_{h+1}\dots a_n) & \overset{*}{\vdash} (\Upsilon\ A[\,]\ \Upsilon_1\ B[\delta\gamma], a_{i+1}\dots a_n) \\ & \overset{*}{\vdash} (\Upsilon\ A[\,]\ \Upsilon_1\ B[\,]\ \Upsilon_2\ D[\delta], a_{p+1}\dots a_n) \\ & \overset{*}{\vdash} (\Upsilon\ A[\,]\ \Upsilon_1\ B[\,]\ \Upsilon_2\ E[\,], a_{q+1}\dots a_n) \\ & \overset{*}{\vdash} (\Upsilon\ A[\,]\ \Upsilon_1\ C[\,], a_{j+1}\dots a_n)\end{array}$$

donde $\gamma \in V_I$, $\delta \in V_I^*$, tanto $B[\alpha\gamma]$ como $D[\alpha]$ son descendientes dependientes de $A[\alpha]$ y no existen $(F[\delta\gamma], f) \neq (B[\delta\gamma], i)$ ni $(G[\delta], g) \neq (D[\delta], p)$ tal que

$$\begin{array}{ll}(\Upsilon\ A[\delta], a_{h+1}\dots a_n) & \overset{*}{\vdash} (\Upsilon\ A[\,]\ \Upsilon_1\ F[\delta\gamma], a_{f+1}\dots a_n) \\ & \overset{*}{\vdash} (\Upsilon\ A[\,]\ \Upsilon_1\ B[\delta\gamma], a_{i+1}\dots a_n) \\ & \overset{*}{\vdash} (\Upsilon\ A[\,]\ \Upsilon_1\ B[\,]\ \Upsilon_2\ G[\delta], a_{g+1}\dots a_n) \\ & \overset{*}{\vdash} (\Upsilon\ A[\,]\ \Upsilon_1\ B[\,]\ \Upsilon_2\ D[\delta], a_{p+1}\dots a_n) \\ & \overset{*}{\vdash} (\Upsilon\ A[\,]\ \Upsilon_1\ B[\,]\ \Upsilon_2\ E[\,], a_{q+1}\dots a_n) \\ & \overset{*}{\vdash} (\Upsilon\ A[\,]\ \Upsilon_1\ C[\,], a_{j+1}\dots a_n)\end{array}$$

La figura 9.4 muestra una representación gráfica de las derivaciones de retorno.

Para cualquier $\Upsilon' \in (V_S[V_I^*])^*$ y pila de índices $\delta' \in V_I^*$ tal que existe una derivación $(D[\delta'], a_{p+1}\dots a_n) \overset{*}{\vdash} (E[\,], a_{q+1}\dots a_n)$ se cumple que

$$\begin{array}{ll}(\Upsilon'\ A[\delta'], a_{h+1}\dots a_n) & \overset{*}{\vdash} (\Upsilon'\ A[\,]\ \Upsilon_1\ B[\delta'\gamma], a_{i+1}\dots a_n) \\ & \overset{*}{\vdash} (\Upsilon'\ A[\,]\ \Upsilon_1\ B[\,]\ \Upsilon_2\ D[\delta'], a_{p+1}\dots a_n) \\ & \overset{*}{\vdash} (\Upsilon'\ A[\,]\ \Upsilon_1\ B[\,]\ \Upsilon_2\ E[\,], a_{q+1}\dots a_n) \\ & \overset{*}{\vdash} (\Upsilon'\ A[\,]\ \Upsilon_1\ C[\,], a_{j+1}\dots a_n)\end{array}$$

Ello permite representar este tipo de derivaciones mediante ítems de la forma

$$[A, h \mid B, i, \gamma, C, j, - \mid D, p, E, q]$$

donde los componentes (A, h) y (D, p, E, q) permiten asegurar que estamos trabajando con la misma pila de índices δ a lo largo de toda la derivación.

Derivaciones de puntos especiales. Son aquellas de la forma

$$(\Upsilon\ B[\,], a_{i+1}\dots a_n) \overset{*}{\vdash} (\Upsilon\ C[\,], a_{j+1}\dots a_n)$$

y no existe $(F[\,], f) \neq (B[\,], i)$ tal que

$$\begin{array}{ll}(\Upsilon\ F[\,], a_{f+1}\dots a_n) & \overset{*}{\vdash} (\Upsilon\ B[\,], a_{i+1}\dots a_n) \\ & \overset{*}{\vdash} (\Upsilon\ C[\,], a_{j+1}\dots a_n)\end{array}$$

La representación gráfica de las derivaciones de puntos especiales se muestra en la figura 9.5. Para cualquier $\Upsilon' \in (V_S[V_I^*])^*$ se cumple que

$$(\Upsilon'\ B[\,], a_{i+1}\dots a_n) \overset{*}{\vdash} (\Upsilon'\ C[\,], a_{j+1}\dots a_n)$$

por lo que este tipo de derivaciones puede ser representado mediante ítems de la forma

$$[-, - \mid B, i, -, C, j, - \mid -, -, -, -]$$

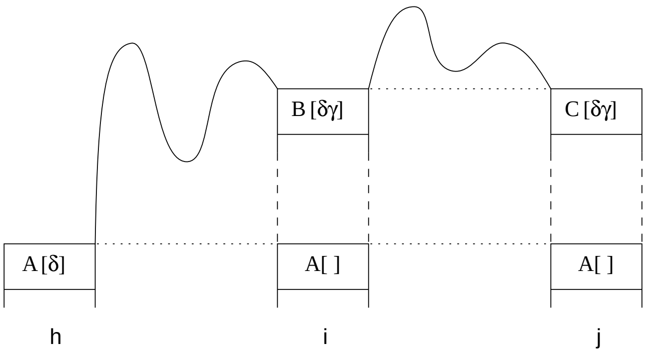

Figura 9.3: Derivaciones de llamada en L–LIA

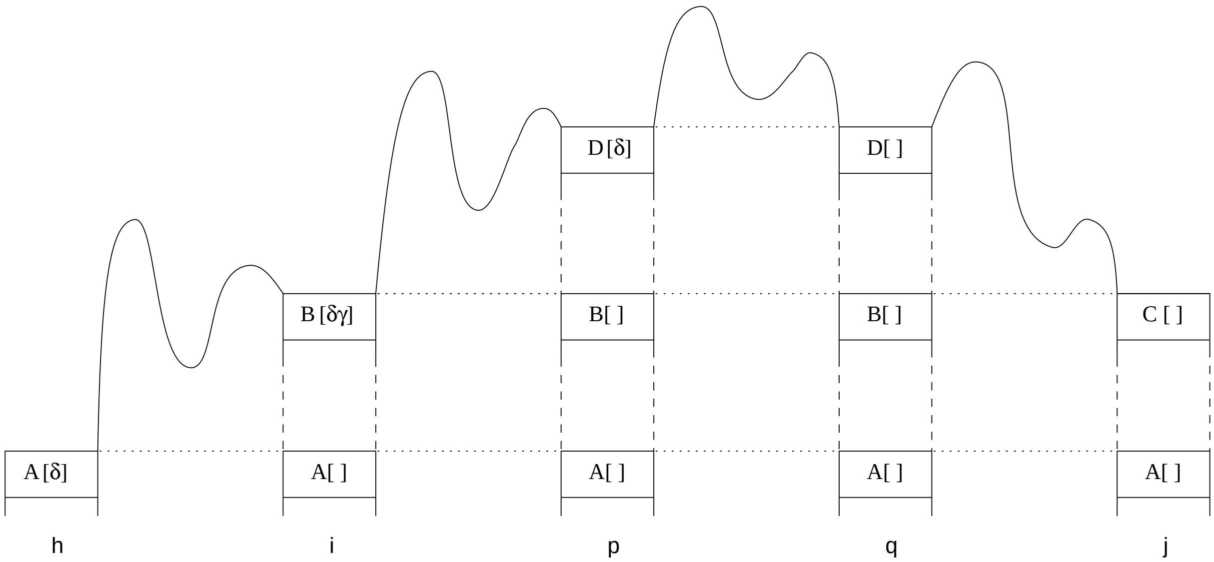

Figura 9.4: Derivaciones de retorno en L–LIA

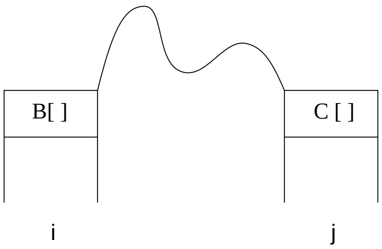

Figura 9.5: Derivaciones de puntos especiales en L–LIA

En la tablas 9.13 y 9.14 se muestran las reglas de combinación de ítems para los diferentes tipos de transiciones. El ítem inicial es

$$[-,- \mid \$_0, 0, -, \$_0, 0, - \mid -,-,-,-]$$

y los ítems finales son de la forma

$$[-,- \mid B, 0, -, \$_f, n, - \mid -,-,-,-]$$

tal que existe una transición $\$_0[\circ\circ] \longmapsto \$_0[\,]\; B[\circ\circ]$ o bien una transición $\$_0[\circ\circ] \longmapsto \$_0[\circ\circ]\; B[\,]$.

Ejemplo 9.1 Para ilustrar la técnica de tabulación propuesta para L–LIA, mostraremos un ejemplo de interpretación tabular del autómata cuyas transiciones se muestran en la tabla 9.15. Dicho autómata, que acepta el lenguaje $\{a^n b^n c^n d^n \mid n \geq 1\}$ está basado en el L–LIA utilizado como ejemplo por Nederhof en [126]. La única modificación realizada ha sido la adaptación de las transiciones al juego de transiciones propuesto en este capítulo. En la parte derecha de la tabla 9.15 se muestra la derivación de la cadena *aabbccdd* por el autómata definido en la parte izquierda de la dicha tabla. La primera columna indica la transición aplicada, la segunda el contenido de la pila del autómata y la tercera la parte de la cadena de entrada que resta por leer. La interpretación tabular de dicha derivación se muestra en la tabla 9.16, en la cual la primera columna se utiliza para identificar los ítems producidos, la segunda muestra el ítem obtenido en cada paso y la tercera muestra los ítems antecedentes y las transiciones involucradas en la generación de dicho ítem. ¶

Teorema 9.2 *La manipulación de configuraciones mediante la aplicación de transiciones en los autómatas lineales de índices orientados a la izquierda es equivalente a la manipulación de ítems mediante las reglas de combinación de las tablas 9.13 y 9.14.*

<u>Demostración</u>:

Puesto que un ítem representa una derivación y toda derivación debe ser representada por algún ítem, es suficiente con demostrar que la combinación de los ítems produce ítems que se corresponden con derivaciones válidas y que para toda derivación que se pueda producir como resultado de la aplicación de una transición, existe una regla de combinación de ítems que produce un ítem que representa a dicha derivación. A continuación se muestra una lista de todos los casos posibles de derivación que se puedan dar junto con la correspondiente regla de combinación de ítems.

- Derivaciones que son el resultado de aplicar una transición $C[\circ\circ] \overset{a}{\longmapsto} F[\circ\circ]$
  - a una derivación de llamada:

$$\begin{array}{rl} (\Upsilon\; A[\delta], a_{h+1}\ldots a_n) & \overset{*}{\vdash} (\Upsilon\; A[\,]\; \Upsilon_1\; B[\delta\gamma], a_{i+1}\ldots a_n) \\ & \overset{*}{\vdash} (\Upsilon\; A[\,]\; \Upsilon_1\; C[\delta\gamma], a_{j+1}\ldots a_n) \\ & \vdash (\Upsilon\; A[\,]\; \Upsilon_1\; F[\delta\gamma], a_{k+1}\ldots a_n) \end{array}$$

$$\frac{[A, h \mid B, i, \gamma, C, j, \gamma \mid -,-,-,-]}{[A, h \mid B, i, \gamma, F, k, \gamma \mid -,-,-,-]} \;\; C[\circ\circ] \overset{a}{\longmapsto} F[\circ\circ],\;\; k = j \text{ si } a = \epsilon,\;\; k = j+1 \text{ si } a \in V_T$$

$$\frac{[A,h \mid B,i,\gamma,C,j,\gamma \mid -,-,-,-]}{[A,h \mid B,i,\gamma,F,k,\gamma \mid -,-,-,-]} \quad C[\circ\circ] \stackrel{a}{\longmapsto} F[\circ\circ],\ k=j \text{ si } a=\epsilon,\ k=j+1 \text{ si } a \in V_T$$

$$\frac{[A,h \mid B,i,\gamma,C,j,- \mid D,p,E,q]}{[A,h \mid B,i,\gamma,F,k,- \mid D,p,E,q]} \quad C[\circ\circ] \stackrel{a}{\longmapsto} F[\circ\circ],\ k=j \text{ si } a=\epsilon,\ k=j+1 \text{ si } a \in V_T$$

$$\frac{[A,h \mid B,i,\gamma,C,j,\gamma \mid -,-,-,-]}{[-,- \mid F,j,-,F,j,- \mid -,-,-,-]} \quad C[\circ\circ] \longmapsto C[\circ\circ]\ F[\,]$$

$$\frac{[A,h \mid B,i,\gamma,C,j,- \mid D,p,E,q]}{[-,- \mid F,j,-,F,j,- \mid -,-,-,-]} \quad C[\circ\circ] \longmapsto C[\circ\circ]\ F[\,]$$

$$\frac{[A,h \mid B,i,\gamma,C,j,\gamma \mid -,-,-,-]}{[A,h \mid F,j,\gamma,F,j,\gamma \mid -,-,-,-]} \quad C[\circ\circ] \longmapsto C[\,]\ F[\circ\circ]$$

$$\frac{[A,h \mid B,i,\gamma,C,j,- \mid D,p,E,q]}{[-,- \mid F,j,-,F,j,- \mid -,-,-,-]} \quad C[\circ\circ] \longmapsto C[\,]\ F[\circ\circ]$$

$$\frac{[A,h \mid B,i,\gamma,C,j,\gamma \mid -,-,-,-]}{[C,j \mid F,j,\gamma',F,j,\gamma' \mid -,-,-,-]} \quad C[\circ\circ] \longmapsto C[\,]\ F[\circ\circ\gamma']$$

$$\frac{[A,h \mid B,i,\gamma,C,j,- \mid D,p,E,q]}{[C,j \mid F,j,\gamma',F,j,\gamma' \mid -,-,-,-]} \quad C[\circ\circ] \longmapsto C[\,]\ F[\circ\circ\gamma']$$

$$\frac{\begin{array}{c}[A,h \mid B,i,\gamma,C,j,\gamma \mid -,-,-,-] \\ [M,m \mid N,t,\gamma',A,h,\gamma' \mid -,-,-,-]\end{array}}{[M,m \mid F,j,\gamma',F,j,\gamma' \mid -,-,-,-]} \quad C[\circ\circ\gamma] \longmapsto C[\,]\ F[\circ\circ]$$

$$\frac{\begin{array}{c}[A,h \mid B,i,\gamma,C,j,\gamma \mid -,-,-,-] \\ [M,m \mid N,t,\gamma',A,h,- \mid D,p,E,q]\end{array}}{[-,- \mid F,j,-,F,j,- \mid -,-,-,-]} \quad C[\circ\circ\gamma] \longmapsto C[\,]\ F[\circ\circ]$$

Tabla 9.13: Combinación de ítems en L–LIA (fase de llamada)

$$\frac{\begin{array}{c}[-,- \mid F',j,-,F,k,- \mid -,-,-,-]\\ [A,h \mid B,i,\gamma,C,j,\gamma \mid -,-,-,-]\end{array}}{[A,h \mid B,i,\gamma,G,k,\gamma \mid -,-,-,-]} \quad \begin{array}{l}C[\circ\circ] \longmapsto C[\circ\circ]\ F'[\,]\\ C[\circ\circ]\ F[\,] \longmapsto G[\circ\circ]\end{array}$$

$$\frac{\begin{array}{c}[-,- \mid F',j,-,F,k,- \mid -,-,-,-]\\ [A,h \mid B,i,\gamma,C,j,- \mid D,p,E,q]\end{array}}{[A,h \mid B,i,\gamma,G,k,- \mid D,p,E,q]} \quad \begin{array}{l}C[\circ\circ] \longmapsto C[\circ\circ]\ F'[\,]\\ C[\circ\circ]\ F[\,] \longmapsto G[\circ\circ]\end{array}$$

$$\frac{\begin{array}{c}[A,h \mid F',j,\gamma,F,k,- \mid D,p,E,q]\\ [A,h \mid B,i,\gamma,C,j,\gamma \mid -,-,-,-]\end{array}}{[A,h \mid B,i,\gamma,G,k,- \mid D,p,E,q]} \quad \begin{array}{l}C[\circ\circ] \longmapsto C[\,]\ F'[\circ\circ]\\ C[\circ\circ]\ F[\,] \longmapsto G[\circ\circ]\end{array}$$

$$\frac{\begin{array}{c}[-,- \mid F',j,-,F,k,- \mid -,-,-,-]\\ [A,h \mid B,i,\gamma,C,j,- \mid D,p,E,q]\end{array}}{[A,h \mid B,i,\gamma,G,k,- \mid D,p,E,q]} \quad \begin{array}{l}C[\circ\circ] \longmapsto C[\,]\ F'[\circ\circ]\\ C[\circ\circ]\ F[\,] \longmapsto G[\circ\circ]\end{array}$$

$$\frac{\begin{array}{c}[C,j \mid F',j,\gamma',F,k,- \mid D,p,E,q]\\ [A,h \mid B,i,\gamma,C,j,\gamma \mid -,-,-,-]\\ [A,h \mid D,p,\gamma,E,q,- \mid O,u,P,v]\end{array}}{[A,h \mid B,i,\gamma,G,k,- \mid O,u,P,v]} \quad \begin{array}{l}C[\circ\circ] \longmapsto C[\,]\ F'[\circ\circ\gamma']\\ C[\circ\circ]\ F[\,] \longmapsto G[\circ\circ]\end{array}$$

$$\frac{\begin{array}{c}[C,j \mid F',j,\gamma',F,k,- \mid O,u,P,v]\\ [A,h \mid B,i,\gamma,C,j,- \mid D,p,E,q]\\ [-,- \mid O,u,-,P,v,- \mid -,-,-,-]\end{array}}{[A,h \mid B,i,\gamma,G,k,- \mid D,p,E,q]} \quad \begin{array}{l}C[\circ\circ] \longmapsto C[\,]\ F'[\circ\circ\gamma']\\ C[\circ\circ]\ F[\,] \longmapsto G[\circ\circ]\end{array}$$

$$\frac{\begin{array}{c}[M,m \mid F',j,\gamma',F,k,- \mid D,p,E,q]\\ [A,h \mid B,i,\gamma,C,j,\gamma \mid -,-,-,-]\\ [M,m \mid N,t,\gamma',A,h,\gamma' \mid -,-,-,-]\end{array}}{[A,h \mid B,i,\gamma,G,k,- \mid F',j,F,k]} \quad \begin{array}{l}C[\circ\circ\gamma] \longmapsto C[\,]\ F'[\circ\circ]\\ C[\circ\circ]\ F[\,] \longmapsto G[\circ\circ]\end{array}$$

$$\frac{\begin{array}{c}[-,- \mid F',j,-,F,k,- \mid -,-,-,-]\\ [A,h \mid B,i,\gamma,C,j,\gamma \mid -,-,-,-]\\ [M,m \mid N,t,\gamma',A,h,- \mid D,p,E,q]\end{array}}{[A,h \mid B,i,\gamma,G,k,- \mid F',j,F,k]} \quad \begin{array}{l}C[\circ\circ\gamma] \longmapsto C[\,]\ F'[\circ\circ]\\ C[\circ\circ]\ F[\,] \longmapsto G[\circ\circ]\end{array}$$

Tabla 9.14: Combinación de ítems en L–LIA (fase de retorno)

(a)	$\$_0[\circ\circ] \longmapsto \$_0[\circ\circ]\ S[\,]$
(b)	$S[\circ\circ] \longmapsto X[\circ\circ]$
(c)	$X[\circ\circ] \longmapsto X[\circ\circ]\ A[\,]$
(d)	$A[\circ\circ] \overset{a}{\longmapsto} A'[\circ\circ]$
(e)	$X[\circ\circ]\ A'[\,] \longmapsto X'[\circ\circ]$
(f)	$X'[\circ\circ] \longmapsto D[\circ\circ]$
(g)	$D[\circ\circ] \longmapsto D[\,]\ Y[\circ\circ\gamma]$
(h)	$Y[\circ\circ] \longmapsto X[\circ\circ]$
(i)	$Y[\circ\circ] \longmapsto Z[\circ\circ]$
(j)	$Z[\circ\circ] \longmapsto Z[\circ\circ]\ B[\,]$
(k)	$B[\circ\circ] \overset{b}{\longmapsto} B'[\circ\circ]$
(l)	$Z[\circ\circ]\ B'[\,] \longmapsto Z'[\circ\circ]$
(m)	$Z'[\circ\circ] \longmapsto C[\circ\circ]$
(n)	$C[\circ\circ\gamma] \longmapsto C[\,]\ P[\circ\circ]$
(p)	$P[\circ\circ] \longmapsto Z[\circ\circ]$
(q)	$C[\circ\circ]\ P[\,] \longmapsto C'[\circ\circ]$
(r)	$C'[\circ\circ] \overset{c}{\longmapsto} C''[\circ\circ]$
(s)	$C[\circ\circ]\ C''[\,] \longmapsto C'[\circ\circ]$
(t)	$D[\circ\circ]\ C''[\,] \longmapsto D'[\circ\circ]$
(u)	$D'[\circ\circ] \overset{d}{\longmapsto} D''[\circ\circ]$
(v)	$D[\circ\circ]\ D''[\,] \longmapsto D'[\circ\circ]$
(w)	$D''[\circ\circ] \longmapsto \$_f[\circ\circ]$

	$\$_0[\,]$	$aabbccdd$
(a)	$\$_0[\,]\ S[\,]$	$aabbccdd$
(b)	$\$_0[\,]\ X[\,]$	$aabbccdd$
(c)	$\$_0[\,]\ X[\,]\ A[\,]$	$aabbccdd$
(d)	$\$_0[\,]\ X[\,]\ A'[\,]$	$abbccdd$
(e)	$\$_0[\,]\ X'[\,]$	$abbccdd$
(f)	$\$_0[\,]\ D[\,]$	$abbccdd$
(g)	$\$_0[\,]\ D[\,]\ Y[\gamma]$	$abbccdd$
(h)	$\$_0[\,]\ D[\,]\ X[\gamma]$	$abbccdd$
(c)	$\$_0[\,]\ D[\,]\ X[\gamma]\ A[\,]$	$abbccdd$
(d)	$\$_0[\,]\ D[\,]\ X[\gamma]\ A'[\,]$	$bbccdd$
(e)	$\$_0[\,]\ D[\,]\ X'[\gamma]$	$bbccdd$
(f)	$\$_0[\,]\ D[\,]\ D[\gamma]$	$bbccdd$
(g)	$\$_0[\,]\ D[\,]\ D[\,]\ Y[\gamma\gamma]$	$bbccdd$
(i)	$\$_0[\,]\ D[\,]\ D[\,]\ Z[\gamma\gamma]$	$bbccdd$
(j)	$\$_0[\,]\ D[\,]\ D[\,]\ Z[\gamma\gamma]\ B[\,]$	$bbccdd$
(k)	$\$_0[\,]\ D[\,]\ D[\,]\ Z[\gamma\gamma]\ B'[\,]$	$bccdd$
(l)	$\$_0[\,]\ D[\,]\ D[\,]\ Z'[\gamma\gamma]$	$bccdd$
(m)	$\$_0[\,]\ D[\,]\ D[\,]\ C[\gamma\gamma]$	$bccdd$
(n)	$\$_0[\,]\ D[\,]\ D[\,]\ C[\,]\ P[\gamma]$	$bccdd$
(p)	$\$_0[\,]\ D[\,]\ D[\,]\ C[\,]\ Z[\gamma]$	$bccdd$
(j)	$\$_0[\,]\ D[\,]\ D[\,]\ C[\,]\ Z[\gamma]\ B[\,]$	$bccdd$
(k)	$\$_0[\,]\ D[\,]\ D[\,]\ C[\,]\ Z[\gamma]\ B'[\,]$	$ccdd$
(l)	$\$_0[\,]\ D[\,]\ D[\,]\ C[\,]\ Z'[\gamma]$	$ccdd$
(m)	$\$_0[\,]\ D[\,]\ D[\,]\ C[\,]\ C[\gamma]$	$ccdd$
(n)	$\$_0[\,]\ D[\,]\ D[\,]\ C[\,]\ C[\,]\ P[\,]$	$ccdd$
(q)	$\$_0[\,]\ D[\,]\ D[\,]\ C[\,]\ C'[\,]$	$ccdd$
(r)	$\$_0[\,]\ D[\,]\ D[\,]\ C[\,]\ C''[\,]$	cdd
(s)	$\$_0[\,]\ D[\,]\ D[\,]\ C'[\,]$	cdd
(r)	$\$_0[\,]\ D[\,]\ D[\,]\ C''[\,]$	dd
(t)	$\$_0[\,]\ D[\,]\ D'[\,]$	dd
(u)	$\$_0[\,]\ D[\,]\ D''[\,]$	d
(v)	$\$_0[\,]\ D'[\,]$	d
(u)	$\$_0[\,]\ D''[\,]$	
(w)	$\$_0[\,]\ \$_f[\,]$	

Tabla 9.15: Transiciones de un L–LIA que acepta $\{a^n b^n c^n d^n \mid n > 0\}$ (izquierda) y derivación de la cadena *aabbccdd* en dicha gramática (derecha)

	Ítem	Origen
0	$[-,- \mid \$_0, 0, -, \$_0, 0, - \mid -,-,-,-]$	
1	$[-,- \mid S, 0, -, S, 0, - \mid -,-,-,-]$	$0 + \$_0[\circ\circ] \longmapsto \$_0[\circ\circ]\ S[\,]$
2	$[-,- \mid S, 0, -, X, 0, - \mid -,-,-,-]$	$1 + S[\circ\circ] \longmapsto X[\circ\circ]$
3	$[-,- \mid A, 0, -, A, 0, - \mid -,-,-,-]$	$2 + X[\circ\circ] \longmapsto X[\circ\circ]\ A[\,]$
4	$[-,- \mid A, 0, -, A', 1, - \mid -,-,-,-]$	$3 + A[\circ\circ] \overset{a}{\longmapsto} A'[\circ\circ]$
5	$[-,- \mid S, 0, -, X', 1, - \mid -,-,-,-]$	$4 + 2 + X[\circ\circ] \longmapsto X[\circ\circ]\ A[\,] + X[\circ\circ]\ A'[\,] \longmapsto X'[\circ\circ]$
6	$[-,- \mid S, 0, -, D, 1, - \mid -,-,-,-]$	$5 + X'[\circ\circ] \longmapsto D[\circ\circ]$
7	$[D, 1 \mid Y, 1, \gamma, Y, 1, \gamma \mid -,-,-,-]$	$6 + D[\circ\circ] \longmapsto D[\,]\ Y[\circ\circ\gamma]$
8	$[D, 1 \mid Y, 1, \gamma, X, 1, \gamma \mid -,-,-,-]$	$7 + Y[\circ\circ] \longmapsto X[\circ\circ]$
9	$[-,- \mid A, 1, -, A, 1, - \mid -,-,-,-]$	$8 + X[\circ\circ] \longmapsto X[\circ\circ]\ A[\,]$
10	$[-,- \mid A, 1, -, A', 2, - \mid -,-,-,-]$	$9 + A[\circ\circ] \overset{a}{\longmapsto} A'[\circ\circ]$
11	$[D, 1 \mid Y, 1, \gamma, X', 2, \gamma \mid -,-,-,-]$	$10 + 8 + X[\circ\circ] \longmapsto X[\circ\circ]\ A[\,] + X[\circ\circ]\ A'[\,] \longmapsto X'[\circ\circ]$
12	$[D, 1 \mid Y, 1, \gamma, D, 2, \gamma \mid -,-,-,-]$	$11 + X'[\circ\circ] \longmapsto D[\circ\circ]$
13	$[D, 2 \mid Y, 2, \gamma, Y, 2, \gamma \mid -,-,-,-]$	$12 + D[\circ\circ] \longmapsto D[\,]\ Y[\circ\circ\gamma]$
14	$[D, 2 \mid Y, 2, \gamma, Z, 2, \gamma \mid -,-,-,-]$	$13 + Y[\circ\circ] \longmapsto Z[\circ\circ]$
15	$[-,- \mid B, 2, -, B, 2, - \mid -,-,-,-]$	$14 + Z[\circ\circ] \longmapsto Z[\circ\circ]\ B[\,]$
16	$[-,- \mid B, 2, -, B', 3, - \mid -,-,-,-]$	$15 + B[\circ\circ] \overset{b}{\longmapsto} B'[\circ\circ]$
17	$[D, 2 \mid Y, 2, \gamma, Z', 3, \gamma \mid -,-,-,-]$	$16 + 14 + Z[\circ\circ] \longmapsto Z[\circ\circ]\ B[\,] + Z[\circ\circ]\ B'[\,] \longmapsto Z'[\circ\circ]$
18	$[D, 2 \mid Y, 2, \gamma, C, 3, \gamma \mid -,-,-,-]$	$17 + Z'[\circ\circ] \longmapsto C[\circ\circ]$
19	$[D, 1 \mid P, 3, \gamma, P, 3, \gamma \mid -,-,-,-]$	$18 + 12 + C[\circ\circ\gamma] \longmapsto C[\,]\ P[\circ\circ]$
20	$[D, 1 \mid P, 3, \gamma, Z, 3, \gamma \mid -,-,-,-]$	$19 + P[\circ\circ] \longmapsto Z[\circ\circ]$
21	$[-,- \mid B, 3, -, B, 3, - \mid -,-,-,-]$	$20 + Z[\circ\circ] \longmapsto Z[\circ\circ]\ B[\,]$
22	$[-,- \mid B, 3, -, B', 4, - \mid -,-,-,-]$	$21 + B[\circ\circ] \overset{b}{\longmapsto} B'[\circ\circ]$
23	$[D, 1 \mid P, 3, \gamma, Z', 4, \gamma \mid -,-,-,-]$	$22 + 20 + Z[\circ\circ] \longmapsto Z[\circ\circ]\ B[\,] + Z[\circ\circ]\ B'[\,] \longmapsto Z'[\circ\circ]$
24	$[D, 1 \mid P, 3, \gamma, C, 4, \gamma \mid -,-,-,-]$	$23 + Z'[\circ\circ] \longmapsto C[\circ\circ]$
25	$[-,- \mid P, 4, -, P, 4, - \mid -,-,-,-]$	$24 + 6 + C[\circ\circ\gamma] \longmapsto C[\,]\ P[\circ\circ]$
26	$[D, 1 \mid P, 3, \gamma, C', 4, - \mid P, 4, P, 4]$	$25 + 24 + 6 + C[\circ\circ\gamma] \longmapsto C[\,]\ P[\circ\circ] + C[\circ\circ]\ P[\,] \longmapsto C'[\circ\circ]$
27	$[D, 1 \mid P, 3, \gamma, C'', 5, - \mid P, 4, P, 4]$	$26 + C'[\circ\circ] \overset{c}{\longmapsto} C''[\circ\circ]$
28	$[D, 2 \mid Y, 2, \gamma, C', 5, - \mid P, 3, C'', 5]$	$27 + 18 + 12 + C[\circ\circ\gamma] \longmapsto C[\,]\ P[\circ\circ] + C[\circ\circ]\ C''[\,] \longmapsto C'[\circ\circ]$
29	$[D, 2 \mid Y, 2, \gamma, C'', 6, - \mid P, 3, C'', 5]$	$28 + C'[\circ\circ] \overset{c}{\longmapsto} C''[\circ\circ]$
30	$[D, 1 \mid Y, 1, \gamma, D', 6, - \mid P, 4, P, 4]$	$29 + 12 + 27 + D[\circ\circ] \longmapsto D[\,]\ Y[\circ\circ\gamma] + D[\circ\circ]\ C''[\,] \longmapsto D'[\circ\circ]$
31	$[D, 1 \mid Y, 1, \gamma, D'', 7, - \mid P, 4, P, 4]$	$30 + D'[\circ\circ] \overset{d}{\longmapsto} D''$
32	$[-,- \mid S, 0, -, D', 7, - \mid -,-,-,-]$	$31 + 6 + 25 + D[\circ\circ] \longmapsto D[\,]\ Y[\circ\circ\gamma] + D[\circ\circ]\ D''[\,] \longmapsto D'$
33	$[-,- \mid S, 0, -, D'', 8, - \mid -,-,-,-]$	$32 + D'[\circ\circ] \overset{d}{\longmapsto} D''$
34	$[-,- \mid S, 0, -, \$_f, 8, - \mid -,-,-,-]$	$33 + D''[\circ\circ] \longmapsto \$_f[\circ\circ]$

Tabla 9.16: Ítems generados durante el reconocimiento de *aabbccdd*

– a una derivación de retorno:

$$\begin{array}{rl}(\Upsilon\ A[\delta], a_{h+1}\ldots a_n) & \overset{*}{\vdash} (\Upsilon\ A[\,]\ \Upsilon_1\ B[\delta\gamma], a_{i+1}\ldots a_n)\\ & \overset{*}{\vdash} (\Upsilon\ A[\,]\ \Upsilon_1\ B[\,]\ \Upsilon_2\ D[\delta], a_{p+1}\ldots a_n)\\ & \overset{*}{\vdash} (\Upsilon\ A[\,]\ \Upsilon_1\ B[\,]\ \Upsilon_2\ E[\,], a_{q+1}\ldots a_n)\\ & \overset{*}{\vdash} (\Upsilon\ A[\,]\ \Upsilon_1\ C[\,], a_{j+1}\ldots a_n)\\ & \vdash (\Upsilon\ A[\,]\ \Upsilon_1\ F[\,], a_{k+1}\ldots a_n)\end{array}$$

$$\frac{[A,h \mid B,i,\gamma,C,j,- \mid D,p,E,q]}{[A,h \mid B,i,\gamma,F,k,- \mid D,p,E,q]}\ C[\circ\circ] \overset{a}{\longmapsto} F[\circ\circ],\ k=j \text{ si } a=\epsilon,\ k=j+1 \text{ si } a\in V_T$$

– a una derivación de puntos especiales:

$$\begin{array}{rl}(\Upsilon\ B[\,], a_{i+1}\ldots a_n) & \overset{*}{\vdash} (\Upsilon\ C[\,], a_{j+1}\ldots a_n)\\ & \vdash (\Upsilon\ F[\,], a_{k+1}\ldots a_n)\end{array}$$

$$\frac{[-,- \mid B,i,-,C,j,- \mid -,-,-,-]}{[-,- \mid B,i,-,F,k,- \mid -,-,-,-]}\ C[\circ\circ] \overset{a}{\longmapsto} F[\circ\circ],\ k=j \text{ si } a=\epsilon,\ k=j+1 \text{ si } a\in V_T$$

- Derivaciones que son el resultado de aplicar una transición $C[\circ\circ] \longmapsto C[\circ\circ]\ F[\,]$
  – a una derivación de llamada:

$$\begin{array}{rl}(\Upsilon\ A[\delta], a_{h+1}\ldots a_n) & \overset{*}{\vdash} (\Upsilon\ A[\,]\ \Upsilon_1\ B[\delta\gamma], a_{i+1}\ldots a_n)\\ & \overset{*}{\vdash} (\Upsilon\ A[\,]\ \Upsilon_1\ C[\delta\gamma], a_{j+1}\ldots a_n)\\ & \vdash (\Upsilon\ A[\,]\ \Upsilon_1\ C[\delta\gamma]\ F[\,], a_{j+1}\ldots a_n)\end{array}$$

$$\frac{[A,h \mid B,i,\gamma,C,j,\gamma \mid -,-,-,-]}{[-,- \mid F,j,-,F,j,- \mid -,-,-,-]}\ C[\circ\circ] \longmapsto C[\circ\circ]\ F[\,]$$

  – a una derivación de retorno:

$$\begin{array}{rl}(\Upsilon\ A[\delta], a_{h+1}\ldots a_n) & \overset{*}{\vdash} (\Upsilon\ A[\,]\ \Upsilon_1\ B[\delta\gamma], a_{i+1}\ldots a_n)\\ & \overset{*}{\vdash} (\Upsilon\ A[\,]\ \Upsilon_1\ B[\,]\ \Upsilon_2\ D[\delta], a_{p+1}\ldots a_n)\\ & \overset{*}{\vdash} (\Upsilon\ A[\,]\ \Upsilon_1\ B[\,]\ \Upsilon_2\ E[\,], a_{q+1}\ldots a_n)\\ & \overset{*}{\vdash} (\Upsilon\ A[\,]\ \Upsilon_1\ C[\,], a_{j+1}\ldots a_n)\\ & \vdash (\Upsilon\ A[\,]\ \Upsilon_1\ C[\,]\ F[\,], a_{j+1}\ldots a_n)\end{array}$$

$$\frac{[A,h \mid B,i,\gamma,C,j,- \mid D,p,E,q]}{[-,- \mid F,j,-,F,j,- \mid -,-,-,-]}\ C[\circ\circ] \longmapsto C[\circ\circ]\ F[\,]$$

  – a una derivación de puntos especiales:

$$\begin{array}{rl}(\Upsilon\ B[\,], a_{i+1}\ldots a_n) & \overset{*}{\vdash} (\Upsilon\ C[\,], a_{j+1}\ldots a_n)\\ & \vdash (\Upsilon\ C[\,]\ F[\,], a_{j+1}\ldots a_n)\end{array}$$

$$\frac{[-,- \mid B,i,-,C,j,- \mid -,-,-,-]}{[-,- \mid F,j,-,F,j,- \mid -,-,-,-]}\ C[\circ\circ] \longmapsto C[\circ\circ]\ F[\,]$$

- Derivaciones que son el resultado de aplicar una transición $C[\circ\circ] \longmapsto C[\,]\ F[\circ\circ]$
  – a una derivación de llamada:

$$\begin{array}{rl}(\Upsilon\ A[\delta], a_{h+1}\ldots a_n) & \overset{*}{\vdash} (\Upsilon\ A[\,]\ \Upsilon_1\ B[\delta\gamma], a_{i+1}\ldots a_n)\\ & \overset{*}{\vdash} (\Upsilon\ A[\,]\ \Upsilon_1\ C[\delta\gamma], a_{j+1}\ldots a_n)\\ & \vdash (\Upsilon\ A[\,]\ \Upsilon_1\ C[\,]\ F[\delta\gamma], a_{j+1}\ldots a_n)\end{array}$$

$$\frac{[A,h \mid B,i,\gamma,C,j,\gamma \mid -,-,-,-]}{[A,h \mid F,j,\gamma,F,j,\gamma \mid -,-,-,-]}\ C[\circ\circ] \longmapsto C[\,]\ F[\circ\circ]$$

– a una derivación de retorno:

$$\begin{array}{rl} (\Upsilon\ A[\delta], a_{h+1}\dots a_n) & \overset{*}{\vdash} (\Upsilon\ A[\,]\ \Upsilon_1\ B[\delta\gamma], a_{i+1}\dots a_n) \\ & \overset{*}{\vdash} (\Upsilon\ A[\,]\ \Upsilon_1\ B[\,]\ \Upsilon_2\ D[\delta], a_{p+1}\dots a_n) \\ & \overset{*}{\vdash} (\Upsilon\ A[\,]\ \Upsilon_1\ B[\,]\ \Upsilon_2\ E[\,], a_{q+1}\dots a_n) \\ & \overset{*}{\vdash} (\Upsilon\ A[\,]\ \Upsilon_1\ C[\,], a_{j+1}\dots a_n) \\ & \vdash (\Upsilon\ A[\,]\ \Upsilon_1\ C[\,]\ F[\,], a_{j+1}\dots a_n) \end{array}$$

$$\frac{[A, h \mid B, i, \gamma, C, j, - \mid D, p, E, q]}{[-, - \mid F, j, -, F, j, - \mid -, -, -, -]}\ \ C[\circ\circ] \longmapsto C[\,]\ F[\circ\circ]$$

– a una derivación de puntos especiales:

$$\begin{array}{rl} (\Upsilon\ B[\,], a_{i+1}\dots a_n) & \overset{*}{\vdash} (\Upsilon\ C[\,], a_{j+1}\dots a_n) \\ & \vdash (\Upsilon\ C[\,]\ F[\,], a_{j+1}\dots a_n) \end{array}$$

$$\frac{[-, - \mid B, i, -, C, j, - \mid -, -, -, -]}{[-, - \mid F, j, -, F, j, - \mid -, -, -, -]}\ \ C[\circ\circ] \longmapsto C[\,]\ F[\circ\circ]$$

- Derivaciones que son el resultado de aplicar una transición $C[\circ\circ] \longmapsto C[\,]\ F[\circ\circ\gamma']$
  – a una derivación de llamada:

$$\begin{array}{rl} (\Upsilon\ A[\delta], a_{h+1}\dots a_n) & \overset{*}{\vdash} (\Upsilon\ A[\,]\ \Upsilon_1\ B[\delta\gamma], a_{i+1}\dots a_n) \\ & \overset{*}{\vdash} (\Upsilon\ A[\,]\ \Upsilon_1\ C[\delta\gamma], a_{j+1}\dots a_n) \\ & \vdash (\Upsilon\ A[\,]\ \Upsilon_1\ C[\,]\ F[\delta\gamma\gamma'], a_{j+1}\dots a_n) \end{array}$$

$$\frac{[A, h \mid B, i, \gamma, C, j, \gamma \mid -, -, -, -]}{[C, j \mid F, j, \gamma', F, j, \gamma' \mid -, -, -, -]}\ \ C[\circ\circ] \longmapsto C[\,]\ F[\circ\circ\gamma']$$

  – a una derivación de retorno:

$$\begin{array}{rl} (\Upsilon\ A[\delta], a_{h+1}\dots a_n) & \overset{*}{\vdash} (\Upsilon\ A[\,]\ \Upsilon_1\ B[\delta\gamma], a_{i+1}\dots a_n) \\ & \overset{*}{\vdash} (\Upsilon\ A[\,]\ \Upsilon_1\ B[\,]\ \Upsilon_2\ D[\delta], a_{p+1}\dots a_n) \\ & \overset{*}{\vdash} (\Upsilon\ A[\,]\ \Upsilon_1\ B[\,]\ \Upsilon_2\ E[\,], a_{q+1}\dots a_n) \\ & \overset{*}{\vdash} (\Upsilon\ A[\,]\ \Upsilon_1\ C[\,], a_{j+1}\dots a_n) \\ & \vdash (\Upsilon\ A[\,]\ \Upsilon_1\ C[\,]\ F[\gamma'], a_{j+1}\dots a_n) \end{array}$$

$$\frac{[A, h \mid B, i, \gamma, C, j, - \mid D, p, E, q]}{[C, j \mid F, j, \gamma', F, j, \gamma' \mid -, -, -, -]}\ \ C[\circ\circ] \longmapsto C[\,]\ F[\circ\circ\gamma']$$

  – a una derivación de puntos especiales:

$$\begin{array}{rl} (\Upsilon\ B[\,], a_{i+1}\dots a_n) & \overset{*}{\vdash} (\Upsilon\ C[\,], a_{j+1}\dots a_n) \\ & \vdash (\Upsilon\ C[\,]\ F[\gamma'], a_{j+1}\dots a_n) \end{array}$$

$$\frac{[-, - \mid B, i, -, C, j, - \mid -, -, -, -]}{[C, j \mid F, j, \gamma', F, j, \gamma' \mid -, -, -, -]}\ \ C[\circ\circ] \longmapsto C[\,]\ F[\circ\circ\gamma']$$

- Derivaciones que son el resultado de aplicar una transición $C[\circ\circ\gamma] \longmapsto C[\,]\ F[\circ\circ]$ a una derivación de llamada, con los tres casos siguientes:
  – la derivación de llamada es a su vez derivada a partir de una derivación de llamada:

$$\begin{array}{rl} (\Upsilon\ M[\delta], a_{m+1}\dots a_n) & \overset{*}{\vdash} (\Upsilon\ M[\,]\ \Upsilon_1\ N[\delta\gamma'], a_{t+1}\dots a_n) \\ & \overset{*}{\vdash} (\Upsilon\ M[\,]\ \Upsilon_1\ A[\delta\gamma'], a_{h+1}\dots a_n) \\ & \overset{*}{\vdash} (\Upsilon\ M[\,]\ \Upsilon_1\ A[\,]\ \Upsilon_1\ B[\delta\gamma'\gamma], a_{i+1}\dots a_n) \\ & \overset{*}{\vdash} (\Upsilon\ M[\,]\ \Upsilon_1\ A[\,]\ \Upsilon_1\ C[\delta\gamma'\gamma], a_{j+1}\dots a_n) \\ & \vdash (\Upsilon\ M[\,]\ \Upsilon_1\ A[\,]\ \Upsilon_1\ C[\,]\ F[\delta\gamma'], a_{j+1}\dots a_n) \end{array}$$

$$\frac{\begin{array}{c}[A,h \mid B,i,\gamma,C,j,\gamma \mid -,-,-,-]\\ [M,m \mid N,t,\gamma',A,h,\gamma' \mid -,-,-,-]\end{array}}{[M,m \mid F,j,\gamma',F,j,\gamma' \mid -,-,-,-]} \quad C[\circ\circ\gamma] \longmapsto C[\,]\ F[\circ\circ]$$

– la derivación de llamada es a su vez derivada a partir de una derivación de retorno:

$$\begin{array}{ll}(\Upsilon\ M[\delta], a_{m+1}\dots a_n) & \overset{*}{\vdash} (\Upsilon\ M[\,]\ \Upsilon_1\ N[\delta\gamma'], a_{t+1}\dots a_n)\\ & \overset{*}{\vdash} (\Upsilon\ M[\,]\ \Upsilon_1\ N[\,]\ \Upsilon_2\ D[\delta], a_{p+1}\dots a_n)\\ & \overset{*}{\vdash} (\Upsilon\ M[\,]\ \Upsilon_1\ N[\,]\ \Upsilon_2\ E[\,], a_{q+1}\dots a_n)\\ & \overset{*}{\vdash} (\Upsilon\ M[\,]\ \Upsilon_1\ A[\,], a_{h+1}\dots a_n)\\ & \overset{*}{\vdash} (\Upsilon\ M[\,]\ \Upsilon_1\ A[\,]\ B[\gamma], a_{i+1}\dots a_n)\\ & \overset{*}{\vdash} (\Upsilon\ M[\,]\ \Upsilon_1\ A[\,]\ C[\gamma], a_{j+1}\dots a_n)\\ & \vdash (\Upsilon\ M[\,]\ \Upsilon_1\ A[\,]\ C[\,]\ F[\,], a_{j+1}\dots a_n)\end{array}$$

$$\frac{\begin{array}{c}[A,h \mid B,i,\gamma,C,j,\gamma \mid -,-,-,-]\\ [M,m \mid N,t,\gamma',A,h,- \mid D,p,E,q]\end{array}}{[-,- \mid F,j,-,F,j,- \mid -,-,-,-]} \quad C[\circ\circ\gamma] \longmapsto C[\,]\ F[\circ\circ]$$

– la derivación de llamada es a su vez derivada a partir de una derivación de puntos especiales:

$$\begin{array}{ll}(\Upsilon\ N[\,], a_{t+1}\dots a_n) & \overset{*}{\vdash} (\Upsilon\ A[\,], a_{h+1}\dots a_n)\\ & \overset{*}{\vdash} (\Upsilon\ A[\,]\ B[\gamma], a_{i+1}\dots a_n)\\ & \overset{*}{\vdash} (\Upsilon\ A[\,]\ C[\gamma], a_{j+1}\dots a_n)\\ & \vdash (\Upsilon\ A[\,]\ C[\,]\ F[\,], a_{j+1}\dots a_n)\end{array}$$

$$\frac{\begin{array}{c}[A,h \mid B,i,\gamma,C,j,\gamma \mid -,-,-,-]\\ [-,- \mid N,t,-,A,h,- \mid -,-,-,-]\end{array}}{[-,- \mid F,j,-,F,j,- \mid -,-,-,-]} \quad C[\circ\circ\gamma] \longmapsto C[\,]\ F[\circ\circ]$$

- Derivaciones que son el resultado de aplicar una transición $C[\circ\circ]\ F[\,] \longmapsto G[\circ\circ]$ a una derivación obtenida tras aplicar una transición $C[\circ\circ] \longmapsto C[\circ\circ]\ F'[\,]$
  - a una derivación de llamada:

$$\begin{array}{ll}(\Upsilon\ A[\delta], a_{h+1}\dots a_n) & \overset{*}{\vdash} (\Upsilon\ A[\,]\ \Upsilon_1\ B[\delta\gamma], a_{i+1}\dots a_n)\\ & \overset{*}{\vdash} (\Upsilon\ A[\,]\ \Upsilon_1\ C[\delta\gamma], a_{j+1}\dots a_n)\\ & \vdash (\Upsilon\ A[\,]\ \Upsilon_1\ C[\delta\gamma]\ F'[\,], a_{j+1}\dots a_n)\\ & \overset{*}{\vdash} (\Upsilon\ A[\,]\ \Upsilon_1\ C[\delta\gamma]\ F[\,], a_{k+1}\dots a_n)\\ & \vdash (\Upsilon\ A[\,]\ \Upsilon_1\ G[\delta\gamma], a_{k+1}\dots a_n)\end{array}$$

$$\frac{\begin{array}{c}[-,- \mid F',j,-,F,k,- \mid -,-,-,-]\\ [A,h \mid B,i,\gamma,C,j,\gamma \mid -,-,-,-]\end{array}}{[A,h \mid B,i,\gamma,G,k,\gamma \mid -,-,-,-]} \quad \begin{array}{l}C[\circ\circ] \longmapsto C[\circ\circ]\ F'[\,]\\ C[\circ\circ]\ F[\,] \longmapsto G[\circ\circ]\end{array}$$

  - a una derivación de retorno:

$$\begin{array}{ll}(\Upsilon\ A[\delta], a_{h+1}\dots a_n) & \overset{*}{\vdash} (\Upsilon\ A[\,]\ \Upsilon_1\ B[\delta\gamma], a_{i+1}\dots a_n)\\ & \overset{*}{\vdash} (\Upsilon\ A[\,]\ \Upsilon_1\ B[\,]\ \Upsilon_2\ D[\delta], a_{p+1}\dots a_n)\\ & \overset{*}{\vdash} (\Upsilon\ A[\,]\ \Upsilon_1\ B[\,]\ \Upsilon_2\ E[\,], a_{q+1}\dots a_n)\\ & \overset{*}{\vdash} (\Upsilon\ A[\,]\ \Upsilon_1\ C[\,], a_{j+1}\dots a_n)\\ & \vdash (\Upsilon\ A[\,]\ \Upsilon_1\ C[\,]\ F'[\,], a_{j+1}\dots a_n)\\ & \overset{*}{\vdash} (\Upsilon\ A[\,]\ \Upsilon_1\ C[\,]\ F[\,], a_{k+1}\dots a_n)\\ & \vdash (\Upsilon\ A[\,]\ \Upsilon_1\ G[\,], a_{k+1}\dots a_n)\end{array}$$

$$\frac{\begin{array}{c}[-,- \mid F',j,-,F,k,- \mid -,-,-,-]\\ [A,h \mid B,i,\gamma,C,j,- \mid D,p,E,q]\end{array}}{[A,h \mid B,i,\gamma,G,k,- \mid D,p,E,q]} \quad \begin{array}{l}C[\circ\circ] \longmapsto C[\circ\circ]\ F'[\,]\\ C[\circ\circ]\ F[\,] \longmapsto G[\circ\circ]\end{array}$$

– a una derivación de puntos especiales:

$$\begin{array}{rl}(\Upsilon\ B[\,], a_{i+1}\ldots a_n) & \overset{*}{\vdash} (\Upsilon\ C[\,], a_{j+1}\ldots a_n)\\ & \vdash (\Upsilon\ C[\,]\ F'[\,], a_{j+1}\ldots a_n)\\ & \overset{*}{\vdash} (\Upsilon\ C[\,]\ F[\,], a_{k+1}\ldots a_n)\\ & \vdash (\Upsilon\ G[\,], a_{k+1}\ldots a_n)\end{array}$$

$$\frac{\begin{array}{c}[-,- \mid F',j,-,F,k,- \mid -,-,-,-]\\ [-,- \mid B,i,\gamma,C,j,- \mid -,-,-,-]\end{array}}{[-,- \mid B,i,\gamma,G,k,- \mid -,-,-,-]} \quad \begin{array}{l}C[\circ\circ] \longmapsto C[\circ\circ]\ F'[\,]\\ C[\circ\circ]\ F[\,] \longmapsto G[\circ\circ]\end{array}$$

- Derivaciones que son el resultado de aplicar una transición $C[\circ\circ]\ F[\,] \longmapsto G[\circ\circ]$ a una derivación obtenida tras aplicar una transición $C[\circ\circ] \longmapsto C[\,]\ F'[\circ\circ]$
  – a una derivación de llamada:

$$\begin{array}{rl}(\Upsilon\ A[\delta], a_{h+1}\ldots a_n) & \overset{*}{\vdash} (\Upsilon\ A[\,]\ \Upsilon_1\ B[\delta\gamma], a_{i+1}\ldots a_n)\\ & \overset{*}{\vdash} (\Upsilon\ A[\,]\ \Upsilon_1\ C[\delta\gamma], a_{j+1}\ldots a_n)\\ & \vdash (\Upsilon\ A[\,]\ \Upsilon_1\ C[\,]\ F'[\delta\gamma], a_{j+1}\ldots a_n)\\ & \overset{*}{\vdash} (\Upsilon\ A[\,]\ \Upsilon_1\ C[\,]\ F'[\,]\ \Upsilon_2\ D[\delta], a_{p+1}\ldots a_n)\\ & \overset{*}{\vdash} (\Upsilon\ A[\,]\ \Upsilon_1\ C[\,]\ F'[\,]\ \Upsilon_2\ E[\,], a_{q+1}\ldots a_n)\\ & \overset{*}{\vdash} (\Upsilon\ A[\,]\ \Upsilon_1\ C[\,]\ F[\,], a_{k+1}\ldots a_n)\\ & \vdash (\Upsilon\ A[\,]\ \Upsilon_1\ G[\,], a_{k+1}\ldots a_n)\end{array}$$

$$\frac{\begin{array}{c}[A,h \mid F',j,\gamma,F,k,- \mid D,p,E,q]\\ [A,h \mid B,i,\gamma,C,j,\gamma \mid -,-,-,-]\end{array}}{[A,h \mid B,i,\gamma,G,k,- \mid D,p,E,q]} \quad \begin{array}{l}C[\circ\circ] \longmapsto C[\,]\ F'[\circ\circ]\\ C[\circ\circ]\ F[\,] \longmapsto G[\circ\circ]\end{array}$$

  – a una derivación de retorno:

$$\begin{array}{rl}(\Upsilon\ A[\delta], a_{h+1}\ldots a_n) & \overset{*}{\vdash} (\Upsilon\ A[\,]\ \Upsilon_1\ B[\delta\gamma], a_{i+1}\ldots a_n)\\ & \overset{*}{\vdash} (\Upsilon\ A[\,]\ \Upsilon_1\ B[\,]\ \Upsilon_2\ D[\delta], a_{p+1}\ldots a_n)\\ & \overset{*}{\vdash} (\Upsilon\ A[\,]\ \Upsilon_1\ C[\,]\ \Upsilon_2\ E[\,], a_{q+1}\ldots a_n)\\ & \overset{*}{\vdash} (\Upsilon\ A[\,]\ \Upsilon_1\ C[\,], a_{j+1}\ldots a_n)\\ & \vdash (\Upsilon\ A[\,]\ \Upsilon_1\ C[\,]\ F'[\,], a_{j+1}\ldots a_n)\\ & \overset{*}{\vdash} (\Upsilon\ A[\,]\ \Upsilon_1\ C[\,]\ F[\,], a_{k+1}\ldots a_n)\\ & \vdash (\Upsilon\ A[\,]\ \Upsilon_1\ G[\,], a_{k+1}\ldots a_n)\end{array}$$

$$\frac{\begin{array}{c}[-,- \mid F',j,-,F,k,- \mid -,-,-,-]\\ [A,h \mid B,i,\gamma,C,j,- \mid D,p,E,q]\end{array}}{[A,h \mid B,i,\gamma,G,k,- \mid D,p,E,q]} \quad \begin{array}{l}C[\circ\circ] \longmapsto C[\,]\ F'[\circ\circ]\\ C[\circ\circ]\ F[\,] \longmapsto G[\circ\circ]\end{array}$$

  – a una derivación de puntos especiales:

$$\begin{array}{rl}(\Upsilon\ B[\,], a_{i+1}\ldots a_n) & \overset{*}{\vdash} (\Upsilon\ C[\,], a_{j+1}\ldots a_n)\\ & \vdash (\Upsilon\ C[\,]\ F'[\,], a_{j+1}\ldots a_n)\\ & \overset{*}{\vdash} (\Upsilon\ C[\,]\ F[\,], a_{k+1}\ldots a_n)\\ & \vdash (\Upsilon\ G[\,], a_{k+1}\ldots a_n)\end{array}$$

$$\frac{\begin{array}{c}[-,- \mid F',j,-,F,k,- \mid -,-,-,-]\\ [-,- \mid B,i,-,C,j,- \mid -,-,-,-]\end{array}}{[-,- \mid B,i,-,G,k,- \mid -,-,-,-]} \quad \begin{array}{l}C[\circ\circ] \longmapsto C[\,]\ F'[\circ\circ]\\ C[\circ\circ]\ F[\,] \longmapsto G[\circ\circ]\end{array}$$

- Derivaciones que son el resultado de aplicar una transición $C[\circ\circ]\ F[\,] \longmapsto G[\circ\circ]$ a una derivación obtenida tras aplicar una transición $C[\circ\circ] \longmapsto C[\,]\ F'[\circ\circ\gamma']$
  - a una derivación de llamada:

$$
\begin{array}{ll}
(\Upsilon\ A[\delta], a_{h+1}\dots a_n) & \overset{*}{\vdash} (\Upsilon\ A[\,]\ \Upsilon_1\ B[\delta\gamma], a_{i+1}\dots a_n) \\
& \overset{*}{\vdash} (\Upsilon\ A[\,]\ \Upsilon_1\ C[\delta\gamma], a_{j+1}\dots a_n) \\
& \vdash (\Upsilon\ A[\,]\ \Upsilon_1\ C[\,]\ F'[\delta\gamma\gamma'], a_{j+1}\dots a_n) \\
& \overset{*}{\vdash} (\Upsilon\ A[\,]\ \Upsilon_1\ C[\,]\ F'[\,]\ \Upsilon_2\ D[\delta\gamma], a_{p+1}\dots a_n) \\
& \overset{*}{\vdash} (\Upsilon\ A[\,]\ \Upsilon_1\ C[\,]\ F'[\,]\ \Upsilon_2\ D[\,]\ \Upsilon_3\ O[\delta], a_{u+1}\dots a_n) \\
& \overset{*}{\vdash} (\Upsilon\ A[\,]\ \Upsilon_1\ C[\,]\ F'[\,]\ \Upsilon_2\ D[\,]\ \Upsilon_3\ P[\,], a_{v+1}\dots a_n) \\
& \overset{*}{\vdash} (\Upsilon\ A[\,]\ \Upsilon_1\ C[\,]\ F'[\,]\ \Upsilon_2\ E[\,], a_{q+1}\dots a_n) \\
& \overset{*}{\vdash} (\Upsilon\ A[\,]\ \Upsilon_1\ C[\,]\ F[\,], a_{k+1}\dots a_n) \\
& \vdash (\Upsilon\ A[\,]\ \Upsilon_1\ G[\,], a_{k+1}\dots a_n)
\end{array}
$$

$$
\begin{array}{cl}
[C,j \mid F',j,\gamma',F,k,- \mid D,p,E,q] & \\
[A,h \mid B,i,\gamma,C,j,\gamma \mid -,-,-,-] & \\
[A,h \mid D,p,\gamma,E,q,- \mid O,u,P,v] & C[\circ\circ] \longmapsto C[\,]\ F'[\circ\circ\gamma'] \\
\hline
[A,h \mid B,i,\gamma,G,k,- \mid O,u,P,v] & C[\circ\circ]\ F[\,] \longmapsto G[\circ\circ]
\end{array}
$$

  - a una derivación de retorno:

$$
\begin{array}{ll}
(\Upsilon\ A[\delta], a_{h+1}\dots a_n) & \overset{*}{\vdash} (\Upsilon\ A[\,]\ \Upsilon_1\ B[\delta\gamma], a_{i+1}\dots a_n) \\
& \overset{*}{\vdash} (\Upsilon\ A[\,]\ \Upsilon_1\ B[\,]\ \Upsilon_2\ D[\delta], a_{p+1}\dots a_n) \\
& \overset{*}{\vdash} (\Upsilon\ A[\,]\ \Upsilon_1\ C[\,]\ \Upsilon_2\ E[\,], a_{q+1}\dots a_n) \\
& \overset{*}{\vdash} (\Upsilon\ A[\,]\ \Upsilon_1\ C[\,], a_{j+1}\dots a_n) \\
& \vdash (\Upsilon\ A[\,]\ \Upsilon_1\ C[\,]\ F'[\gamma'], a_{j+1}\dots a_n) \\
& \overset{*}{\vdash} (\Upsilon\ A[\,]\ \Upsilon_1\ C[\,]\ F'[\,]\ \Upsilon_3\ O[\,], a_{u+1}\dots a_n) \\
& \overset{*}{\vdash} (\Upsilon\ A[\,]\ \Upsilon_1\ C[\,]\ F'[\,]\ \Upsilon_3\ P[\,], a_{v+1}\dots a_n) \\
& \overset{*}{\vdash} (\Upsilon\ A[\,]\ \Upsilon_1\ C[\,]\ F[\,], a_{k+1}\dots a_n) \\
& \vdash (\Upsilon\ A[\,]\ \Upsilon_1\ G[\,], a_{k+1}\dots a_n)
\end{array}
$$

$$
\begin{array}{cl}
[C,j \mid F',j,\gamma',F,k,- \mid O,u,P,v] & \\
[A,h \mid B,i,\gamma,C,j,- \mid D,p,E,q] & \\
[-,- \mid O,u,-,P,v,- \mid -,-,-,-] & C[\circ\circ] \longmapsto C[\,]\ F'[\circ\circ\gamma'] \\
\hline
[A,h \mid B,i,\gamma,G,k,- \mid D,p,E,q] & C[\circ\circ]\ F[\,] \longmapsto G[\circ\circ]
\end{array}
$$

  - a una derivación de puntos especiales:

$$
\begin{array}{ll}
(\Upsilon\ B[\,], a_{i+1}\dots a_n) & \overset{*}{\vdash} (\Upsilon\ C[\,], a_{j+1}\dots a_n) \\
& \vdash (\Upsilon\ C[\,]\ F'[\gamma'], a_{j+1}\dots a_n) \\
& \overset{*}{\vdash} (\Upsilon\ C[\,]\ F'[\,]\ \Upsilon_1\ O[\,], a_{u+1}\dots a_n) \\
& \overset{*}{\vdash} (\Upsilon\ C[\,]\ F'[\,]\ \Upsilon_1\ P[\,], a_{v+1}\dots a_n) \\
& \overset{*}{\vdash} (\Upsilon\ C[\,]\ F[\,], a_{k+1}\dots a_n) \\
& \vdash (\Upsilon\ G[\,], a_{k+1}\dots a_n)
\end{array}
$$

$$
\begin{array}{cl}
[C,j \mid F',j,\gamma',F,k,- \mid O,u,P,v] & \\
[-,- \mid B,i,-,C,j,- \mid -,-,-,-] & \\
[-,- \mid O,u,-,P,v,- \mid -,-,-,-] & C[\circ\circ] \longmapsto C[\,]\ F'[\circ\circ\gamma'] \\
\hline
[-,- \mid B,i,-,G,k,- \mid -,-,-,-] & C[\circ\circ]\ F[\,] \longmapsto G[\circ\circ]
\end{array}
$$

- Derivaciones que son el resultado de aplicar una transición $C[\circ\circ]\ F[\,] \longmapsto G[\circ\circ]$ a una derivación obtenida tras aplicar una transición $C[\circ\circ\gamma] \longmapsto C[\,]\ F'[\circ\circ]$ a una derivación de llamada, con los tres casos siguientes:
  - la derivación de llamada es a su vez derivada de una derivación de llamada:

$$
\begin{array}{rl}
(\Upsilon\ M[\delta], a_{m+1}\dots a_n) & \overset{*}{\vdash} (\Upsilon\ M[\,]\ \Upsilon_1\ N[\delta\gamma'], a_{t+1}\dots a_n) \\
& \overset{*}{\vdash} (\Upsilon\ M[\,]\ \Upsilon_1\ A[\delta\gamma'], a_{h+1}\dots a_n) \\
& \overset{*}{\vdash} (\Upsilon\ M[\,]\ \Upsilon_1\ A[\,]\ \Upsilon_2\ B[\delta\gamma'\gamma], a_{i+1}\dots a_n) \\
& \overset{*}{\vdash} (\Upsilon\ M[\,]\ \Upsilon_1\ A[\,]\ \Upsilon_2\ C[\delta\gamma'\gamma], a_{j+1}\dots a_n) \\
& \vdash (\Upsilon\ M[\,]\ \Upsilon_1\ A[\,]\ \Upsilon_2\ C[\,]\ F'[\delta\gamma'], a_{j+1}\dots a_n) \\
& \overset{*}{\vdash} (\Upsilon\ M[\,]\ \Upsilon_1\ A[\,]\ \Upsilon_2\ C[\,]\ F'[\,]\ \Upsilon_3\ D[\delta], a_{p+1}\dots a_n) \\
& \overset{*}{\vdash} (\Upsilon\ M[\,]\ \Upsilon_1\ A[\,]\ \Upsilon_2\ C[\,]\ F'[\,]\ \Upsilon_3\ E[\,], a_{q+1}\dots a_n) \\
& \overset{*}{\vdash} (\Upsilon\ M[\,]\ \Upsilon_1\ A[\,]\ \Upsilon_2\ C[\,]\ F[\,], a_{k+1}\dots a_n) \\
& \vdash (\Upsilon\ M[\,]\ \Upsilon_1\ A[\,]\ \Upsilon_2\ G[\,], a_{k+1}\dots a_n)
\end{array}
$$

$$
\begin{array}{c}
[M,m \mid F',j,\gamma',F,k,- \mid D,p,E,q] \\
[A,h \mid B,i,\gamma,C,j,\gamma \mid -,-,-,-] \\
[M,m \mid N,t,\gamma',A,h,\gamma' \mid -,-,-,-] \\
\hline
[A,h \mid B,i,\gamma,G,k,- \mid F',j,F,k]
\end{array}
\quad
\begin{array}{l}
C[\circ\circ\gamma] \longmapsto C[\,]\ F'[\circ\circ] \\
C[\circ\circ]\ F[\,] \longmapsto G[\circ\circ]
\end{array}
$$

  - la derivación de llamada es a su vez derivada de una derivación de retorno:

$$
\begin{array}{rl}
(\Upsilon\ M[\delta], a_{m+1}\dots a_n) & \overset{*}{\vdash} (\Upsilon\ M[\,]\ \Upsilon_1\ N[\delta\gamma'], a_{t+1}\dots a_n) \\
& \overset{*}{\vdash} (\Upsilon\ M[\,]\ \Upsilon_1\ N[\,]\ \Upsilon_2\ D[\delta], a_{p+1}\dots a_n) \\
& \overset{*}{\vdash} (\Upsilon\ M[\,]\ \Upsilon_1\ N[\,]\ \Upsilon_2\ E[\,], a_{q+1}\dots a_n) \\
& \overset{*}{\vdash} (\Upsilon\ M[\,]\ \Upsilon_1\ A[\,], a_{h+1}\dots a_n) \\
& \overset{*}{\vdash} (\Upsilon\ M[\,]\ \Upsilon_1\ A[\,]\ \Upsilon_3\ B[\gamma], a_{i+1}\dots a_n) \\
& \overset{*}{\vdash} (\Upsilon\ M[\,]\ \Upsilon_1\ A[\,]\ \Upsilon_3\ C[\gamma], a_{j+1}\dots a_n) \\
& \vdash (\Upsilon\ M[\,]\ \Upsilon_1\ A[\,]\ \Upsilon_3\ C[\gamma]\ F'[\,], a_{j+1}\dots a_n) \\
& \overset{*}{\vdash} (\Upsilon\ M[\,]\ \Upsilon_1\ A[\,]\ \Upsilon_3\ C[\gamma]\ F[\,], a_{k+1}\dots a_n) \\
& \vdash (\Upsilon\ M[\,]\ \Upsilon_1\ A[\,]\ \Upsilon_3\ G[\,], a_{k+1}\dots a_n)
\end{array}
$$

$$
\begin{array}{c}
[-,- \mid F',j,-,F,k,- \mid -,-,-,-] \\
[A,h \mid B,i,\gamma,C,j,\gamma \mid -,-,-,-] \\
[M,m \mid N,t,\gamma',A,h,- \mid D,p,E,q] \\
\hline
[A,h \mid B,i,\gamma,G,k,- \mid F',j,F,k]
\end{array}
\quad
\begin{array}{l}
C[\circ\circ\gamma] \longmapsto C[\,]\ F'[\circ\circ] \\
C[\circ\circ]\ F[\,] \longmapsto G[\circ\circ]
\end{array}
$$

  - la derivación de llamada es a su vez derivada de una derivación de puntos especiales:

$$
\begin{array}{rl}
(N[\delta\gamma'], a_{t+1}\dots a_n) & \overset{*}{\vdash} (A[\,], a_{h+1}\dots a_n) \\
& \overset{*}{\vdash} (A[\,]\ \Upsilon_3\ B[\gamma], a_{i+1}\dots a_n) \\
& \overset{*}{\vdash} (A[\,]\ \Upsilon_3\ C[\gamma], a_{j+1}\dots a_n) \\
& \vdash (A[\,]\ \Upsilon_3\ C[\,]\ F'[\,], a_{j+1}\dots a_n) \\
& \overset{*}{\vdash} (A[\,]\ \Upsilon_3\ C[\,]\ F[\,], a_{k+1}\dots a_n) \\
& \vdash (A[\,]\ \Upsilon_3\ G[\,], a_{k+1}\dots a_n)
\end{array}
$$

$$
\begin{array}{c}
[-,- \mid F',j,-,F,k,- \mid -,-,-,-] \\
[A,h \mid B,i,\gamma,C,j,\gamma \mid -,-,-,-] \\
[-,- \mid N,t,\gamma',A,h,- \mid -,-,-,-] \\
\hline
[A,h \mid B,i,\gamma,G,k,- \mid F',j,F,k]
\end{array}
\quad
\begin{array}{l}
C[\circ\circ\gamma] \longmapsto C[\,]\ F'[\circ\circ] \\
C[\circ\circ]\ F[\,] \longmapsto G[\circ\circ]
\end{array}
$$

□

La complejidad espacial de la técnica de tabulación con respecto a la longitud n de la cadena de entrada es $\mathcal{O}(n^5)$ puesto que cada ítem almacena 5 posiciones de la cadena de entrada. La complejidad temporal en el peor caso es $\mathcal{O}(n^7)$ y viene dada por la regla de combinación de ítems

$$\frac{\begin{array}{l}[C,j \mid F',j,\gamma',F,k,- \mid D,p,E,q] \\ [A,h \mid B,i,\gamma,C,j,\gamma \mid -,-,-,-] \\ [A,h \mid D,p,\gamma,E,q,- \mid O,u,P,v]\end{array}}{[A,h \mid B,i,\gamma,G,k,- \mid O,u,P,v]} \quad \begin{array}{l} C[\circ\circ] \longmapsto C[\,]\ F'[\circ\circ\gamma'] \\ C[\circ\circ]\ F[\,] \longmapsto G[\circ\circ]\end{array}$$

puesto que aunque se combinan ocho posiciones de la cadena de entrada, mediante aplicación parcial sólo es preciso utilizar simultáneamente siete de dichas posiciones. La complejidad temporal puede reducirse utilizando la técnica propuesta en [53, 125], consistente en dividir la regla mencionada en dos reglas de menor complejidad, de tal modo que la primera genere un pseudo-ítem intermedio que proporcione la información relevante para la segunda. En el caso que nos ocupa, se trata de repartir la información proporcionada por el ítem $[C,j \mid F',j,\gamma',F,k,- \mid D,p,E,q]$ entre las dos nuevas reglas

$$\frac{\begin{array}{l}[C,j \mid F',j,\gamma',F,k,- \mid D,p,E,q] \\ [A,h \mid D,p,\gamma,E,q,- \mid O,u,P,v]\end{array}}{[[C,j \mid F',j,\gamma',F,k,- \mid O,u,P,v]]} \quad \begin{array}{l} C[\circ\circ] \longmapsto C[\,]\ F'[\circ\circ\gamma'] \\ C[\circ\circ]\ F[\,] \longmapsto G[\circ\circ]\end{array}$$

$$\frac{\begin{array}{l}[[C,j \mid F',j,\gamma',F,k,- \mid O,u,P,v]] \\ [A,h \mid B,i,\gamma,C,j,\gamma \mid -,-,-,-] \\ [A,h \mid D,p,\gamma,E,q,- \mid O,u,P,v]\end{array}}{[A,h \mid B,i,\gamma,G,k,- \mid O,u,P,v]} \quad \begin{array}{l} C[\circ\circ] \longmapsto C[\,]\ F'[\circ\circ\gamma'] \\ C[\circ\circ]\ F[\,] \longmapsto G[\circ\circ]\end{array}$$

donde $[[C,j \mid F',j,\gamma',F,k,- \mid O,u,P,v]]$ es un pseudo-ítem intermedio que relaciona las dos reglas. La primera regla ignora la posición h, que es posteriormente recuperada del segundo y tercer ítem que intervienen en la segunda regla, que junto con el pseudo-ítem son suficientes para garantizar la existencia del ítem $[C,j \mid F',j,\gamma',F,k,- \mid D,p,E,q]$ por la definición de derivaciones de llamada y retorno. La primera regla presenta una complejidad temporal $\mathcal{O}(n^6)$ (la posición h no interviene) y la segunda presenta también una complejidad $\mathcal{O}(n^6)$ (las posiciones p y q no intervienen) por lo que hemos logrado rebajar la complejidad temporal en el peor caso de la técnica de tabulación a $\mathcal{O}(n^6)$.

9.4 Autómatas lineales de índices fuertemente dirigidos

Un autómata lineal de índices no orientado (*Non-oriented Lineal Indexed Automata*, N-LIA) es aquel capaz de describir estrategias de análisis para gramáticas lineales de índices en las cuales las pilas de índices pueden ser construidas durante la fase descendente, durante la fase ascendente o en ambas fases. En el caso de las gramáticas de adjunción de árboles, tendríamos que dichos autómatas deben permitir la definición de algoritmos de análisis en los cuales las adjunciones se reconocen de forma descendente, ascendente o mixta.

Debido a su carencia de orientación, los N-LIA deben poder utilizar un juego de transiciones que sea el resultado de unir las transiciones permitidas en los R–LIA con aquellas permitidas en los L–LIA:

- $C[\circ\circ\gamma] \overset{a}{\longmapsto} F[\circ\circ\gamma']$

- $C[\circ\circ] \overset{a}{\longmapsto} C[\circ\circ]\ F[\,]$
- $C[\circ\circ\gamma] \overset{a}{\longmapsto} C[\,]\ F[\circ\circ\gamma']$
- $C[\circ\circ\gamma]\ F[\,] \overset{a}{\longmapsto} G[\circ\circ\gamma']$
- $C[\,]\ F[\circ\circ\gamma] \overset{a}{\longmapsto} G[\circ\circ\gamma']$

Sin embargo, la utilización de este juego de transiciones permite definir autómatas que aceptan lenguajes que no pueden ser generados por ninguna gramática lineal de índices, tal y como se muestra en el siguiente ejemplo.

Ejemplo 9.2 El siguiente conjunto de transiciones define un N-LIA que acepta el lenguaje $a_1^n \dots a_{2k}^n \mid k > 0, n \geq 0$. En dichas transiciones, el subíndice i denota números enteros impares mientras que el subíndice j denota números enteros pares.

$$
\begin{array}{lc}
(a) & \$_0[\circ\circ] \longmapsto \$_0[\circ\circ]\ A_1[\,] \\
(b) & A_1[\circ\circ] \overset{a_1}{\longmapsto} A_1[\,]\ A_1[\circ\circ\gamma] \\
(c) & A_i[\circ\circ] \longmapsto A_{i+1}[\circ\circ] \\
(d) & A_j[\circ\circ\gamma] \overset{a_j}{\longmapsto} A_j[\,]\ A_j[\circ\circ] \\
(e) & A_j[\circ\circ] \longmapsto A_{j+1}[\circ\circ] \\
(f) & A_j[\circ\circ]\ A_{j+1}[\,] \overset{a_{j+1}}{\longmapsto} A_{j+1}[\circ\circ\gamma] \\
(g) & A_j[\,]\ A_{j+1}[\circ\circ] \overset{a_{j+1}}{\longmapsto} A_{j+1}[\circ\circ\gamma] \\
(h) & A_1[\,]\ A_{2k}[\circ\circ\gamma] \overset{a_{2k}}{\longmapsto} A_{2k}[\circ\circ]
\end{array}
$$

En la tabla 9.17 se muestra la derivación correspondiente a la cadena $a_1^2 \dots a_{2k}^2$. ¶

Con el fin de limitar los lenguajes aceptados a los lenguajes de adjunción de árboles, debemos establecer dos modos de funcionamiento en este tipo de autómatas:

- Un modo de escritura **w** en el cual no se permite extraer elementos de la pila del autómata.
- Un modo de borrado **e** en el cual no se permite realizar apilamientos en la pila del autómata.

Por convención, definimos una relación de orden entre modos, de tal modo que se cumple que $\mathbf{w} < \mathbf{e}$.

Adicionalmente, restringiremos las operaciones de las pilas de índices estableciendo que en los modos **w** y **e** se aplicarán las mismas operaciones de apilamiento y extracción de elementos de las pilas de índices, pero en orden inverso. Para ello, cada transición PUSH realizada en modo **w** escribirá una marca de acción que indique la operación realizada sobre la pila de índices:

$\models^{\mathbf{w}}$ para indicar la creación de una nueva pila de índices a partir de una configuración en modo de escritura.

$\models^{\mathbf{e}}$ para indicar la creación de una nueva pila de índices a partir de una configuración en modo de borrado.

$\nearrow$ para indicar el apilamiento de un índice.

	$\$_0[\,]$	$a_1a_1a_2a_2a_3a_3a_4a_4 \ldots a_{2k}a_{2k}$
(a)	$\$_0[\,]\ A_1[\,]$	$a_1a_1a_2a_2a_3a_3a_4a_4 \ldots a_{2k}a_{2k}$
(b)	$\$_0[\,]\ A_1[\,]\ A_1[\gamma]$	$a_1a_2a_2a_3a_3a_4a_4 \ldots a_{2k}a_{2k}$
(b)	$\$_0[\,]\ A_1[\,]\ A_1[\,]\ A_1[\gamma\gamma]$	$a_2a_2a_3a_3a_4a_4 \ldots a_{2k}a_{2k}$
(c)	$\$_0[\,]\ A_1[\,]\ A_1[\,]\ A_2[\gamma\gamma]$	$a_2a_2a_3a_3a_4a_4 \ldots a_{2k}a_{2k}$
(d)	$\$_0[\,]\ A_1[\,]\ A_1[\,]\ A_2[\,]\ A_2[\gamma]$	$a_2a_3a_3a_4a_4 \ldots a_{2k}a_{2k}$
(d)	$\$_0[\,]\ A_1[\,]\ A_1[\,]\ A_2[\,]\ A_2[\,]\ A_2[\,]$	$a_3a_3a_4a_4 \ldots a_{2k}a_{2k}$
(e)	$\$_0[\,]\ A_1[\,]\ A_1[\,]\ A_2[\,]\ A_2[\,]\ A_3[\,]$	$a_3a_3a_4a_4 \ldots a_{2k}a_{2k}$
(f)	$\$_0[\,]\ A_1[\,]\ A_1[\,]\ A_2[\,]\ A_3[\gamma]$	$a_3a_4a_4 \ldots a_{2k}a_{2k}$
(g)	$\$_0[\,]\ A_1[\,]\ A_1[\,]\ A_3[\gamma\gamma]$	$a_4a_4 \ldots a_{2k}a_{2k}$
(c)	$\$_0[\,]\ A_1[\,]\ A_1[\,]\ A_4[\gamma\gamma]$	$a_4a_4 \ldots a_{2k}a_{2k}$
(d)	$\$_0[\,]\ A_1[\,]\ A_1[\,]\ A_4[\,]\ A_4[\gamma]$	$a_4 \ldots a_{2k}a_{2k}$
(d)	$\$_0[\,]\ A_1[\,]\ A_1[\,]\ A_4[\,]\ A_4[\,]\ A_4[\,]$	$\ldots a_{2k}a_{2k}$
	$\vdots$	
(g)	$\$_0[\,]\ A_1[\,]\ A_1[\,]\ A_{2k-1}[\gamma\gamma]$	$a_{2k}a_{2k}$
(c)	$\$_0[\,]\ A_1[\,]\ A_1[\,]\ A_{2k}[\gamma\gamma]$	$a_{2k}a_{2k}$
(h)	$\$_0[\,]\ A_1[\,]\ A_{2k}[\gamma]$	a_{2k}
(h)	$\$_0[\,]\ A_{2k}[\,]$	

Tabla 9.17: derivación de la cadena $a_1^2 \ldots a_{2k}^2$ en N-LIA

$\rightarrow$ para indicar que no se ha realizado modificación alguna sobre una pila de índices.

$\searrow$ para indicar que se ha extraído el índice de la cima.

Estas marcas de acción dirigirán, durante el modo **e**, las operaciones que las transiciones POP pueden aplicar sobre las pilas de índices. Puesto que las transiciones PUSH escriben las marcas en el modo **w**, denominaremos $\otimes$**WRITE** a dichas transiciones. Las transiciones POP son las encargadas de borrar las marcas en el modo **e**, por lo que recibirán el nombre de transiciones $\otimes$**ERASE**, donde $\otimes \in \{\models^{\mathbf{w}}, \models^{\mathbf{e}}, \nearrow, \rightarrow, \searrow\}$.

Una configuración (m, Υ, w) del autómata vendrá determinada por el modo m en que se encuentre, el contenido Υ de la pila y la parte w de la cadena de entrada que resta por leer.

Observamos que en la nueva clase de autómatas que acabamos de formular, los movimientos que se pueden realizar en un momento dado están restringidos en gran medida por movimientos realizados en algún momento anterior. Es por ello que recibirán el nombre de *autómatas lineales de índices fuertemente dirigidos* (*Strongly-Driven Linear Indexed Automata*, SD–LIA). Formalmente, definiremos un autómata lineal de índices fuertemente dirigido como una tupla $(V_T, V_S, \$_0, \$_f, V_I, \mathcal{D}, \mathcal{T})$, donde:

- V_T es un conjunto finito de símbolos terminales.
- V_S es un conjunto finito de símbolos de pila.
- $\$_0 \in V_S$ es el símbolo inicial de la pila.

- $\$_f \in V_S$ es el símbolo final de pila.
- V_I es un conjunto finito de índices.
- $\mathcal{D} = \{\models^{\mathbf{w}}, \models^{\mathbf{e}} \nearrow, \rightarrow, \searrow\}$ es el conjunto de marcas de acción.
- $\mathcal{T}$ es un conjunto finito de transiciones de los siguientes tipos:

SWAP1 : Transiciones de la forma $C[\circ\circ] \quad m \overset{a}{\longmapsto} m \quad F[\circ\circ]$. El resultado de aplicar una transición de este tipo a una configuración $(m, \Upsilon C, aw)$ es una configuración $(m, \Upsilon F, w)$.

SWAP2 : Transiciones de la forma $C[\,] \;\; \mathbf{w} \overset{a}{\longmapsto} \mathbf{e} \;\; F[\,]$. El resultado de aplicar una transición de este tipo a una configuración $(\mathbf{w}, \Upsilon C[\,], aw)$ es una configuración $(\mathbf{e}, \Upsilon F[\,], w)$. Observamos que este tipo de transiciones permite cambiar de modo de escritura a modo de borrado, pero no a la inversa.

$\models$**WRITE** : Transiciones de la forma $C[\circ\circ] \; m \longmapsto \mathbf{w} \; C[\circ\circ] \models^m F[\,]$ que al ser aplicadas a una configuración $(m, \Upsilon C[\alpha], w)$ producen una configuración $(\mathbf{w}, \Upsilon C[\alpha] \models^m F[\,], w)$. Este tipo de transiciones permite iniciar un nuevo modo de escritura.

$\rightarrow$**WRITE** : Transiciones de la forma $C[\circ\circ] \; \mathbf{w} \longmapsto \mathbf{w} \; C[\,] \rightarrow F[\circ\circ]$ que al ser aplicadas a una configuración $(\mathbf{w}, \Upsilon C[\alpha], w)$ derivan una configuración $(\mathbf{w}, \Upsilon C[\,] \rightarrow F[\alpha])$.

$\nearrow$**WRITE** : Transiciones de la forma $C[\circ\circ] \; \mathbf{w} \longmapsto \mathbf{w} \; C[\,] \nearrow F[\circ\circ\gamma']$ que al ser aplicadas a una configuración $(\mathbf{w}, \Upsilon C[\alpha], w)$ producen una configuración $(\mathbf{w}, \Upsilon C[\,] \nearrow F[\alpha\gamma'])$.

$\searrow$**WRITE** : Transiciones de la forma $C[\circ\circ\gamma] \; \mathbf{w} \longmapsto \mathbf{w} \; C[\,] \searrow F[\circ\circ]$ que al ser aplicadas a una configuración $(\mathbf{w}, \Upsilon C[\alpha\gamma], w)$ producen una configuración $(\mathbf{w}, \Upsilon C[\,] \searrow F[\alpha])$.

$\models$**ERASE** : Transiciones de la forma $C[\circ\circ] \models^m F[\,] \; \mathbf{e} \longmapsto m \; G[\circ\circ]$ que al ser aplicadas a una configuración $(\mathbf{e}, \Upsilon C[\alpha] \models^m F[\,])$ derivan una configuración $(m, \Upsilon G[\alpha], w)$. Este tipo de transiciones permite volver al modo en que se encontraba el autómata antes de aplicar una transición $\models$**WRITE**.

$\rightarrow$**ERASE** : Transiciones de la forma $C[\,] \rightarrow F[\circ\circ] \; \mathbf{e} \longmapsto \mathbf{e} \; G[\circ\circ]$ que al ser aplicadas a una configuración $(\mathbf{e}, \Upsilon C[\,] \rightarrow F[\alpha], w)$ producen una configuración $(\mathbf{e}, \Upsilon G[\alpha], w)$.

$\nearrow$**ERASE** : Transiciones de la forma $C[\,] \nearrow F[\circ\circ\eta'] \; \mathbf{e} \longmapsto \mathbf{e} \; G[\circ\circ]$ que al ser aplicadas a una transición $(\mathbf{e}, \Upsilon C[\,] \nearrow F[\beta\eta'])$ producen una configuración de la forma $(\mathbf{e}, \Upsilon G[\beta], w)$.

$\searrow$**ERASE** : Transiciones de la forma $C[\,] \searrow F[\circ\circ] \; \mathbf{e} \longmapsto \mathbf{e} \; G[\circ\circ\eta]$ que al ser aplicadas a una configuración $(\mathbf{w}, \Upsilon C[\,] \searrow F[\beta], w)$ derivan una configuración $(\mathbf{e}, \Upsilon G[\beta\eta], w)$.

Una *configuración* de un autómata lineal de índices fuertemente dirigido es un triple (m, Υ, w), donde $m \in \{\mathbf{w}, \mathbf{e}\}$, $\Upsilon \in (\mathcal{D} \; V_S[V_I^*])^*$ y $w \in V_T^*$, tal que m es el modo en que se encuentra el autómata, Υ es el contenido de la pila del autómata y w representa la parte de la cadena de entrada que resta por leer. Una configuración (m, Υ_1, aw) deriva una configuración (m', Υ_2, w), denotado mediante $(m, \Upsilon_1, aw) \vdash (m', \Upsilon_2, w)$ si y sólo si existe una transición que aplicada en modo m transforma la pila Υ_1 en la pila Υ_2 leyendo $a \in V_T \cup \{\epsilon\}$ de la cadena de entrada mientras el autómata pasa a modo m'. En caso de ser necesario identificar una derivación d concreta, utilizaremos la notación $\vdash_d$. Denotamos por $\overset{*}{\vdash}$ el cierre reflexivo y transitivo de $\vdash$. Decimos que una cadena de entrada w es aceptada por un autómata lineal de índices fuertemente dirigido si

$$(\mathbf{w}, \models^{\mathbf{w}} \$_0[\,], w) \overset{*}{\vdash} (\mathbf{e}, \models^{\mathbf{w}} \$_0[\,] \; \models^{\mathbf{w}} \$_f[\,], \epsilon)$$

El lenguaje aceptado por un SD–LIA es el conjunto

$$\{w \in V_T^* \mid (\mathbf{w}, \models^{\mathbf{w}} \$_0[\,], w) \overset{*}{\vdash} (\mathbf{e}, \models^{\mathbf{w}} \$_0[\,] \; \models^{\mathbf{w}} \$_f[\,], \epsilon)\}$$

9.4.1 Esquemas de compilación de gramáticas lineales de índices

Definiremos un esquema de compilación genérico de LIG en SD–LIA en el cual la información predicha en la fase de llamada y la información propagada en la fase de retorno está parametrizada en función de:

- $\overrightarrow{A}$, que se refiere a la predicción realizada sobre el no-terminal A durante la fase descendente de la estrategia de análisis.
- $\overrightarrow{\gamma}$, que se refiere a la predicción realizada sobre el índice γ.
- $\overleftarrow{A}$, que se refiere a la propagación de información respecto al no-terminal A durante la fase ascendente de la estrategia de análisis.
- $\overleftarrow{\gamma}$, que se refiere a la propagación de información respecto al índice γ.

Esquema de compilación 9.11 El esquema de compilación genérico de una gramática lineal de índices en un autómata lineal de índices fuertemente dirigido queda definido por el conjunto de reglas mostrado en la tabla 9.18 y por los elementos inicial $\$_0[\,]$ y final $\overleftarrow{S}[\,]$. §

El esquema de compilación genérico se puede convertir en esquemas de compilación que incorporan estrategias específicas. En la tabla 9.19 se muestran los valores que toman los diferentes parámetros para las estrategias de análisis de LIG más comunes. En dicha tabla, $\Box$ indica un símbolo de pila especial que no aparece inicialmente en V_S y $\Diamond$ indica un índice especial que no aparece inicialmente en V_I. A este respecto, debemos señalar una diferencia significativa entre SD–LIA y R–LIA, pues mientras los esquemas de compilación de LIG en R–LIA predecían una pila vacía en la fase de llamada, los esquemas de compilación $*$-ascendentes de LIG en SD–LIA construirán durante la fase de llamada pilas de índices compuestas por elementos $\Diamond$ con lo cual, aunque no se predice el contenido de la pila, se indica la altura que debería tener la pila de haber realizado la predicción. Análogamente, los esquemas de compilación $*$-descendentes de LIG en SD–LIA propagan pilas de índices constituidas por una serie de $\Diamond$, en lugar de una pila de índices vacía, tal y como sucedía en el caso de los esquemas de compilación de LIG en L–LIA.

9.4.2 Esquemas de compilación de gramáticas de adjunción de árboles

Definiremos en primer lugar un esquema genérico de compilación de TAG en SD–LIA, en el cual el flujo de información está parametrizado en función de

- $\overrightarrow{N^{\gamma}_{r,s}}$, la información predicha acerca del nodo $N^{\gamma}_{r,s}$.
- $\overleftarrow{N^{\gamma}_{r,s}}$, la información propagada acerca del nodo $N^{\gamma}_{r,s}$.

y en el que las pilas de índices almacenan la pila de adjunciones pendientes en un determinado nodo de un árbol elemental.

Esquema de compilación 9.12 El esquema de compilación genérico de una gramática de adjunción de árboles en un autómata lineal de índices fuertemente dirigido queda definido por el conjunto de reglas mostrado en la tabla 9.20 y los elementos inicial $\$_0[\,]$ y final $\overleftarrow{\top^{\alpha}}[\,]$, con $\alpha \in \boldsymbol{I}$.

[INIT]	$\$_0[\circ\circ] \;\; \mathbf{w} \longmapsto \mathbf{w} \;\; \$_0[\circ\circ] \models^{\mathbf{w}} \nabla_{0,0}[\,]$	
[CALL]	$\nabla_{r,s}[\circ\circ] \;\; m \longmapsto \mathbf{w} \;\; \nabla_{r,s}[\circ\circ] \models^{m} \overrightarrow{A_{r,s+1}}[\,]$	$A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\,]\Upsilon_2$
[SCALL-1]	$\nabla_{r,s}[\circ\circ] \;\; \mathbf{w} \longmapsto \mathbf{w} \;\; \nabla_{r,s}[\,] \to \overrightarrow{A_{r,s+1}}[\circ\circ]$	$A_{r,0}[\circ\circ] \to \Upsilon_1 A_{r,s+1}[\circ\circ]\Upsilon_2$
[SCALL-2]	$\nabla_{r,s}[\circ\circ] \;\; \mathbf{w} \longmapsto \mathbf{w} \;\; \nabla_{r,s}[\,] \nearrow \overrightarrow{A_{r,s+1}}[\circ\circ\overrightarrow{\gamma'}]$	$A_{r,0}[\circ\circ] \to \Upsilon_1 A_{r,s+1}[\circ\circ\gamma']\Upsilon_2$
[SCALL-3]	$\nabla_{r,s}[\circ\circ\overrightarrow{\gamma}] \;\; \mathbf{w} \longmapsto \mathbf{w} \;\; \nabla_{r,s}[\,] \searrow \overrightarrow{A_{r,s+1}}[\circ\circ]$	$A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\circ\circ]\Upsilon_2$
[SEL]	$\overrightarrow{A_{r,0}}[\circ\circ] \;\; \mathbf{w} \longmapsto \mathbf{w} \;\; \nabla_{r,0}[\circ\circ]$	$r \neq 0$
[PUB]	$\nabla_{r,n_r}[\circ\circ] \;\; \mathbf{e} \longmapsto \mathbf{e} \;\; \overleftarrow{A_{r,0}}[\circ\circ]$	
[RET]	$\nabla_{r,s}[\circ\circ] \models^{m} \overleftarrow{A_{r,s+1}}[\,] \;\; \mathbf{e} \longmapsto m \;\; \nabla_{r,s+1}[\circ\circ]$	$A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\,]\Upsilon_2$
[SRET-1]	$\nabla_{r,s}[\,] \to \overleftarrow{A_{r,s+1}}[\circ\circ] \;\; \mathbf{e} \longmapsto \mathbf{e} \;\; \nabla_{r,s+1}[\circ\circ]$	$A_{r,0}[\circ\circ] \to \Upsilon_1 A_{r,s+1}[\circ\circ]\Upsilon_2$
[SRET-2]	$\nabla_{r,s}[\,] \nearrow \overleftarrow{A_{r,s+1}}[\circ\circ\overleftarrow{\gamma'}] \;\; \mathbf{e} \longmapsto \mathbf{e} \;\; \nabla_{r,s+1}[\circ\circ]$	$A_{r,0}[\circ\circ] \to \Upsilon_1 A_{r,s+1}[\circ\circ\gamma']\Upsilon_2$
[SRET-3]	$\nabla_{r,s}[\,] \searrow \overleftarrow{A_{r,s+1}}[\circ\circ] \;\; \mathbf{e} \longmapsto \mathbf{e} \;\; \nabla_{r,s+1}[\circ\circ\overleftarrow{\gamma}]$	$A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\circ\circ]\Upsilon_2$
[SCAN]	$\overrightarrow{A_{r,0}}[\,] \;\; \mathbf{w} \overset{a}{\longmapsto} \mathbf{e} \;\; \overleftarrow{A_{r,0}}[\,]$	$A_{r,0}[\,] \to a$

Tabla 9.18: Reglas del esquema de compilación genérico de LIG en SD–LIA

estrategia-CF	**estrategia-índices**	$\overrightarrow{A_{r,s+1}}$	$\overrightarrow{\gamma}$	$\overleftarrow{A_{r,s+1}}$	$\overleftarrow{\gamma}$
Ascendente	ascendente	$\Box$	$\Diamond$	$A_{r,s+1}$	γ
Earley		$\overline{A_{r,s+1}}$	$\Diamond$	$\overline{\overline{A_{r,s+1}}}$	γ
Descendente		$A_{r,s+1}$	$\Diamond$	$\Box$	γ
Ascendente	Earley	$\Box$	γ	$A_{r,s+1}$	γ
Earley		$\overline{A_{r,s+1}}$	γ	$\overline{\overline{A_{r,s+1}}}$	γ
Descendente		$A_{r,s+1}$	γ	$\Box$	γ
Ascendente	descendente	$\Box$	γ	$A_{r,s+1}$	$\Diamond$
Earley		$\overline{A_{r,s+1}}$	γ	$\overline{\overline{A_{r,s+1}}}$	$\Diamond$
Descendente		$A_{r,s+1}$	γ	$\Box$	$\Diamond$

Tabla 9.19: Parámetros del esquema de compilación genérico de LIG en SD–LIA

En este esquema de análisis sintáctico el cambio de modo de escritura a modo de borrado no se produce tan solo en las transiciones producidas por la regla de compilación **[SCAN]** sino también en las transiciones producidas por la regla de compilación **[PUB2]**. Ello se debe a que en las producciones de los árboles iniciales no son aplicables las reglas **[SCALL]** y **[SRET]**, por lo cual todo cambio de modo realizado por una regla **[SCAN]** es eliminado por una regla **[RET]**. §

El esquema de compilación genérico puede convertirse en esquemas de compilación para diferentes estrategias de análisis según los valores que tomen los parámetros, tal y como se indica en la tabla 9.21. Una diferencia importante con respecto a los esquemas de compilación de TAG en R–LIA es que mientras en estos últimos se predecían pilas de adjunciones vacías en la fase descendente de la estrategia, en los esquemas de compilación $*$-ascendentes de TAG en SD–LIA se predecirán pilas de $\diamond$, cada uno de los cuales representa la realización de una adjunción en un nodo desconocido. Análogamente, en los esquemas de compilación $*$-descendentes, se propagarán pilas de $\diamond$, mientras que en los esquemas de compilación de TAG en L–LIA se propagaban pilas de índices vacías.

9.4.3 SD–LIA y los lenguajes de adjunción de árboles

En esta sección trataremos de demostrar la equivalencia entre los autómatas lineales de índices fuertemente dirigidos y los lenguajes de adjunción de árboles. Para ello enunciaremos y demostraremos los dos teoremas siguientes.

Teorema 9.3 *Los lenguajes adjunción de árboles son un subconjunto de los lenguajes aceptados por la clase de los autómatas lineales de índices fuertemente dirigidos.*

Demostración:

Por el esquema de compilación de TAG en SD–LIA presentado anteriormente, a partir de cualquier gramática de adjunción de árboles es posible construir un SD–LIA que acepta el lenguaje reconocido por dicha gramática. Análogamente, por el esquema de compilación de LIG en SD–LIA, a partir de cualquier gramática lineal de índices es posible construir un SD–LIA que acepta el lenguaje reconocido por dicha gramática. □

Teorema 9.4 *La clase de los lenguajes aceptados por los SD–LIA es un subconjunto de los lenguajes de adjunción de árboles.*

Demostración:

Mostraremos que para todo SD–LIA existe una gramática lineal de índices tal que el lenguaje reconocido por la gramática coincide con el lenguaje aceptado por el autómata.

Sea $\mathcal{A} = (V_T, V_S, \$_0, \$_f, V_I, \mathcal{D}, \mathcal{T})$ un autómata lineal de índices fuertemente dirigido. Construiremos una gramática lineal de índices $\mathcal{L} = (V_T, V_N, V_I', S, P)$. El conjunto V_N de no-terminales estará formado por pares $\langle E, B\rangle$ tal que $A, B \in V_S^{\mathcal{D}}$, donde $V_S^{\mathcal{D}} = \{E^m \mid E \in V_S,\ m \in \mathcal{D}\}$. El conjunto V_I' estará formado por pares $\langle\gamma,\eta\rangle$ tal que $\gamma, \eta \in V_I$. Para que $\mathcal{L}$ reconozca el lenguaje aceptado por $\mathcal{A}$ el conjunto de producciones en P ha de construirse a partir de las transiciones en $\mathcal{T}$ de la siguiente manera:

- Para toda transición $C[\circ\circ]\ m \overset{a}{\longmapsto} m\ F[\circ\circ]$ y para todo $E^{m'} \in V_S^{\mathcal{D}}$ tal que $m' \leq m$ creamos una producción

$$\langle E^{m'}, F^m\rangle[\circ\circ] \rightarrow \langle E^{m'}, C^m\rangle[\circ\circ]\ a$$

[INIT]	$\$_0[\circ\circ]\ \mathbf{w}\longmapsto\mathbf{w}\ \$_0[\circ\circ]\ \models^{\mathbf{w}} \nabla^{\alpha}_{0,0}[\,]$	$\alpha \in \boldsymbol{I}$
[CALL]	$\nabla^{\gamma}_{r,s}[\circ\circ]\ m\longmapsto\mathbf{w}\ \nabla^{\gamma}_{r,s}[\circ\circ]\ \models^{m} \overrightarrow{N^{\gamma}_{r,s+1}}[\,]$	$N^{\gamma}_{r,s+1} \not\in \text{espina}(\gamma),\ \mathbf{nil} \in \text{adj}(N_{r,s+1})$
[SCALL]	$\nabla^{\gamma}_{r,s}[\circ\circ]\ \mathbf{w}\longmapsto\mathbf{w}\ \nabla^{\gamma}_{r,s}[\,] \rightarrow \overrightarrow{N^{\gamma}_{r,s+1}}[\circ\circ]$	$N^{\gamma}_{r,s+1} \in \text{espina}(\gamma),\ \mathbf{nil} \in \text{adj}(N_{r,s+1})$
[SEL]	$\overrightarrow{N^{\gamma}_{r,0}}[\circ\circ]\ \mathbf{w}\longmapsto\mathbf{w}\ \nabla^{\gamma}_{r,0}[\circ\circ]$	$r \neq 0$
[PUB1]	$\nabla^{\gamma}_{r,n_r}[\circ\circ]\ \mathbf{e}\longmapsto\mathbf{e}\ \overleftarrow{N^{\gamma}_{r,0}}[\circ\circ]$	
[PUB2]	$\nabla^{\gamma}_{r,n_r}[\,]\ \mathbf{w}\longmapsto\mathbf{e}\ \overleftarrow{N^{\gamma}_{r,0}}[\,]$	
[RET]	$\nabla^{\gamma}_{r,s}[\circ\circ]\ \models^{m} \overleftarrow{N^{\gamma}_{r,s+1}}[\,]\ \mathbf{e}\longmapsto m\ \nabla^{\gamma}_{r,s+1}[\circ\circ]$	$N^{\gamma}_{r,s+1} \not\in \text{espina}(\gamma),\ \mathbf{nil} \in \text{adj}(N_{r,s+1})$
[SRET]	$\nabla^{\gamma}_{r,s}[\,] \rightarrow \overleftarrow{N^{\gamma}_{r,s+1}}[\circ\circ]\ \mathbf{e}\longmapsto\mathbf{e}\ \nabla^{\gamma}_{r,s+1}[\circ\circ]$	$N^{\gamma}_{r,s+1} \in \text{espina}(\gamma),\ \mathbf{nil} \in \text{adj}(N_{r,s+1})$
[SCAN]	$\overrightarrow{N^{\gamma}_{r,0}}[\,]\ \mathbf{w}\stackrel{a}{\longmapsto}\mathbf{e}\ \overleftarrow{N^{\gamma}_{r,0}}[\,]$	$N^{\gamma}_{r,0}[\,] \rightarrow a$
[ACALL]	$\nabla^{\gamma}_{r,s}[\circ\circ]\ \mathbf{w}\longmapsto\mathbf{w}\ \nabla^{\gamma}_{r,s}[\,] \nearrow \overrightarrow{\top^{\beta}}[\circ\circ\overrightarrow{N^{\gamma}_{r,s+1}}]$	$\beta \in \text{adj}(N^{\gamma}_{r,s+1})$
[ARET]	$\nabla^{\gamma}_{r,s}[\,] \nearrow \overleftarrow{\top^{\beta}}[\circ\circ\overleftarrow{N^{\gamma}_{r,s+1}}]\ \mathbf{e}\longmapsto\mathbf{e}\ \nabla^{\gamma}_{r,s+1}[\circ\circ]$	$\beta \in \text{adj}(N^{\gamma}_{r,s+1})$
[FCALL]	$\nabla^{\beta}_{f,0}[\circ\circ\overrightarrow{N^{\gamma}_{r,s+1}}]\ \mathbf{w}\longmapsto\mathbf{w}\ \nabla^{\beta}_{f,0}[\,] \searrow \overrightarrow{N^{\gamma}_{r,s+1}}[\circ\circ]$	$N^{\beta}_{f,0} = \mathbf{F}^{\beta},\ \beta \in \text{adj}(N^{\gamma}_{r,s+1})$
[FRET]	$\nabla^{\beta}_{f,0}[\,] \searrow \overleftarrow{N^{\gamma}_{r,s+1}}[\circ\circ]\ \mathbf{e}\longmapsto\mathbf{e}\ \nabla^{\beta}_{f,1}[\circ\circ\overleftarrow{N^{\gamma}_{r,s+1}}]$	$N^{\beta}_{f,0} = \mathbf{F}^{\beta},\ \beta \in \text{adj}(N^{\gamma}_{r,s+1})$

Tabla 9.20: Reglas del esquema de compilación genérico de TAG en SD–LIA

Estrategia-CF	**Estrategia-adjunción**	$\overrightarrow{N^{\gamma}_{r,s+1}}$	$\circ\circ\overrightarrow{N^{\gamma}_{r,s+1}}$	$\overleftarrow{N^{\gamma}_{r,s+1}}$	$\circ\circ\overleftarrow{N^{\gamma}_{r,s+1}}$
Ascendente	ascendente	$\Box$	$\circ\circ\Diamond$	$N^{\gamma}_{r,s+1}$	$\circ\circ N^{\gamma}_{r,s+1}$
Earley		$\overline{N^{\gamma}_{r,s+1}}$	$\circ\circ\Diamond$	$\overline{\overline{N^{\gamma}_{r,s+1}}}$	$\circ\circ N^{\gamma}_{r,s+1}$
Descendente		$N^{\gamma}_{r,s+1}$	$\circ\circ\Diamond$	$\Box$	$\circ\circ N^{\gamma}_{r,s+1}$
Ascendente	Earley	$\Box$	$\circ\circ N^{\gamma}_{r,s+1}$	$N^{\gamma}_{r,s+1}$	$\circ\circ N^{\gamma}_{r,s+1}$
Earley		$\overline{N^{\gamma}_{r,s+1}}$	$\circ\circ N^{\gamma}_{r,s+1}$	$\overline{\overline{N^{\gamma}_{r,s+1}}}$	$\circ\circ N^{\gamma}_{r,s+1}$
Descendente		$N^{\gamma}_{r,s+1}$	$\circ\circ N^{\gamma}_{r,s+1}$	$\Box$	$\circ\circ N^{\gamma}_{r,s+1}$
Ascendente	descendente	$\Box$	$\circ\circ N^{\gamma}_{r,s+1}$	$N^{\gamma}_{r,s+1}$	$\circ\circ\Diamond$
Earley		$\overline{N^{\gamma}_{r,s+1}}$	$\circ\circ N^{\gamma}_{r,s+1}$	$\overline{\overline{N^{\gamma}_{r,s+1}}}$	$\circ\circ\Diamond$
Descendente		$N^{\gamma}_{r,s+1}$	$\circ\circ N^{\gamma}_{r,s+1}$	$\Box$	$\circ\circ\Diamond$

Tabla 9.21: Parámetros del esquema de compilación genérico de TAG en SD–LIA

- Para toda transición $C[\,]\ \mathbf{w}\overset{a}{\longmapsto}\mathbf{e}\ F[\,]$ y para todo $E^{\mathbf{w}} \in V_S^{\mathcal{D}}$ creamos la producción

$$\langle E^{\mathbf{w}}, F^{\mathbf{e}}\rangle[\,] \to \langle E^{\mathbf{w}}, C^{\mathbf{w}}\rangle[\,]\ a$$

- Para todo par de transiciones $C[\circ\circ] \models^m F[\,]\ \mathbf{e}\longmapsto m\ G[\circ\circ]$ y $C[\circ\circ]\ m\longmapsto\mathbf{w}\ C[\circ\circ] \models^m F'[\,]$, y para todo $E^{m''} \in V_S^{\mathcal{D}}$ tal que $m'' \leq m$ creamos una producción

$$\langle E^{m''}, G^m\rangle[\circ\circ] \to \langle E^{m''}, C^m\rangle[\circ\circ]\ \ \langle F'^{\mathbf{w}}, F^{\mathbf{e}}\rangle[\,]$$

- Para todo par de transiciones $C[\,] \to F[\circ\circ]\ \mathbf{e}\longmapsto\mathbf{e}\ G[\circ\circ]$ y $C[\circ\circ]\ \mathbf{w}\longmapsto\mathbf{w}\ C[\,] \to F'[\circ\circ]$, y para todo $E^{\mathbf{w}} \in V_S^{\mathcal{D}}$ creamos una producción

$$\langle E^{\mathbf{w}}, G^{\mathbf{e}}\rangle[\circ\circ] \to \langle E^{\mathbf{w}}, C^{\mathbf{w}}\rangle[\,]\ \ \langle F'^{\mathbf{w}}, F^{\mathbf{e}}\rangle[\circ\circ]$$

- Para todo par de transiciones $C[\,] \nearrow F[\circ\circ\eta']\ \mathbf{e}\longmapsto\mathbf{e}\ G[\circ\circ]$ y $C[\circ\circ]\ \mathbf{w}\longmapsto\mathbf{w}\ C[\,] \nearrow F'[\circ\circ\gamma']$, y para todo $E^{\mathbf{w}} \in V_S^{\mathcal{D}}$ creamos una producción

$$\langle E^{\mathbf{w}}, G^{\mathbf{e}}\rangle[\circ\circ] \to \langle E^{\mathbf{w}}, C^{\mathbf{w}}\rangle[\,]\ \ \langle F'^{\mathbf{w}}, F^{\mathbf{e}}\rangle[\circ\circ\langle\gamma', \eta'\rangle]$$

- Para todo par de transiciones $C[\,] \searrow F[\circ\circ]\ \mathbf{e}\longmapsto\mathbf{e}\ G[\circ\circ\eta]$ y $C[\circ\circ\gamma]\ \mathbf{w}\longmapsto\mathbf{w}\ C[\,] \searrow F'[\circ\circ]$, y para todo $E^{\mathbf{w}} \in V_S^{\mathcal{D}}$ creamos una producción

$$\langle E^{\mathbf{w}}, G^{\mathbf{e}}\rangle[\circ\circ\langle\gamma, \eta\rangle] \to \langle E^{\mathbf{w}}, C^{\mathbf{w}}\rangle[\,]\ \ \langle F'^{\mathbf{w}}, F^{\mathbf{e}}\rangle[\circ\circ]$$

- Para todo $E^m \in V_S^{\mathcal{D}}$ creamos una producción

$$\langle E^m, E^m\rangle[\,] \to \epsilon$$

- Para toda transición $\$_0[\circ\circ]\ \mathbf{w}\longmapsto\mathbf{w}\ \$_0[\circ\circ] \models^{\mathbf{w}} F[\,]$, donde $F \in V_S - \{\$_0\}$, creamos una producción

$$\langle \$_0^{\mathbf{w}}, \$_0^{\mathbf{w}}\rangle[\circ\circ] \to \langle F^{\mathbf{w}}, \$_f^{\mathbf{e}}\rangle[\circ\circ]$$

Con respecto al axioma de la gramática, tenemos que $S = \langle \$_0^{\mathbf{w}}, \$_0^{\mathbf{w}}\rangle$.

Para realizar la demostración, consideraremos tres casos diferentes de derivación, tratando cada uno de los mismos por separado.

Caso 1. Existe una derivación en el autómata $(\mathbf{w}, E[\alpha_1], w) \overset{*}{\vdash} (\mathbf{w}, C[\alpha_1], \epsilon)$ para algún $\alpha_1 \in V_I^*$, en la que sólo se han aplicado transiciones de tipo **SWAP1**, si y sólo si existe una derivación en la gramática $\langle E^{\mathbf{w}}, C^{\mathbf{w}}\rangle[\,] \overset{*}{\Rightarrow} w$. La demostración se realiza por inducción. El caso base lo constituyen los dos casos siguientes:

- Si $(\mathbf{w}, E[\alpha_1], \epsilon) \overset{0}{\vdash} (\mathbf{w}, E[\alpha_1], \epsilon)$ entonces existe una producción $\langle E^{\mathbf{w}}, E^{\mathbf{w}}\rangle[\,] \to \epsilon$, por lo que $\langle E^{\mathbf{w}}, E^{\mathbf{w}}\rangle[\,] \overset{*}{\Rightarrow} \epsilon$.

- Si $\langle E^{\mathbf{w}}, E^{\mathbf{w}}\rangle[\,] \to \epsilon$ entonces existe una derivación $(\mathbf{w}, E[\alpha_1], \epsilon) \overset{0}{\vdash} (\mathbf{w}, E[\alpha_1], \epsilon)$ en el autómata.

Por hipótesis de inducción suponemos que se cumple para toda derivación de longitud inferior a s. El paso de inducción contempla los dos casos siguientes:

- Si $(\mathbf{w}, E[\alpha_1], wa) \overset{s}{\vdash} (\mathbf{w}, C[\alpha_1], a) \vdash (\mathbf{w}, F[\alpha_1], \epsilon)$, entonces existe una producción $\langle E^{\mathbf{w}}, F^{\mathbf{w}}\rangle[\circ\circ] \to \langle E^{\mathbf{w}}, C^{\mathbf{w}}\rangle[\circ\circ]\ a$, por hipótesis de inducción tenemos que $\langle E^{\mathbf{w}}, C^{\mathbf{w}}\rangle[\,] \overset{*}{\Rightarrow} w$, por lo que $\langle E^{\mathbf{w}}, F^{\mathbf{w}}\rangle[\,] \overset{*}{\Rightarrow} wa$.
- Si $\langle E^{\mathbf{w}}, F^{\mathbf{w}}\rangle[\,] \Rightarrow \langle E^{\mathbf{w}}, C^{\mathbf{w}}\rangle[\,]\ a \overset{s}{\Rightarrow} wa$, entonces existe una transición $C[\circ\circ]\ \mathbf{w}\overset{a}{\longmapsto}\mathbf{w}\ F[\circ\circ]$, por hipótesis de inducción $(\mathbf{w}, E[\alpha_1], w) \overset{s}{\vdash} (\mathbf{w}, C[\alpha_1], \epsilon)$ para algún α_1 y en consecuencia $(\mathbf{w}, E[\alpha_1], wa) \overset{s}{\vdash} (\mathbf{w}, F[\alpha_1], \epsilon)$.

Caso 2. Existe una derivación del autómata

$$\begin{aligned}(\mathbf{w}, \Upsilon E[\alpha_1], w_1 w_2) &\overset{*}{\vdash} (\mathbf{w}, \Upsilon C[\alpha_1], w_2)\\ &\vdash (\mathbf{w}, \Upsilon C[\alpha_1] \models^{\mathbf{w}} F'[\,], w_2)\\ &\overset{*}{\vdash} (\mathbf{e}, \Upsilon C[\alpha_1] \models^{\mathbf{w}} F[\,], \epsilon)\\ &\vdash (\mathbf{w}, \Upsilon G[\alpha_1], \epsilon)\end{aligned}$$

si y sólo si $\langle E^{\mathbf{w}}, G^{\mathbf{w}}\rangle[\,] \overset{*}{\Rightarrow} w_1 w_2$. La demostración se obtiene inmediatamente a partir del caso anterior y de la existencia de una producción $\langle E^{\mathbf{w}}, G^{\mathbf{w}}\rangle[\circ\circ] \to \langle E^{\mathbf{w}}, C^{\mathbf{w}}\rangle[\circ\circ]\, \langle F'^{\mathbf{w}}, F^{\mathbf{e}}\rangle[\,]$.

Caso 3. El caso principal de esta demostración establece que $\langle E^{\mathbf{w}}, C^{\mathbf{e}}\rangle[\alpha] \overset{*}{\Rightarrow} w$ si y sólo si $(\mathbf{w}, E[\delta], w) \overset{*}{\vdash} (\mathbf{e}, C[\beta], \epsilon)$ y se cumple que $\delta = \gamma_1\gamma_2 \ldots \gamma_p$, $\beta = \eta_1\eta_2 \ldots \eta_p$ y $\alpha = \langle\gamma_1, \eta_1\rangle\langle\gamma_2, \eta_2\rangle \ldots \langle\gamma_p, \eta_p\rangle$, esto es, δ es la pila obtenida como resultado de la proyección del primer componente de los elementos almacenados en α mientras que β es la pila obtenida como resultado de la proyección del segundo componente de los elementos almacenados en la pila de índices α.

Tratamos a continuación de demostrar cada una de las direcciones de la implicación:

- Si una derivación $(\mathbf{w}, E[\delta], w) \overset{*}{\vdash} (\mathbf{e}, C[\beta], \epsilon)$ es el resultado de aplicar la secuencia $t_1, \ldots, t_s$ de transiciones en $\mathcal{T}$, entonces existe una secuencia $p_1, \ldots, p'_s$ de producciones en P tal que la derivación $\langle E^{\mathbf{w}}, C^{\mathbf{e}}\rangle[\alpha] \overset{*}{\Rightarrow} w$ resultado de aplicar $p_1, \ldots, p'_s$ reconoce w. La demostración se realiza por inducción en la longitud de la derivación del autómata. El caso base lo constituye la derivación $(\mathbf{w}, E[\,], \epsilon) \vdash (\mathbf{e}, C[\,], \epsilon)$, para la que existe la producción $\langle E^{\mathbf{w}}, C^{\mathbf{e}}\rangle[\,] \to \langle E^{\mathbf{w}}, E^{\mathbf{w}}\rangle[\,]\, a$ y la producción $\langle E^{\mathbf{w}}, E^{\mathbf{w}}\rangle[\,] \to \epsilon$, por lo que $\langle E^{\mathbf{w}}, C^{\mathbf{e}}\rangle[\,] \overset{*}{\Rightarrow} a$.
  Por hipótesis de inducción suponemos que la proposición se cumple para cualquier derivación del autómata de longitud s. En tal caso, durante el paso de inducción verificamos que se cumple para cualquier posible derivación de longitud mayor que s:
  - Si $(\mathbf{w}, E[\delta], wa) \overset{s}{\vdash} (\mathbf{e}, C[\beta], a) \vdash (\mathbf{e}, F[\beta], \epsilon)$, existe una producción $\langle E^{\mathbf{w}}, F^{\mathbf{e}}\rangle[\circ\circ] \to \langle E^{\mathbf{w}}, C^{\mathbf{e}}\rangle[\circ\circ]\, a$, por hipótesis de inducción se cumple que $\langle E^{\mathbf{w}}, C^{\mathbf{e}}\rangle[\alpha] \overset{*}{\Rightarrow} w$, y en consecuencia $\langle E^{\mathbf{w}}, F^{\mathbf{e}}\rangle[\alpha] \overset{*}{\Rightarrow} wa$.
  - Si $(\mathbf{w}, E[\delta], w_1 w_2) \overset{s_1}{\vdash} (\mathbf{e}, C[\beta], w_2) \vdash (\mathbf{w}, C[\beta] \models^{\mathbf{e}} F'[\,], w_2) \overset{s_2}{\vdash} (\mathbf{e}, C[\beta] \models^{\mathbf{e}} F[\,], \epsilon) \vdash (\mathbf{e}, G[\beta], \epsilon)$, existe una producción $\langle E^{\mathbf{w}}, G^{\mathbf{e}}\rangle[\circ\circ] \to \langle E^{\mathbf{w}}, C^{\mathbf{e}}\rangle[\circ\circ]\, \langle F'^{\mathbf{w}}, F^{\mathbf{e}}\rangle[\,]$, por hipótesis de inducción se cumple que $\langle E^{\mathbf{w}}, C^{\mathbf{e}}\rangle[\alpha] \overset{*}{\Rightarrow} w_1$ y $\langle F'^{\mathbf{w}}, F^{\mathbf{e}}\rangle[\,] \overset{*}{\Rightarrow} w_2$, y en consecuencia $\langle E^{\mathbf{w}}, G^{\mathbf{e}}\rangle[\alpha] \overset{*}{\Rightarrow} w_1 w_2$.
  - Si $(\mathbf{w}, E[\delta], w_1 w_2) \overset{s_1}{\vdash} (\mathbf{w}, C[\delta], w_2) \vdash (\mathbf{w}, C[\,] \to F'[\beta], w_2) \overset{s_2}{\vdash} (\mathbf{e}, C[\,] \to F[\beta], \epsilon) \vdash (\mathbf{e}, G[\beta], \epsilon)$, existe una producción $\langle E^{\mathbf{w}}, G^{\mathbf{e}}\rangle[\circ\circ] \to \langle E^{\mathbf{w}}, C^{\mathbf{w}}\rangle[\,]\, \langle F'^{\mathbf{w}}, F^{\mathbf{e}}\rangle[\circ\circ]$, por los casos 1 y 2 se cumple que $\langle E^{\mathbf{w}}, C^{\mathbf{w}}\rangle[\,] \overset{*}{\Rightarrow} w_1$ y $\langle F'^{\mathbf{w}}, F^{\mathbf{e}}\rangle[\,] \overset{*}{\Rightarrow} w_2$, y en consecuencia $\langle E^{\mathbf{w}}, G^{\mathbf{e}}\rangle[\alpha] \overset{*}{\Rightarrow} w_1 w_2$.
  - Si $(\mathbf{w}, E[\delta], w_1 w_2) \overset{s_1}{\vdash} (\mathbf{w}, C[\delta], w_2) \vdash (\mathbf{w}, C[\,] \nearrow F'[\beta\gamma'], w_2) \overset{s_2}{\vdash} (\mathbf{e}, C[\,] \nearrow F[\beta\eta'], \epsilon) \vdash (\mathbf{e}, G[\beta], \epsilon)$, existe una producción $\langle E^{\mathbf{w}}, G^{\mathbf{e}}\rangle[\circ\circ] \to \langle E^{\mathbf{w}}, C^{\mathbf{w}}\rangle[\,]\, \langle F'^{\mathbf{w}}, F^{\mathbf{e}}\rangle[\circ\circ\langle\gamma', \eta'\rangle]$, por los casos 1 y 2 se cumple que $\langle E^{\mathbf{w}}, C^{\mathbf{w}}\rangle[\,] \overset{*}{\Rightarrow} w_1$ y $\langle F'^{\mathbf{w}}, F^{\mathbf{e}}\rangle[\alpha\langle\gamma', \eta\rangle] \overset{*}{\Rightarrow} w_2$, y en consecuencia $\langle E^{\mathbf{w}}, G^{\mathbf{e}}\rangle[\alpha] \overset{*}{\Rightarrow} w_1 w_2$.
  - Si $(\mathbf{w}, E[\delta\gamma], w_1 w_2) \overset{s_1}{\vdash} (\mathbf{w}, C[\delta], w_2) \vdash (\mathbf{w}, C[\,] \searrow F'[\beta], w_2) \overset{s_2}{\vdash} (\mathbf{e}, C[\,] \searrow F[\beta], \epsilon) \vdash (\mathbf{e}, G[\beta\eta], \epsilon)$, existe una producción $\langle E^{\mathbf{w}}, G^{\mathbf{e}}\rangle[\circ\circ\langle\gamma, \eta\rangle] \to \langle E^{\mathbf{w}}, C^{\mathbf{w}}\rangle[\,]\, \langle F'^{\mathbf{w}}, F^{\mathbf{e}}\rangle[\circ\circ]$, por los casos 1 y 2 se cumple que $\langle E^{\mathbf{w}}, C^{\mathbf{w}}\rangle[\,] \overset{*}{\Rightarrow} w_1$ y $\langle F'^{\mathbf{w}}, F^{\mathbf{e}}\rangle[\alpha] \overset{*}{\Rightarrow} w_2$, y en consecuencia $\langle E^{\mathbf{w}}, G^{\mathbf{e}}\rangle[\alpha\langle\gamma', \eta\rangle] \overset{*}{\Rightarrow} w_1 w_2$.

- Si una derivación izquierda $\langle C^{\mathbf{w}}, E^{\mathbf{e}}\rangle[\alpha] \stackrel{*}{\Rightarrow} w$ reconoce la cadena w como resultado de aplicar la secuencia $p_1, \ldots, p_s$ de producciones en P, entonces existe una secuencia de transiciones $t_1, \ldots, t_s$ en $\mathcal{T}$ tal que la derivación $(\mathbf{w}, C[\delta], w) \overset{*}{\vdash} (\mathbf{e}, E[\beta], \epsilon)$ es el resultado de aplicar la secuencia de transiciones $t_1, \ldots, t_s$. La demostración se realiza por inducción en la longitud de la derivación de la gramática.

  El caso base lo constituye la derivación $\langle E^{\mathbf{w}}, C^{\mathbf{e}}\rangle[\,] \Rightarrow \langle E^{\mathbf{w}}, E^{\mathbf{w}}\rangle[\,]\ a \Rightarrow a$, para la que existe una derivación $(\mathbf{w}, E[\,], a) \vdash (\mathbf{e}, C[\,], \epsilon)$ en el autómata.

  Por hipótesis de inducción suponemos que la proposición se cumple para cualquier derivación de la gramática de longitud s. En tal caso, durante el paso de inducción verificamos que se cumple para cualquier posible derivación de longitud mayor que s:

  - Si $\langle E^{\mathbf{w}}, F^{\mathbf{e}}\rangle[\alpha] \Rightarrow \langle E^{\mathbf{w}}, C^{\mathbf{e}}\rangle[\alpha]\ a \stackrel{s}{\Rightarrow} wa$, existe una transición $C[\circ\circ]\ \mathbf{e}\overset{a}{\longmapsto}\mathbf{e}\ F[\circ\circ]$, por hipótesis de inducción se cumple que $(\mathbf{w}, E[\delta], w) \overset{*}{\vdash} (\mathbf{e}, C[\beta], \epsilon)$ y por consiguiente $(\mathbf{w}, E[\delta], wa) \overset{*}{\vdash} (\mathbf{e}, F[\beta], \epsilon)$.
  - Si $\langle E^{\mathbf{w}}, G^{\mathbf{e}}\rangle[\alpha] \Rightarrow \langle E^{\mathbf{w}}, C^{\mathbf{e}}\rangle[\alpha]\ \langle F'^{\mathbf{w}}, F^{\mathbf{e}}\rangle[\,] \stackrel{s_1}{\Rightarrow} w_1\ \langle F'^{\mathbf{w}}, F^{\mathbf{e}}\rangle[\,] \stackrel{s_2}{\Rightarrow} w_1 w_2$, existe una transición $C[\circ\circ]\ \mathbf{e}\longmapsto\mathbf{w}\ C[\circ\circ] \models^{\mathbf{e}} F'[\,]$ y una transición $C[\,] \models^{\mathbf{e}} F[\,]\ \mathbf{e}\longmapsto\mathbf{e}\ G[\circ\circ]$, por hipótesis de inducción se cumple que $(\mathbf{w}, E[\delta], w_1) \overset{*}{\vdash} (\mathbf{e}, C[\beta], \epsilon)$ y que $(\mathbf{w}, F'[\ ], w_2) \overset{*}{\vdash} (\mathbf{e}, F[\ ], \epsilon)$ y en consecuencia se cumple que $(\mathbf{w}, E[\delta], w_1 w_2) \overset{*}{\vdash} (\mathbf{e}, C[\beta], w_2) \vdash (\mathbf{w}, C[\beta] \models^{\mathbf{e}} F'[\ ], w_2) \overset{*}{\vdash} (\mathbf{e}, C[\beta] \models^{\mathbf{e}} F[\,], \epsilon) \vdash (\mathbf{e}, G[\beta], \epsilon)$.
  - Si $\langle E^{\mathbf{w}}, G^{\mathbf{e}}\rangle[\alpha] \Rightarrow \langle E^{\mathbf{w}}, C^{\mathbf{w}}\rangle[\,]\ \langle F'^{\mathbf{w}}, F^{\mathbf{e}}\rangle[\alpha] \stackrel{s_1}{\Rightarrow} w_1\ \langle F'^{\mathbf{w}}, F^{\mathbf{e}}\rangle[\alpha] \stackrel{s_2}{\Rightarrow} w_1 w_2$, existe una transición $C[\circ\circ]\ \mathbf{w}\longmapsto\mathbf{w}\ C[\,] \rightarrow F'[\circ\circ]$ y una transición $C[\,] \rightarrow F[\circ\circ]\ \mathbf{e}\longmapsto\mathbf{e}\ G[\circ\circ]$, por los casos 1 y 2 se cumple que $(\mathbf{w}, E[\delta], w_1) \overset{*}{\vdash} (\mathbf{w}, C[\delta], \epsilon)$ y por hipótesis de inducción $(\mathbf{w}, F'[\delta], w_2) \overset{*}{\vdash} (\mathbf{e}, F[\beta], \epsilon)$ y en consecuencia se cumple que $(\mathbf{w}, E[\delta], w_1 w_2) \overset{*}{\vdash} (\mathbf{w}, C[\delta], w_2) \vdash (\mathbf{w}, C[\ ] \rightarrow F'[\delta], w_2) \overset{*}{\vdash} (\mathbf{e}, C[\,] \rightarrow F[\beta], \epsilon) \vdash (\mathbf{e}, G[\beta], \epsilon)$.
  - Si $\langle E^{\mathbf{w}}, G^{\mathbf{e}}\rangle[\alpha] \Rightarrow \langle E^{\mathbf{w}}, C^{\mathbf{w}}\rangle[\ ]\ \langle F'^{\mathbf{w}}, F^{\mathbf{e}}\rangle[\alpha\langle\gamma', \eta'\rangle] \stackrel{s_1}{\Rightarrow} w_1\ \langle F'^{\mathbf{w}}, F^{\mathbf{e}}\rangle[\alpha\langle\gamma', \eta'\rangle] \stackrel{s_2}{\Rightarrow} w_1 w_2$, existe una transición $C[\circ\circ]\ \mathbf{w}\longmapsto\mathbf{w}\ C[\ ] \nearrow F'[\circ\circ\gamma']$ y una transición $C[\circ\circ] \nearrow F[\circ\circ\eta']\ \mathbf{e}\longmapsto\mathbf{e}\ G[\circ\circ]$, por los casos 1 y 2 se cumple que $(\mathbf{w}, E[\delta], w_1) \overset{*}{\vdash} (\mathbf{w}, C[\delta], \epsilon)$ y por hipótesis de inducción $(\mathbf{w}, F'[\delta\gamma'], w_2) \overset{*}{\vdash} (\mathbf{e}, F[\beta\eta'], \epsilon)$ y en consecuencia se cumple que $(\mathbf{w}, E[\delta], w_1 w_2) \overset{*}{\vdash} (\mathbf{w}, C[\delta], w_2) \vdash (\mathbf{w}, C[\,] \nearrow F'[\delta\gamma'], w_2) \overset{*}{\vdash} (\mathbf{e}, C[\,] \nearrow F[\beta\eta'], \epsilon) \vdash (\mathbf{e}, G[\beta], \epsilon)$.
  - Si $\langle E^{\mathbf{w}}, G^{\mathbf{e}}\rangle[\alpha\langle\gamma, \eta\rangle] \Rightarrow \langle E^{\mathbf{w}}, C^{\mathbf{w}}\rangle[\,]\ \langle F'^{\mathbf{w}}, F^{\mathbf{e}}\rangle[\alpha] \stackrel{s_1}{\Rightarrow} w_1\ \langle F'^{\mathbf{w}}, F^{\mathbf{e}}\rangle[\alpha] \stackrel{s_2}{\Rightarrow} w_1 w_2$, existe una transición $C[\circ\circ\gamma]\ \mathbf{w}\longmapsto\mathbf{w}\ C[\,] \searrow F'[\circ\circ]$ y una transición $C[\,] \searrow F[\circ\circ]\ \mathbf{e}\longmapsto\mathbf{e}\ G[\circ\circ\eta]$, por los casos 1 y 2 se cumple que $(\mathbf{w}, E[\delta\gamma], w_1) \overset{*}{\vdash} (\mathbf{w}, C[\delta\gamma], \epsilon)$ y por hipótesis de inducción $(\mathbf{w}, F'[\delta], w_2) \overset{*}{\vdash} (\mathbf{e}, F[\beta], \epsilon)$ y en consecuencia se cumple que $(\mathbf{w}, E[\delta\gamma], w_1 w_2) \overset{*}{\vdash} (\mathbf{w}, C[\delta\gamma], w_2) \vdash (\mathbf{w}, C[\,] \searrow F'[\delta], w_2) \overset{*}{\vdash} (\mathbf{e}, C[\,] \searrow F[\beta], \epsilon) \vdash (\mathbf{e}, G[\beta\eta], \epsilon)$.

$\square$

Ejemplo 9.3 Consideremos el autómata lineal de índices fuertemente dirigido cuyas transiciones se muestran en la tabla 9.22, que acepta el lenguaje $\{a^n b^n c^n d^n \mid n > 0\}$. En la tabla 9.23 se muestra cómo dicho autómata acepta la cadena de entrada *aaabbbcccddd*. La primera columna indica la transición aplicada, la segunda señala el modo del autómata en ese momento, la tercera el contenido de la pila del autómata y la cuarta contiene la parte de la cadena de entrada que

(a)	$\$_0[\circ\circ]\ \mathbf{w}\!\longmapsto\!\mathbf{w}\ \$_0[\,]\models^{\mathbf{w}}A[\,]$
(b)	$A[\circ\circ]\ \mathbf{w}\!\overset{a}{\longmapsto}\!\mathbf{w}\ A'[\circ\circ]$
(c)	$A'[\circ\circ]\ \mathbf{w}\!\longmapsto\!\mathbf{w}\ A'[\,]\nearrow A[\circ\circ\gamma]$
(d)	$A'[\circ\circ]\ \mathbf{w}\!\overset{b}{\longmapsto}\!\mathbf{w}\ B'[\circ\circ]$
(e)	$B'[\circ\circ\gamma]\ \mathbf{w}\!\longmapsto\!\mathbf{w}\ B'[\,]\searrow B[\circ\circ]$
(f)	$B[\circ\circ]\ \mathbf{w}\!\overset{b}{\longmapsto}\!\mathbf{w}\ B'[\circ\circ]$
(g)	$B'[\,]\ \mathbf{w}\!\overset{c}{\longmapsto}\!\mathbf{e}\ C'[\,]$
(h)	$B'[\,]\searrow C'[\circ\circ]\ \mathbf{e}\!\longmapsto\!\mathbf{e}\ C[\circ\circ\eta]$
(i)	$C[\circ\circ]\ \mathbf{e}\!\overset{c}{\longmapsto}\!\mathbf{e}\ C'[\circ\circ]$
(j)	$C'[\circ\circ]\ \mathbf{e}\!\overset{d}{\longmapsto}\!\mathbf{e}\ D'[\circ\circ]$
(k)	$A'[\,]\nearrow D'[\circ\circ\eta]\ \mathbf{e}\!\longmapsto\!\mathbf{e}\ C[\circ\circ]$
(l)	$D[\circ\circ]\ \mathbf{e}\!\overset{d}{\longmapsto}\!\mathbf{e}\ D'[\circ\circ]$
(m)	$D'[\circ\circ]\ \mathbf{e}\!\longmapsto\!\mathbf{e}\ \$_f[\circ\circ]$

Tabla 9.22: Transiciones del SD–LIA que acepta $\{a^n b^n c^n d^n \mid n > 0\}$

	$\mathbf{w}$	$\models^{\mathbf{w}}\$_0[\,]$	$aaabbbcccddd$
(a)	$\mathbf{w}$	$\models^{\mathbf{w}}\$_0[\,]\models^{\mathbf{w}}A[\,]$	$aaabbbcccddd$
(b)	$\mathbf{w}$	$\models^{\mathbf{w}}\$_0[\,]\models^{\mathbf{w}}A'[\,]$	$aabbbcccddd$
(c)	$\mathbf{w}$	$\models^{\mathbf{w}}\$_0[\,]\models^{\mathbf{w}}A'[\,]\nearrow A[\gamma]$	$aabbbcccddd$
(b)	$\mathbf{w}$	$\models^{\mathbf{w}}\$_0[\,]\models^{\mathbf{w}}A'[\,]\nearrow A'[\gamma]$	$abbbcccddd$
(c)	$\mathbf{w}$	$\models^{\mathbf{w}}\$_0[\,]\models^{\mathbf{w}}A'[\,]\nearrow A'[\,]\nearrow A[\gamma\gamma]$	$abbbcccddd$
(b)	$\mathbf{w}$	$\models^{\mathbf{w}}\$_0[\,]\models^{\mathbf{w}}A'[\,]\nearrow A'[\,]\nearrow A'[\gamma\gamma]$	$bbbcccddd$
(d)	$\mathbf{w}$	$\models^{\mathbf{w}}\$_0[\,]\models^{\mathbf{w}}A'[\,]\nearrow A'[\,]\nearrow B'[\gamma\gamma]$	$bbcccddd$
(e)	$\mathbf{w}$	$\models^{\mathbf{w}}\$_0[\,]\models^{\mathbf{w}}A'[\,]\nearrow A'[\,]\nearrow B'[\,]\searrow B[\gamma]$	$bbcccddd$
(f)	$\mathbf{w}$	$\models^{\mathbf{w}}\$_0[\,]\models^{\mathbf{w}}A'[\,]\nearrow A'[\,]\nearrow B'[\,]\searrow B'[\gamma]$	$bcccddd$
(e)	$\mathbf{w}$	$\models^{\mathbf{w}}\$_0[\,]\models^{\mathbf{w}}A'[\,]\nearrow A'[\,]\nearrow B'[\,]\searrow B'[\,]\searrow B[\,]$	$bcccddd$
(f)	$\mathbf{w}$	$\models^{\mathbf{w}}\$_0[\,]\models^{\mathbf{w}}A'[\,]\nearrow A'[\,]\nearrow B'[\,]\searrow B'[\,]\searrow B'[\,]$	$cccddd$
(g)	$\mathbf{e}$	$\models^{\mathbf{w}}\$_0[\,]\models^{\mathbf{w}}A'[\,]\nearrow A'[\,]\nearrow B'[\,]\searrow B'[\,]\searrow C'[\,]$	$ccddd$
(h)	$\mathbf{e}$	$\models^{\mathbf{w}}\$_0[\,]\models^{\mathbf{w}}A'[\,]\nearrow A'[\,]\nearrow B'[\,]\searrow C[\eta]$	$ccddd$
(i)	$\mathbf{e}$	$\models^{\mathbf{w}}\$_0[\,]\models^{\mathbf{w}}A'[\,]\nearrow A'[\,]\nearrow B'[\,]\searrow C'[\eta]$	$cddd$
(h)	$\mathbf{e}$	$\models^{\mathbf{w}}\$_0[\,]\models^{\mathbf{w}}A'[\,]\nearrow A'[\,]\nearrow C[\eta\eta]$	$cddd$
(i)	$\mathbf{e}$	$\models^{\mathbf{w}}\$_0[\,]\models^{\mathbf{w}}A'[\,]\nearrow A'[\,]\nearrow C'[\eta\eta]$	ddd
(j)	$\mathbf{e}$	$\models^{\mathbf{w}}\$_0[\,]\models^{\mathbf{w}}A'[\,]\nearrow A'[\,]\nearrow D'[\eta\eta]$	dd
(k)	$\mathbf{e}$	$\models^{\mathbf{w}}\$_0[\,]\models^{\mathbf{w}}A'[\,]\nearrow D[\eta]$	dd
(l)	$\mathbf{e}$	$\models^{\mathbf{w}}\$_0[\,]\models^{\mathbf{w}}A'[\,]\nearrow D'[\eta]$	d
(k)	$\mathbf{e}$	$\models^{\mathbf{w}}\$_0[\,]\models^{\mathbf{w}}D[\,]$	d
(l)	$\mathbf{e}$	$\models^{\mathbf{w}}\$_0[\,]\models^{\mathbf{w}}D'[\,]$	
(m)	$\mathbf{e}$	$\models^{\mathbf{w}}\$_0[\,]\models^{\mathbf{w}}\$_f[\,]$	

Tabla 9.23: Configuraciones del SD–LIA para la cadena de entrada *aaabbbcccddd*

(a)	$\langle \$_0^{\mathbf{w}}, \$_0^{\mathbf{w}} \rangle[\circ\circ] \rightarrow \langle A^{\mathbf{w}}, \$_f^{\mathbf{e}} \rangle[\circ\circ]$
(b)	$\langle \Gamma^{\mathbf{w}}, A'^{\mathbf{w}} \rangle[\circ\circ] \rightarrow \langle \Gamma, A^{\mathbf{w}} \rangle[\circ\circ] \;\; a$
(c)	$\langle \Gamma^{\mathbf{w}}, D^{\mathbf{e}} \rangle[\circ\circ] \rightarrow \langle \Gamma, A'^{\mathbf{w}} \rangle[\,] \;\; \langle A^{\mathbf{w}}, D'^{\mathbf{e}} \rangle[\circ\circ\langle\gamma,\eta\rangle]$
(d)	$\langle \Gamma^{\mathbf{w}}, B'^{\mathbf{w}} \rangle[\circ\circ] \rightarrow \langle \Gamma, A'^{\mathbf{w}} \rangle[\circ\circ] \;\; b$
(e)	$\langle \Gamma^{\mathbf{w}}, C^{\mathbf{e}} \rangle[\circ\circ\langle\eta,\gamma\rangle] \rightarrow \langle \Gamma, B'^{\mathbf{w}} \rangle[\,] \;\; \langle B^{\mathbf{w}}, C'^{\mathbf{e}} \rangle[\circ\circ]$
(f)	$\langle \Gamma^{\mathbf{w}}, B'^{\mathbf{w}} \rangle[\circ\circ] \rightarrow \langle \gamma, B^{\mathbf{w}} \rangle[\circ\circ] \;\; b$
(g)	$\langle \Gamma^{\mathbf{w}}, C'^{\mathbf{e}} \rangle[\circ\circ] \rightarrow \langle \Gamma, B'^{\mathbf{w}} \rangle[\circ\circ] \;\; c$
(i)	$\langle \Gamma^{\mathbf{w}}, C'^{\mathbf{e}} \rangle[\circ\circ] \rightarrow \langle \Gamma, C^{\mathbf{e}} \rangle[\circ\circ] \;\; c$
(j)	$\langle \Gamma^{\mathbf{w}}, D'^{\mathbf{e}} \rangle[\circ\circ] \rightarrow \langle \Gamma, C'^{\mathbf{e}} \rangle[\circ\circ] \;\; d$
(l)	$\langle \Gamma^{\mathbf{w}}, D'^{\mathbf{e}} \rangle[\circ\circ] \rightarrow \langle \Gamma, D^{\mathbf{e}} \rangle[\circ\circ] \;\; d$
(m)	$\langle \Gamma^{\mathbf{w}}, \$_f^{\mathbf{e}} \rangle[\circ\circ] \rightarrow \langle \Gamma, D'^{\mathbf{e}} \rangle[\circ\circ]$
(n)	$\langle A^m, A^m \rangle[\,] \rightarrow \epsilon$
(o)	$\langle B^m, B^m \rangle[\,] \rightarrow \epsilon$
(p)	$\langle C^m, C^m \rangle[\,] \rightarrow \epsilon$
(q)	$\langle D^m, D^m \rangle[\,] \rightarrow \epsilon$
(r)	$\langle \$_0^{\mathbf{w}}, \$_0^{\mathbf{w}} \rangle[\,] \rightarrow \epsilon$

Tabla 9.24: Producciones de la LIG obtenida a partir del SD–LIA

		$\langle \$_0^{\mathbf{w}}, \$_0^{\mathbf{w}} \rangle[\,]$
(a)	$\Rightarrow$	$\langle \$_0^{\mathbf{w}}, \$_0^{\mathbf{w}} \rangle[\,] \;\; \langle A^{\mathbf{w}}, \$_f^{\mathbf{e}} \rangle[\,]$
(m)	$\Rightarrow$	$\langle \$_0^{\mathbf{w}}, \$_0^{\mathbf{w}} \rangle[\,] \;\; \langle A^{\mathbf{w}}, D'^{\mathbf{e}} \rangle[\,]$
(l)	$\Rightarrow$	$\langle \$_0^{\mathbf{w}}, \$_0^{\mathbf{w}} \rangle[\,] \;\; \langle A^{\mathbf{w}}, D^{\mathbf{e}} \rangle[\,] \;\; d$
(c)	$\Rightarrow$	$\langle \$_0^{\mathbf{w}}, \$_0^{\mathbf{w}} \rangle[\,] \;\; \langle A^{\mathbf{w}}, A'^{\mathbf{w}} \rangle[\,] \;\; \langle A^{\mathbf{w}}, D'^{\mathbf{e}} \rangle[\langle\gamma,\eta\rangle] \;\; d$
(l)	$\Rightarrow$	$\langle \$_0^{\mathbf{w}}, \$_0^{\mathbf{w}} \rangle[\,] \;\; \langle A^{\mathbf{w}}, A'^{\mathbf{w}} \rangle[\,] \;\; \langle A^{\mathbf{w}}, D^{\mathbf{e}} \rangle[\langle\gamma,\eta\rangle] \;\; dd$
(c)	$\Rightarrow$	$\langle \$_0^{\mathbf{w}}, \$_0^{\mathbf{w}} \rangle[\,] \;\; \langle A^{\mathbf{w}}, A'^{\mathbf{w}} \rangle[\,] \;\; \langle A^{\mathbf{w}}, A'^{\mathbf{w}} \rangle[\,] \;\; \langle A^{\mathbf{w}}, D'^{\mathbf{e}} \rangle[\langle\gamma,\eta\rangle\langle\gamma,\eta\rangle] \;\; dd$
(j)	$\Rightarrow$	$\langle \$_0^{\mathbf{w}}, \$_0^{\mathbf{w}} \rangle[\,] \;\; \langle A^{\mathbf{w}}, A'^{\mathbf{w}} \rangle[\,] \;\; \langle A^{\mathbf{w}}, A'^{\mathbf{w}} \rangle[\,] \;\; \langle A^{\mathbf{w}}, C'^{\mathbf{e}} \rangle[\langle\gamma,\eta\rangle\langle\gamma,\eta\rangle] \;\; ddd$
(i)	$\Rightarrow$	$\langle \$_0^{\mathbf{w}}, \$_0^{\mathbf{w}} \rangle[\,] \;\; \langle A^{\mathbf{w}}, A'^{\mathbf{w}} \rangle[\,] \;\; \langle A^{\mathbf{w}}, A'^{\mathbf{w}} \rangle[\,] \;\; \langle A^{\mathbf{w}}, C^{\mathbf{e}} \rangle[\langle\gamma,\eta\rangle\langle\gamma,\eta\rangle] \;\; cddd$
(e)	$\Rightarrow$	$\langle \$_0^{\mathbf{w}}, \$_0^{\mathbf{w}} \rangle[\,] \;\; \langle A^{\mathbf{w}}, A'^{\mathbf{w}} \rangle[\,] \;\; \langle A^{\mathbf{w}}, A'^{\mathbf{w}} \rangle[\,] \;\; \langle A^{\mathbf{w}}, B'^{\mathbf{w}} \rangle[\,] \;\; \langle B^{\mathbf{w}}, C'^{\mathbf{e}} \rangle[\langle\gamma,\eta\rangle] \;\; cddd$
(i)	$\Rightarrow$	$\langle \$_0^{\mathbf{w}}, \$_0^{\mathbf{w}} \rangle[\,] \;\; \langle A^{\mathbf{w}}, A'^{\mathbf{w}} \rangle[\,] \;\; \langle A^{\mathbf{w}}, A'^{\mathbf{w}} \rangle[\,] \;\; \langle A^{\mathbf{w}}, B'^{\mathbf{w}} \rangle[\,] \;\; \langle B^{\mathbf{w}}, C^{\mathbf{e}} \rangle[\langle\gamma,\eta\rangle] \;\; ccddd$
(e)	$\Rightarrow$	$\langle \$_0^{\mathbf{w}}, \$_0^{\mathbf{w}} \rangle[\,] \;\; \langle A^{\mathbf{w}}, A'^{\mathbf{w}} \rangle[\,] \;\; \langle A^{\mathbf{w}}, A'^{\mathbf{w}} \rangle[\,] \;\; \langle A^{\mathbf{w}}, B'^{\mathbf{w}} \rangle[\,] \;\; \langle B^{\mathbf{w}}, B'^{\mathbf{w}} \rangle[\,] \;\; \langle B^{\mathbf{w}}, C'^{\mathbf{e}} \rangle[\,] \;\; ccddd$
(g)	$\Rightarrow$	$\langle \$_0^{\mathbf{w}}, \$_0^{\mathbf{w}} \rangle[\,] \;\; \langle A^{\mathbf{w}}, A'^{\mathbf{w}} \rangle[\,] \;\; \langle A^{\mathbf{w}}, A'^{\mathbf{w}} \rangle[\,] \;\; \langle A^{\mathbf{w}}, B'^{\mathbf{w}} \rangle[\,] \;\; \langle B^{\mathbf{w}}, B'^{\mathbf{w}} \rangle[\,] \;\; \langle B^{\mathbf{w}}, B'^{\mathbf{w}} \rangle[\,] \;\; cccddd$
(f)	$\Rightarrow$	$\langle \$_0^{\mathbf{w}}, \$_0^{\mathbf{w}} \rangle[\,] \;\; \langle A^{\mathbf{w}}, A'^{\mathbf{w}} \rangle[\,] \;\; \langle A^{\mathbf{w}}, A'^{\mathbf{w}} \rangle[\,] \;\; \langle A^{\mathbf{w}}, B'^{\mathbf{w}} \rangle[\,] \;\; \langle B^{\mathbf{w}}, B'^{\mathbf{w}} \rangle[\,] \;\; \langle B^{\mathbf{w}}, B^{\mathbf{w}} \rangle[\,] \;\; bcccddd$
(o)	$\Rightarrow$	$\langle \$_0^{\mathbf{w}}, \$_0^{\mathbf{w}} \rangle[\,] \;\; \langle A^{\mathbf{w}}, A'^{\mathbf{w}} \rangle[\,] \;\; \langle A^{\mathbf{w}}, A'^{\mathbf{w}} \rangle[\,] \;\; \langle A^{\mathbf{w}}, B'^{\mathbf{w}} \rangle[\,] \;\; \langle B^{\mathbf{w}}, B'^{\mathbf{w}} \rangle[\,] \;\; bcccddd$
(f)	$\Rightarrow$	$\langle \$_0^{\mathbf{w}}, \$_0^{\mathbf{w}} \rangle[\,] \;\; \langle A^{\mathbf{w}}, A'^{\mathbf{w}} \rangle[\,] \;\; \langle A^{\mathbf{w}}, A'^{\mathbf{w}} \rangle[\,] \;\; \langle A^{\mathbf{w}}, B'^{\mathbf{w}} \rangle[\,] \;\; \langle B^{\mathbf{w}}, B^{\mathbf{w}} \rangle[\,] \;\; bbcccddd$
(o)	$\Rightarrow$	$\langle \$_0^{\mathbf{w}}, \$_0^{\mathbf{w}} \rangle[\,] \;\; \langle A^{\mathbf{w}}, A'^{\mathbf{w}} \rangle[\,] \;\; \langle A^{\mathbf{w}}, A'^{\mathbf{w}} \rangle[\,] \;\; \langle A^{\mathbf{w}}, B'^{\mathbf{w}} \rangle[\,] \;\; bbcccddd$
(d)	$\Rightarrow$	$\langle \$_0^{\mathbf{w}}, \$_0^{\mathbf{w}} \rangle[\,] \;\; \langle A^{\mathbf{w}}, A'^{\mathbf{w}} \rangle[\,] \;\; \langle A^{\mathbf{w}}, A'^{\mathbf{w}} \rangle[\,] \;\; \langle A^{\mathbf{w}}, A'^{\mathbf{w}} \rangle[\,] \;\; bbbcccddd$
(b)	$\Rightarrow$	$\langle \$_0^{\mathbf{w}}, \$_0^{\mathbf{w}} \rangle[\,] \;\; \langle A^{\mathbf{w}}, A'^{\mathbf{w}} \rangle[\,] \;\; \langle A^{\mathbf{w}}, A'^{\mathbf{w}} \rangle[\,] \;\; \langle A^{\mathbf{w}}, A^{\mathbf{w}} \rangle[\,] \;\; abbbcccddd$
(n)	$\Rightarrow$	$\langle \$_0^{\mathbf{w}}, \$_0^{\mathbf{w}} \rangle[\,] \;\; \langle A^{\mathbf{w}}, A'^{\mathbf{w}} \rangle[\,] \;\; \langle A^{\mathbf{w}}, A'^{\mathbf{w}} \rangle[\,] \;\; abbbcccddd$
(b)	$\Rightarrow$	$\langle \$_0^{\mathbf{w}}, \$_0^{\mathbf{w}} \rangle[\,] \;\; \langle A^{\mathbf{w}}, A'^{\mathbf{w}} \rangle[\,] \;\; \langle A^{\mathbf{w}}, A^{\mathbf{w}} \rangle[\,] \;\; aabbbcccddd$
(n)	$\Rightarrow$	$\langle \$_0^{\mathbf{w}}, \$_0^{\mathbf{w}} \rangle[\,] \;\; \langle A^{\mathbf{w}}, A'^{\mathbf{w}} \rangle[\,] \;\; aabbbcccddd$
(b)	$\Rightarrow$	$\langle \$_0^{\mathbf{w}}, \$_0^{\mathbf{w}} \rangle[\,] \;\; \langle A^{\mathbf{w}}, A^{\mathbf{w}} \rangle[\,] \;\; aaabbbcccddd$
(n)	$\Rightarrow$	$\langle \$_0^{\mathbf{w}}, \$_0^{\mathbf{w}} \rangle[\,] \;\; aaabbbcccddd$
(r)	$\Rightarrow$	$aaabbbcccddd$

Tabla 9.25: Derivación en LIG de la cadena *aaabbbcccddd*

resta por leer. En la tabla 9.24 se muestran las producciones de la gramática lineal de índices obtenida a partir de las transiciones del autómata. Utilizamos Γ para representar cualquier símbolo de pila. La derivación de la cadena *aaabbbcccddd* con esta gramática se muestra en la tabla 9.25, en la cual la primera columna indica la producción aplicada. ¶

9.4.4 Tabulación

A continuación definimos los diferentes tipos de derivaciones que se pueden dar en los autómatas lineales de índices fuertemente dirigidos junto con los tipos de ítems que pueden ser utilizados para representar dichas derivaciones.

Derivaciones de llamada. Son aquellas que se inician y terminan en modo de escritura, por lo que presentan la forma

$$\begin{aligned}(\mathbf{w}, \Upsilon\ A[\delta], a_{h+1}\dots a_n) &\overset{*}{\vdash} (\mathbf{w}, \Upsilon\ A[\,]\ \Upsilon_1\ \otimes B[\delta\gamma], a_{i+1}\dots a_n)\\ &\overset{*}{\vdash} (\mathbf{w}, \Upsilon\ A[\,]\ \Upsilon_1\ \otimes C[\delta\gamma], a_{j+1}\dots a_n)\end{aligned}$$

donde $\Upsilon, \Upsilon_1 \in (\mathcal{D}\ V_S[V_I^*])^*$, $\gamma \in V_I$, $\delta \in (V_S[V_I^*])^*$, $\otimes \in \{\nearrow, \rightarrow, \searrow\}$, tanto $B[\alpha\gamma]$ como $C[\alpha\gamma]$ son descendientes dependientes de $A[\alpha]$ y no existe $(B[\delta\gamma], f) \neq (B[\delta\gamma], i)$ tal que

$$\begin{aligned}(\mathbf{w}, \Upsilon\ A[\delta], a_{h+1}\dots a_n) &\overset{*}{\vdash} (\mathbf{w}, \Upsilon\ A[\,]\ \Upsilon_1\ \otimes F[\delta\gamma], a_{f+1}\dots a_n)\\ &\overset{*}{\vdash} (\mathbf{w}, \Upsilon\ A[\,]\ \Upsilon_1\ \otimes B[\delta\gamma], a_{i+1}\dots a_n)\\ &\overset{*}{\vdash} (\mathbf{w}, \Upsilon\ A[\,]\ \Upsilon_1\ \otimes C[\delta\gamma], a_{j+1}\dots a_n)\end{aligned}$$

En la figura 9.6 se muestra una representación gráfica de las derivaciones de llamada.

Para cualquier $\Upsilon' \in (\mathcal{D}\ V_S[V_I^*])^*$ y $\alpha \in V_I^*$ se cumple que

$$\begin{aligned}(\mathbf{w}, \Upsilon'\ A[\alpha], a_{h+1}\dots a_n) &\overset{*}{\vdash} (\mathbf{w}, \Upsilon'\ A[\,]\ \Upsilon_1\ \otimes B[\alpha\gamma], a_{i+1}\dots a_n)\\ &\overset{*}{\vdash} (\mathbf{w}, \Upsilon'\ A[\,]\ \Upsilon_1\ \otimes C[\alpha\gamma], a_{j+1}\dots a_n)\end{aligned}$$

Es por ello que este tipo de derivaciones puede ser representado mediante ítems de la forma

$$[A, h \mid B, i, \gamma, \otimes C, j, \gamma \mid -, -, -, -]\mathbf{w}$$

Derivaciones de retorno. Son aquellas que se inician en modo de escritura y que finalizan en modo borrado, por lo que tienen la siguiente forma:

$$\begin{aligned}(\mathbf{w}, \Upsilon\ A[\delta], a_{h+1}\dots a_n) &\overset{*}{\vdash} (\mathbf{w}, \Upsilon\ A[\,]\ \Upsilon_1\ \otimes B[\delta\gamma], a_{i+1}\dots a_n)\\ &\overset{*}{\vdash} (\mathbf{w}, \Upsilon\ A[\,]\ \Upsilon_1\ \otimes B[\,]\ \Upsilon_2 \searrow D[\delta], a_{p+1}\dots a_n)\\ &\overset{*}{\vdash} (\mathbf{e}, \Upsilon\ A[\,]\ \Upsilon_1\ \otimes B[\,]\ \Upsilon_2 \searrow E[\beta], a_{q+1}\dots a_n)\\ &\overset{*}{\vdash} (\mathbf{e}, \Upsilon\ A[\,]\ \Upsilon_1\ \otimes C[\beta\eta], a_{j+1}\dots a_n)\end{aligned}$$

donde $\Upsilon, \Upsilon_1, \Upsilon_2 \in (\mathcal{D}\ V_S[V_I^*])^*$ $\gamma, \eta \in V_I$, $\delta, \beta \in (V_S[V_I^*])^*$, $\otimes \in \{\nearrow, \rightarrow, \searrow\}$, $B[\alpha\gamma]$ y $D[\alpha]$ son descendientes dependientes de $A[\alpha]$, $C[\alpha\eta]$ es un descendiente dependiente de $D[\alpha]$ y no existen $(F[\delta\gamma], f) \neq (B[\delta\gamma], i)$ ni $(G[\delta], g) \neq (D[\delta], p)$ tal que

$$\begin{array}{rl}(\mathbf{w}, \Upsilon\ A[\delta], a_{h+1}\dots a_n) & \overset{*}{\vdash} (\mathbf{w}, \Upsilon\ A[\,]\ \Upsilon_1\ \otimes F[\delta\gamma], a_{f+1}\dots a_n)\\ & \overset{*}{\vdash} (\mathbf{w}, \Upsilon\ A[\,]\ \Upsilon_1\ \otimes B[\delta\gamma], a_{i+1}\dots a_n)\\ & \overset{*}{\vdash} (\mathbf{w}, \Upsilon\ A[\,]\ \Upsilon_1\ \otimes B[\,]\ \Upsilon_2 \searrow G[\delta], a_{g+1}\dots a_n)\\ & \overset{*}{\vdash} (\mathbf{w}, \Upsilon\ A[\,]\ \Upsilon_1\ \otimes B[\,]\ \Upsilon_2 \searrow D[\delta], a_{p+1}\dots a_n)\\ & \overset{*}{\vdash} (\mathbf{e}, \Upsilon\ A[\,]\ \Upsilon_1\ \otimes B[\,]\ \Upsilon_2 \searrow E[\beta], a_{q+1}\dots a_n)\\ & \overset{*}{\vdash} (\mathbf{e}, \Upsilon\ A[\,]\ \Upsilon_1\ \otimes C[\beta\eta], a_{j+1}\dots a_n)\end{array}$$

Una representación gráfica de las derivaciones de retorno se muestra en la figura 9.7.

Para cualquier $\Upsilon' \in (\mathcal{D}\ V_S[V_I^*])^*$ y pilas de índices $\delta', \beta' \in V_I^*$ tal que existe una derivación $(\mathbf{w}, D[\delta'], a_{p+1}\dots a_n) \overset{*}{\vdash} (\mathbf{e}, E[\beta'], a_{q+1}\dots a_n)$ se cumple que

$$\begin{array}{rl}(\mathbf{w}, \Upsilon'\ A[\delta'], a_{h+1}\dots a_n) & \overset{*}{\vdash} (\mathbf{w}, \Upsilon'\ A[\,]\ \Upsilon_1\ \otimes B[\delta'\gamma], a_{i+1}\dots a_n)\\ & \overset{*}{\vdash} (\mathbf{w}, \Upsilon'\ A[\,]\ \Upsilon_1\ \otimes B[\,]\ \Upsilon_2 \searrow D[\delta'], a_{p+1}\dots a_n)\\ & \overset{*}{\vdash} (\mathbf{e}, \Upsilon'\ A[\,]\ \Upsilon_1\ \otimes B[\,]\ \Upsilon_2 \searrow E[\beta'], a_{q+1}\dots a_n)\\ & \overset{*}{\vdash} (\mathbf{e}, \Upsilon'\ A[\,]\ \Upsilon_1\ \otimes C[\beta'\eta], a_{j+1}\dots a_n)\end{array}$$

por lo que este tipo de derivaciones puede ser representado de modo compacto por ítems de la forma

$$[A, h \mid B, i, \gamma, \otimes C, j, \eta \mid D, p, E, q]\mathbf{e}$$

Derivaciones de puntos especiales. Son aquellas derivaciones que involucran la propagación de una pila de índices vacía. La importancia de este tipo de derivaciones radica en que son las únicas en las cuales es posible realizar el cambio de modo de escritura a borrado. Las derivaciones de puntos especiales presentan la siguiente forma:

$$(m, \Upsilon\ \otimes B[\,], a_{i+1}\dots a_n)\ \overset{*}{\vdash} (m', \Upsilon\ \otimes C[\,], a_{j+1}\dots a_n)$$

donde $\Upsilon \in (\mathcal{D}\ V_S[V_I^*])^*$, $\otimes \in \mathcal{D}$, $m \leq m'$ y no existe $(m'', F[\,], f) \neq (m, B[\,], i)$, con $m'' < m$, tal que

$$\begin{array}{rl}(m'', \Upsilon\ \otimes F[\,], a_{f+1}\dots a_n) & \overset{*}{\vdash} (m, \Upsilon\ \otimes B[\,], a_{i+1}\dots a_n)\\ & \overset{*}{\vdash} (m', \Upsilon\ \otimes C[\,], a_{j+1}\dots a_n)\end{array}$$

La figura 9.8 muestra las derivaciones de puntos especiales de un modo más gráfico.

Para todo $\Upsilon' \in (\mathcal{D}\ V_S[V_I^*])^*$ se cumple

$$(m, \Upsilon'\ B[\,], a_{i+1}\dots a_n)\ \overset{*}{\vdash} (m', \Upsilon'\ C[\,], a_{j+1}\dots a_n)$$

por lo que podemos utilizar los siguientes ítems para representar este tipo de derivaciones:

$$[-, - \mid B, i, -, \otimes C, j, - \mid -, -, -, -]m'$$

Los ítems se combinan mediante las reglas de combinación descritas en la tabla 9.26, a partir del ítem inicial

$$[-, - \mid \$_0, 0, -, \models^{\mathbf{w}} \$_0, 0, - \mid -, -, -, -]\mathbf{w}$$

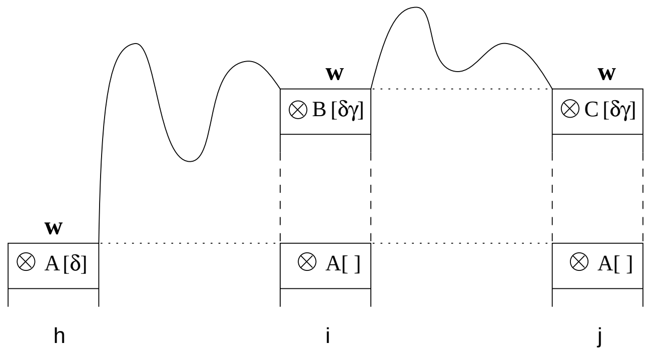

Figura 9.6: Derivaciones de llamada en SD–LIA

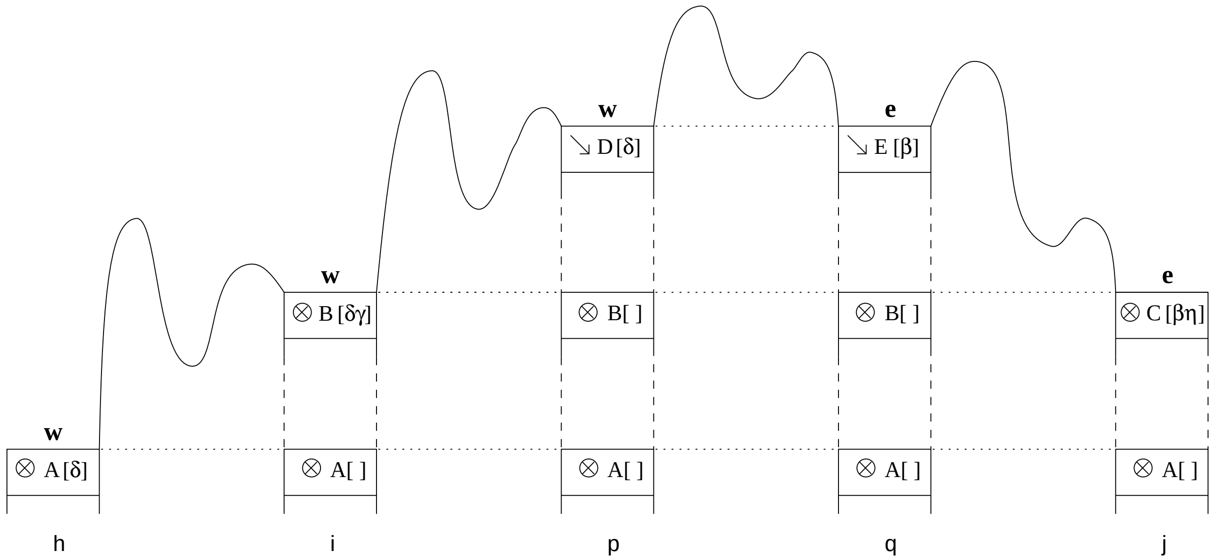

Figura 9.7: Derivaciones de retorno en SD–LIA

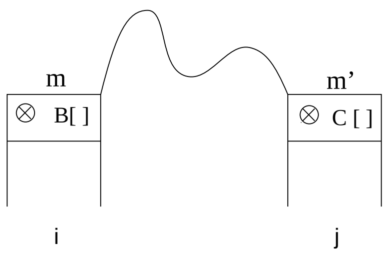

Figura 9.8: Derivaciones de puntos especiales en SD–LIA

$$\frac{[A,h \mid B,i,\gamma,\otimes C,j,\eta \mid D,p,E,q]m}{[A,h \mid B,i,\gamma,\otimes F,k,\eta \mid D,p,E,q]m} \quad C[\circ\circ]\ m \overset{a}{\longmapsto} m\ F[\circ\circ],\ \ k=j \text{ si } a=\epsilon,\ \ k=j+1 \text{ si } a \in V_T$$

$$\frac{[A,h \mid B,i,\gamma,\otimes C,j,\eta \mid D,p,E,q]m}{[-,- \mid F,j,-,\models^m F,j,- \mid -,-,-,-]\mathbf{w}} \quad C[\circ\circ]\ m \longmapsto \mathbf{w}\ C[\circ\circ] \models^m F[\,]$$

$$\frac{[A,h \mid B,i,\gamma,\otimes C,j,\gamma \mid -,-,-,-]\mathbf{w}}{[A,h \mid F,j,\gamma,\rightarrow F,j,\gamma \mid -,-,-,-]\mathbf{w}} \quad C[\circ\circ]\ \mathbf{w} \longmapsto \mathbf{w}\ C[\,] \rightarrow F[\circ\circ]$$

$$\frac{[A,h \mid B,i,\gamma,\otimes C,j,\gamma \mid -,-,-,-]\mathbf{w}}{[C,j \mid F,j,\gamma',\nearrow F,j,\gamma' \mid -,-,-,-]\mathbf{w}} \quad C[\circ\circ]\ \mathbf{w} \longmapsto \mathbf{w}\ C[\,] \nearrow F[\circ\circ\gamma']$$

$$\frac{\begin{array}{c}[A,h \mid B,i,\gamma,\otimes_1 C,j,\gamma \mid -,-,-,-]\mathbf{w} \\ [M,m \mid N,t,\gamma',\otimes_2 A,h,\gamma' \mid -,-,-,-]\mathbf{w}\end{array}}{[M,m \mid F,j,\gamma',\searrow F,j,\gamma' \mid -,-,-,-]\mathbf{w}} \quad C[\circ\circ\gamma]\ \mathbf{w} \longmapsto \mathbf{w}\ C[\,] \searrow F[\circ\circ]$$

$$\frac{[-,- \mid B,i,-,\otimes C,j,- \mid -,-,-,-]\mathbf{w}}{[-,- \mid B,i,-,\otimes F,k,- \mid -,-,-,-]\mathbf{e}} \quad C[\,]\ \mathbf{w} \overset{a}{\longmapsto} \mathbf{e}\ F[\,],\ \ k=j \text{ si } a=\epsilon,\ \ k=j+1 \text{ si } a \in V_T$$

$$\frac{\begin{array}{c}[-,- \mid F',j,-,\models^m F,k,- \mid -,-,-,-]\mathbf{e} \\ [A,h \mid B,i,\gamma,\otimes C,j,\eta \mid D,p,E,q]m\end{array}}{[A,h \mid B,i,\gamma,\otimes G,k,\eta \mid D,p,E,q]m} \quad \begin{array}{l} C[\circ\circ]\ m \longmapsto \mathbf{w}\ C[\circ\circ] \models^m F'[\,] \\ C[\circ\circ] \models^m F[\,]\ \mathbf{e} \longmapsto m\ G[\circ\circ]\end{array}$$

$$\frac{\begin{array}{c}[A,h \mid F',j,\gamma,\rightarrow F,k,\eta \mid D,p,E,q]\mathbf{e} \\ [A,h \mid B,i,\gamma,\otimes C,j,\gamma \mid -,-,-,-]\mathbf{w}\end{array}}{[A,h \mid B,i,\gamma,\otimes G,k,\eta \mid D,p,E,q]\mathbf{e}} \quad \begin{array}{l} C[\circ\circ]\ \mathbf{w} \longmapsto \mathbf{w}\ C[\,] \rightarrow F'[\circ\circ] \\ C[\,] \rightarrow F[\circ\circ]\ \mathbf{e} \longmapsto \mathbf{e}\ G[\circ\circ]\end{array}$$

$$\frac{\begin{array}{c}[C,j \mid F',j,\gamma',\nearrow F,k,\eta' \mid D,p,E,q]\mathbf{e} \\ [A,h \mid B,i,\gamma,\otimes C,j,\gamma \mid -,-,-,-]\mathbf{w} \\ [A,h \mid D,p,\gamma,\searrow E,q,\eta \mid O,u,P,v]\mathbf{e}\end{array}}{[A,h \mid B,i,\gamma,\otimes G,k,\eta \mid O,u,P,v]\mathbf{e}} \quad \begin{array}{l} C[\circ\circ]\ \mathbf{w} \longmapsto \mathbf{w}\ C[\,] \nearrow F'[\circ\circ\gamma'] \\ C[\,] \nearrow F[\circ\circ\eta']\ \mathbf{e} \longmapsto \mathbf{e}\ G[\circ\circ]\end{array}$$

$$\frac{\begin{array}{c}[M,m \mid F',j,\gamma',\searrow F,k,\eta' \mid D,p,E,q]\mathbf{e} \\ [A,h \mid B,i,\gamma,\otimes_1 C,j,\gamma \mid -,-,-,-]\mathbf{w} \\ [M,m \mid N,t,\gamma',\otimes_2 A,h,\gamma' \mid -,-,-,-]\mathbf{w}\end{array}}{[A,h \mid B,i,\gamma,\otimes_1 G,k,\eta \mid F',j,F,k]\mathbf{e}} \quad \begin{array}{l} C[\circ\circ\gamma]\ \mathbf{w} \longmapsto \mathbf{w}\ C[\,] \searrow F'[\circ\circ] \\ C[\,] \searrow F[\circ\circ]\ \mathbf{e} \longmapsto \mathbf{e}\ G[\circ\circ\eta]\end{array}$$

Tabla 9.26: Combinación de ítems en SD–LIA

La aceptación de una cadena de entrada $a_1 \dots a_n$ se indica mediante la presencia de ítems finales de la forma

$$[-,- \mid F,0,-,\models^{\mathbf{w}} \$_f,0,- \mid -,-,-,-]\mathbf{e}$$

tal que existe una transición $\$_0[\circ\circ]\ \mathbf{w}{\longmapsto}\mathbf{w}\ \$_0[\circ\circ] \models^{\mathbf{w}} F[\,]$.

Teorema 9.5 *La manipulación de configuraciones mediante la aplicación de transiciones en los autómatas lineales de índices fuertemente dirigidos es equivalente a la manipulación de ítems mediante las reglas de combinación de la tabla 9.26.*

Demostración:

Mostraremos que para toda derivación existe una regla de combinación que produce un ítem que representa de forma compacta dicha derivación y que toda regla de combinación se corresponde con una derivación válida del autómata. Para ello detallaremos todas las posibles derivaciones, junto con las reglas de combinación de ítems correspondientes, en la siguiente lista.

- Derivaciones que son el resultado de aplicar una transición $C[\circ\circ]\ m\overset{a}{\longmapsto}m\ F[\circ\circ]$
  - a una derivación de llamada:

$$\begin{array}{rl}(\mathbf{w}, \Upsilon\ A[\delta], a_{h+1}\dots a_n) & \overset{*}{\vdash} (\mathbf{w}, \Upsilon\ A[\,]\ \Upsilon_1\ \otimes B[\delta\gamma], a_{i+1}\dots a_n) \\ & \overset{*}{\vdash} (\mathbf{w}, \Upsilon\ A[\,]\ \Upsilon_1\ \otimes C[\delta\gamma], a_{j+1}\dots a_n) \\ & \vdash (\mathbf{w}, \Upsilon\ A[\,]\ \Upsilon_1\ \otimes F[\delta\gamma], a_{j+1}\dots a_n)\end{array}$$

$$\frac{[A,h \mid B,i,\gamma,\otimes C,j,\gamma \mid -,-,-,-]\mathbf{w}}{[A,h \mid B,i,\gamma,\otimes F,k,\gamma \mid -,-,-,-]\mathbf{w}}\quad C[\circ\circ]\ \mathbf{w}\overset{a}{\longmapsto}\mathbf{w}\ F[\circ\circ]$$

  donde $k = j$ si $a = \epsilon$ y $k = j+1$ si $a \in V_T$.
  - a una derivación de retorno:

$$\begin{array}{rl}(\mathbf{w}, \Upsilon\ A[\delta], a_{h+1}\dots a_n) & \overset{*}{\vdash} (\mathbf{w}, \Upsilon\ A[\,]\ \Upsilon_1\ \otimes B[\delta\gamma], a_{i+1}\dots a_n) \\ & \overset{*}{\vdash} (\mathbf{w}, \Upsilon\ A[\,]\ \Upsilon_1\ \otimes B[\,]\ \Upsilon_2 \searrow D[\delta], a_{p+1}\dots a_n) \\ & \overset{*}{\vdash} (\mathbf{e}, \Upsilon\ A[\,]\ \Upsilon_1\ \otimes B[\,]\ \Upsilon_2 \searrow E[\beta], a_{q+1}\dots a_n) \\ & \overset{*}{\vdash} (\mathbf{e}, \Upsilon\ A[\,]\ \Upsilon_1\ \otimes C[\beta\eta], a_{j+1}\dots a_n) \\ & \vdash (\mathbf{e}, \Upsilon\ A[\,]\ \Upsilon_1\ \otimes F[\beta\eta], a_{j+1}\dots a_n)\end{array}$$

$$\frac{[A,h \mid B,i,\gamma,\otimes C,j,\eta \mid D,p,E,q]\mathbf{e}}{[A,h \mid B,i,\gamma,\otimes F,k,\eta \mid D,p,E,q]\mathbf{e}}\quad C[\circ\circ]\ \mathbf{e}\overset{a}{\longmapsto}\mathbf{e}\ F[\circ\circ]$$

  donde $k = j$ si $a = \epsilon$ y $k = j+1$ si $a \in V_T$.
  - a una derivación de puntos especiales:

$$\begin{array}{rl}(m, \Upsilon\ \otimes B[\,], a_{i+1}\dots a_n) & \overset{*}{\vdash} (m', \Upsilon\ \otimes C[\,], a_{j+1}\dots a_n) \\ & \overset{*}{\vdash} (m', \Upsilon\ \otimes F[\,], a_{j+1}\dots a_n)\end{array}$$

$$\frac{[-,- \mid B,i,-,\otimes C,j,- \mid -,-,-,-]m'}{[-,- \mid B,i,-,\otimes F,k,- \mid -,-,-,-]m'}\quad C[\circ\circ]\ m'\overset{a}{\longmapsto}m'\ F[\circ\circ]$$

  donde $k = j$ si $a = \epsilon$ y $k = j+1$ si $a \in V_T$.
- Derivaciones que son el resultado de aplicar una transición $C[\circ\circ]\ m{\longmapsto}\mathbf{w}\ C[\circ\circ] \models^m F[\,]$

– a una derivación de llamada:

$$\begin{array}{ll}(\mathbf{w}, \Upsilon\ A[\delta], a_{h+1}\ldots a_n) & \overset{*}{\vdash} (\mathbf{w}, \Upsilon\ A[\,]\ \Upsilon_1\ \otimes B[\delta\gamma], a_{i+1}\ldots a_n)\\ & \overset{*}{\vdash} (\mathbf{w}, \Upsilon\ A[\,]\ \Upsilon_1\ \otimes C[\delta\gamma], a_{j+1}\ldots a_n)\\ & \vdash (\mathbf{w}, \Upsilon\ A[\,]\ \Upsilon_1\ \otimes C[\delta\gamma] \models^{\mathbf{w}} F[\,], a_{j+1}\ldots a_n)\end{array}$$

$$\frac{[A,h \mid B,i,\gamma,\otimes C,j,\gamma \mid -,-,-,-]\mathbf{w}}{[-,- \mid F,j,-,\models^{\mathbf{w}} F,j,- \mid -,-,-,-]\mathbf{w}}\quad C[\circ\circ]\ \mathbf{w}\!\longmapsto\!\mathbf{w}\ C[\circ\circ] \models^{\mathbf{w}} F[\,]$$

– a una derivación de retorno:

$$\begin{array}{ll}(\mathbf{w}, \Upsilon\ A[\delta], a_{h+1}\ldots a_n) & \overset{*}{\vdash} (\mathbf{w}, \Upsilon\ A[\,]\ \Upsilon_1\ \otimes B[\delta\gamma], a_{i+1}\ldots a_n)\\ & \overset{*}{\vdash} (\mathbf{w}, \Upsilon\ A[\,]\ \Upsilon_1\ \otimes B[\,]\ \Upsilon_2 \searrow D[\delta], a_{p+1}\ldots a_n)\\ & \overset{*}{\vdash} (\mathbf{e}, \Upsilon\ A[\,]\ \Upsilon_1\ \otimes B[\,]\ \Upsilon_2 \searrow E[\beta], a_{q+1}\ldots a_n)\\ & \overset{*}{\vdash} (\mathbf{e}, \Upsilon\ A[\,]\ \Upsilon_1\ \otimes C[\beta\eta], a_{j+1}\ldots a_n)\\ & \vdash (\mathbf{e}, \Upsilon\ A[\,]\ \Upsilon_1\ \otimes C[\beta\eta] \models^{\mathbf{e}} F[\,], a_{j+1}\ldots a_n)\end{array}$$

$$\frac{[A,h \mid B,i,\gamma,\otimes C,j,\eta \mid D,p,E,q]\mathbf{e}}{[-,- \mid F,j,-,\models^{\mathbf{e}} F,j,- \mid -,-,-,-]\mathbf{w}}\quad C[\circ\circ]\ \mathbf{e}\!\longmapsto\!\mathbf{w}\ C[\circ\circ] \models^{\mathbf{e}} F[\,]$$

– a una derivación de puntos especiales:

$$\begin{array}{ll}(m, \Upsilon\ \otimes B[\,], a_{i+1}\ldots a_n) & \overset{*}{\vdash} (m', \Upsilon\ \otimes C[\,], a_{j+1}\ldots a_n)\\ & \vdash (\mathbf{w}, \Upsilon\ \otimes C[\,] \models^{m'} F[\,], a_{j+1}\ldots a_n)\end{array}$$

$$\frac{[-,- \mid B,i,-,\otimes C,j,- \mid -,-,-,-]m'}{[-,- \mid F,j,-,\models^{m'} F,j,- \mid -,-,-,-]\mathbf{w}}\quad C[\circ\circ]\ m'\!\longmapsto\!\mathbf{w}\ C[\circ\circ] \models^{m'} F[\,]$$

- Derivaciones que son el resultado de aplicar una transición $C[\circ\circ]\ \mathbf{w}\!\longmapsto\!\mathbf{w}\ C[\,] \to F[\circ\circ]$
  - a una derivación de llamada:

$$\begin{array}{ll}(\mathbf{w}, \Upsilon\ A[\delta], a_{h+1}\ldots a_n) & \overset{*}{\vdash} (\mathbf{w}, \Upsilon\ A[\,]\ \Upsilon_1\ \otimes B[\delta\gamma], a_{i+1}\ldots a_n)\\ & \overset{*}{\vdash} (\mathbf{w}, \Upsilon\ A[\,]\ \Upsilon_1\ \otimes C[\delta\gamma], a_{j+1}\ldots a_n)\\ & \vdash (\mathbf{w}, \Upsilon\ A[\,]\ \Upsilon_1\ \otimes C[\,] \to F[\delta\gamma], a_{j+1}\ldots a_n)\end{array}$$

$$\frac{[A,h \mid B,i,\gamma,\otimes C,j,\gamma \mid -,-,-,-]\mathbf{w}}{[A,h \mid F,j,\gamma,\to F,j,\gamma \mid -,-,-,-]\mathbf{w}}\quad C[\circ\circ]\ \mathbf{w}\!\longmapsto\!\mathbf{w}\ C[\,] \to F[\circ\circ]$$

  - a una derivación de puntos especiales:

$$\begin{array}{ll}(\mathbf{w}, \Upsilon\ \otimes B[\,], a_{i+1}\ldots a_n) & \overset{*}{\vdash} (\mathbf{w}, \Upsilon\ \otimes C[\,], a_{j+1}\ldots a_n)\\ & \vdash (\mathbf{w}, \Upsilon\ \otimes C[\,] \to F[\,], a_{j+1}\ldots a_n)\end{array}$$

$$\frac{[-,- \mid B,i,-,\otimes C,j,- \mid -,-,-,-]\mathbf{w}}{[-,- \mid F,j,-,\to F,j,- \mid -,-,-,-]\mathbf{w}}\quad C[\circ\circ]\ \mathbf{w}\!\longmapsto\!\mathbf{w}\ C[\,] \to F[\circ\circ]$$

- Derivaciones que son el resultado de aplicar una transición $C[\circ\circ]\ \mathbf{w}\!\longmapsto\!\mathbf{w}\ C[\,] \nearrow F[\circ\circ\gamma']$
  - a una derivación de llamada:

$$\begin{array}{ll}(\mathbf{w}, \Upsilon\ A[\delta], a_{h+1}\ldots a_n) & \overset{*}{\vdash} (\mathbf{w}, \Upsilon\ A[\,]\ \Upsilon_1\ \otimes B[\delta\gamma], a_{i+1}\ldots a_n)\\ & \overset{*}{\vdash} (\mathbf{w}, \Upsilon\ A[\,]\ \Upsilon_1\ \otimes C[\delta\gamma], a_{j+1}\ldots a_n)\\ & \vdash (\mathbf{w}, \Upsilon\ A[\,]\ \Upsilon_1\ \otimes C[\,] \nearrow F[\delta\gamma\gamma'], a_{j+1}\ldots a_n)\end{array}$$

$$\frac{[A,h \mid B,i,\gamma,\otimes C,j,\gamma \mid -,-,-,-]\mathbf{w}}{[C,j \mid F,j,\gamma',\nearrow F,j,\gamma' \mid -,-,-,-]\mathbf{w}}\quad C[\circ\circ]\ \mathbf{w}\!\longmapsto\!\mathbf{w}\ C[\,] \nearrow F[\circ\circ\gamma']$$

– a una derivación de puntos especiales:

$$\begin{array}{ll}(\mathbf{w}, \Upsilon\ \otimes B[\,], a_{i+1}\dots a_n) & \overset{*}{\vdash} (\mathbf{w}, \Upsilon\ \otimes C[\,], a_{j+1}\dots a_n)\\ & \vdash (\mathbf{w}, \Upsilon\ \otimes C[\,] \to F[\gamma'], a_{j+1}\dots a_n)\end{array}$$

$$\frac{[-,-\mid B,i,-,\otimes C,j,-\mid -,-,-,-]\mathbf{w}}{[C,j\mid F,j,\gamma',\nearrow F,j,\gamma'\mid -,-,-,-]\mathbf{w}}\quad C[\circ\circ]\ \mathbf{w}\!\longmapsto\!\mathbf{w}\ C[\,]\nearrow F[\circ\circ\gamma']$$

- Derivaciones que son el resultado de aplicar una transición $C[\circ\circ\gamma]\ \mathbf{w}\!\longmapsto\!\mathbf{w}\ C[\,]\searrow F[\circ\circ]$ a una derivación de llamada:

$$\begin{array}{ll}(\mathbf{w}, \Upsilon\ M[\delta], a_{m+1}\dots a_n) & \overset{*}{\vdash} (\mathbf{w}, \Upsilon\ M[\,]\ \Upsilon_1\ \otimes_1 N[\delta\gamma'], a_{t+1}\dots a_n)\\ & \overset{*}{\vdash} (\mathbf{w}, \Upsilon\ M[\,]\ \Upsilon_1\ \otimes_1 A[\delta\gamma'], a_{h+1}\dots a_n)\\ & \overset{*}{\vdash} (\mathbf{w}, \Upsilon\ M[\,]\ \Upsilon_1\ \otimes_1 A[\,]\ \Upsilon_2\ \otimes_2 B[\delta\gamma'\gamma], a_{i+1}\dots a_n)\\ & \overset{*}{\vdash} (\mathbf{w}, \Upsilon\ M[\,]\ \Upsilon_1\ \otimes_1 A[\,]\ \Upsilon_2\ \otimes_2 C[\delta\gamma'\gamma], a_{j+1}\dots a_n)\\ & \vdash (\mathbf{w}, \Upsilon\ M[\,]\ \Upsilon_1\ \otimes_1 A[\,]\ \Upsilon_2\ \otimes_2 C[\,]\searrow F[\delta\gamma'], a_{j+1}\dots a_n)\end{array}$$

$$\frac{\begin{array}{c}[A,h\mid B,i,\gamma,\otimes_1 C,j,\gamma\mid -,-,-,-]\mathbf{w}\\ [M,m\mid N,t,\gamma',\otimes_2 A,h,\gamma'\mid -,-,-,-]\mathbf{w}\end{array}}{[M,m\mid F,j,\gamma',\searrow F,j,\gamma'\mid -,-,-,-]\mathbf{w}}\quad C[\circ\circ\gamma]\ \mathbf{w}\!\longmapsto\!\mathbf{w}\ C[\,]\searrow F[\circ\circ]$$

- Derivaciones que son el resultado de aplicar una transición $C[\,]\ \mathbf{w}\overset{a}{\longmapsto}\mathbf{e}\ F[\,]$ a una derivación de puntos especiales:

$$\begin{array}{ll}(\mathbf{w}, \Upsilon\ \otimes B[\,], a_{i+1}\dots a_n) & \overset{*}{\vdash} (\mathbf{w}, \Upsilon\ \otimes C[\,], a_{j+1}\dots a_n)\\ & \vdash (\mathbf{e}, \Upsilon\ \otimes F[\,], a_{k+1}\dots a_n)\end{array}$$

$$\frac{[-,-\mid B,i,-,\otimes C,j,-\mid -,-,-,-]\mathbf{w}}{[-,-\mid B,i,-,\otimes F,k,-\mid -,-,-,-]\mathbf{e}}\quad C[\,]\ \mathbf{w}\overset{a}{\longmapsto}\mathbf{e}\ F[\,]$$

donde $k = j$ si $a = \epsilon$ y $k = j + 1$ si $a \in V_T$.

- Derivaciones que son el resultado de aplicar una transición $C[\circ\circ] \models^m F[\,]\ \mathbf{e}\!\longmapsto\! m\ G[\circ\circ]$ a una derivación obtenida tras aplicar una transición $C[\circ\circ]\ m\!\longmapsto\!\mathbf{w}\ C[\circ\circ] \models^m F'[\,]$

– a una derivación de llamada:

$$\begin{array}{ll}(\mathbf{w}, \Upsilon\ A[\delta], a_{h+1}\dots a_n) & \overset{*}{\vdash} (\mathbf{w}, \Upsilon\ A[\,]\ \Upsilon_1\ \otimes B[\delta\gamma], a_{i+1}\dots a_n)\\ & \overset{*}{\vdash} (\mathbf{w}, \Upsilon\ A[\,]\ \Upsilon_1\ \otimes C[\delta\gamma], a_{j+1}\dots a_n)\\ & \vdash (\mathbf{w}, \Upsilon\ A[\,]\ \Upsilon_1\ \otimes C[\delta\gamma] \models^{\mathbf{w}} F'[\,], a_{j+1}\dots a_n)\\ & \overset{*}{\vdash} (\mathbf{e}, \Upsilon\ A[\,]\ \Upsilon_1\ \otimes C[\delta\gamma] \models^{\mathbf{w}} F[\,], a_{k+1}\dots a_n)\\ & \overset{*}{\vdash} (\mathbf{w}, \Upsilon\ A[\,]\ \Upsilon_1\ \otimes G[\delta\gamma], a_{k+1}\dots a_n)\end{array}$$

$$\frac{\begin{array}{c}[-,-\mid F',j,-,\models^m F,k,-\mid -,-,-,-]\mathbf{e}\\ [A,h\mid B,i,\gamma,\otimes C,j,\gamma\mid -,-,-,-]\mathbf{w}\end{array}}{[A,h\mid B,i,\gamma,\otimes G,k,\gamma\mid -,-,-,-]\mathbf{w}}\quad \begin{array}{l}C[\circ\circ]\ \mathbf{w}\!\longmapsto\!\mathbf{w}\ C[\circ\circ] \models^{\mathbf{w}} F'[\,]\\ C[\circ\circ] \models^{\mathbf{w}} F[\,]\ \mathbf{e}\!\longmapsto\!\mathbf{w}\ G[\circ\circ]\end{array}$$

– a una derivación de retorno:

$$\begin{array}{ll}(\mathbf{w}, \Upsilon\ A[\delta], a_{h+1}\dots a_n) & \overset{*}{\vdash} (\mathbf{w}, \Upsilon\ A[\,]\ \Upsilon_1\ \otimes B[\delta\gamma], a_{i+1}\dots a_n)\\ & \overset{*}{\vdash} (\mathbf{w}, \Upsilon\ A[\,]\ \Upsilon_1\ \otimes B[\,]\ \Upsilon_2 \searrow D[\delta], a_{p+1}\dots a_n)\\ & \overset{*}{\vdash} (\mathbf{e}, \Upsilon\ A[\,]\ \Upsilon_1\ \otimes B[\,]\ \Upsilon_2 \searrow E[\beta], a_{q+1}\dots a_n)\\ & \overset{*}{\vdash} (\mathbf{e}, \Upsilon\ A[\,]\ \Upsilon_1\ \otimes C[\beta\eta], a_{j+1}\dots a_n)\\ & \vdash (\mathbf{e}, \Upsilon\ A[\,]\ \Upsilon_1\ \otimes C[\beta\eta] \models^{\mathbf{e}} F'[\,], a_{j+1}\dots a_n)\\ & \overset{*}{\vdash} (\mathbf{e}, \Upsilon\ A[\,]\ \Upsilon_1\ \otimes C[\beta\eta] \models^{\mathbf{e}} F[\,], a_{k+1}\dots a_n)\\ & \vdash (\mathbf{e}, \Upsilon\ A[\,]\ \Upsilon_1\ \otimes G[\beta\eta], a_{k+1}\dots a_n)\end{array}$$

$$\frac{\begin{array}{c}[-,- \mid F',j,-,\models^{\mathbf{e}} F,k,- \mid -,-,-,-]\mathbf{e} \\ [A,h \mid B,i,\gamma,\otimes C,j,\eta \mid D,p,E,q]\mathbf{e}\end{array}}{[A,h \mid B,i,\gamma,\otimes G,k,\eta \mid D,p,E,q]\mathbf{e}} \quad \begin{array}{l} C[\circ\circ]\ \mathbf{e}\!\longmapsto\!\mathbf{w}\ C[\circ\circ] \models^{\mathbf{e}} F'[\,] \\ C[\circ\circ] \models^{\mathbf{e}} F[\,]\ \mathbf{e}\!\longmapsto\!\mathbf{e}\ G[\circ\circ]\end{array}$$

– a una derivación de puntos especiales:

$$\begin{array}{rl}(m,\Upsilon\ \otimes B[\,],a_{i+1}\dots a_n) & \overset{*}{\vdash} (m',\Upsilon\ \otimes C[\,],a_{j+1}\dots a_n) \\ & \vdash (\mathbf{w},\Upsilon\ \otimes C[\,] \models^{m'} F'[\,],a_{j+1}\dots a_n) \\ & \overset{*}{\vdash} (\mathbf{e},\Upsilon\ \otimes C[\,] \models^{m'} F[\,],a_{k+1}\dots a_n) \\ & \vdash (m',\Upsilon\ \otimes G[\,],a_{j+1}\dots a_n)\end{array}$$

$$\frac{\begin{array}{c}[-,- \mid F',j,-,\models^{m} F,k,- \mid -,-,-,-]\mathbf{e} \\ [-,- \mid B,i,-,\otimes C,j,- \mid -,-,-,-]m'\end{array}}{[-,- \mid B,i,-,\otimes G,k,- \mid -,-,-,-]m'} \quad \begin{array}{l} C[\circ\circ]\ m'\!\longmapsto\!\mathbf{w}\ C[\circ\circ] \models^{m'} F'[\,] \\ C[\circ\circ] \models^{m'} F[\,]\ \mathbf{e}\!\longmapsto\! m'\ G[\circ\circ]\end{array}$$

- Derivaciones que son el resultado de aplicar una transición $C[\,] \to F[\circ\circ]\ \mathbf{e}\!\longmapsto\!\mathbf{e}\ G[\circ\circ]$ a una derivación obtenida tras aplicar una transición $C[\circ\circ]\ \mathbf{w}\!\longmapsto\!\mathbf{w}\ C[\,] \to F'[\circ\circ]$

– a una derivación de llamada:

$$\begin{array}{rl}(\mathbf{w},\Upsilon\ A[\delta],a_{h+1}\dots a_n) & \overset{*}{\vdash} (\mathbf{w},\Upsilon\ A[\,]\ \Upsilon_1\ \otimes B[\delta\gamma],a_{i+1}\dots a_n) \\ & \overset{*}{\vdash} (\mathbf{w},\Upsilon\ A[\,]\ \Upsilon_1\ \otimes C[\delta\gamma],a_{j+1}\dots a_n) \\ & \vdash (\mathbf{w},\Upsilon\ A[\,]\ \Upsilon_1\ \otimes C[\,] \to F'[\delta\gamma],a_{j+1}\dots a_n) \\ & \overset{*}{\vdash} (\mathbf{w},\Upsilon\ A[\,]\ \Upsilon_1\ \otimes C[\,] \to F'[\,]\ \Upsilon_2 \searrow D[\delta],a_{p+1}\dots a_n) \\ & \overset{*}{\vdash} (\mathbf{e},\Upsilon\ A[\,]\ \Upsilon_1\ \otimes C[\,] \to F'[\,]\ \Upsilon_2 \searrow E[\beta],a_{q+1}\dots a_n) \\ & \overset{*}{\vdash} (\mathbf{e},\Upsilon\ A[\,]\ \Upsilon_1\ \otimes C[\,] \to F[\beta\eta],a_{k+1}\dots a_n) \\ & \overset{*}{\vdash} (\mathbf{e},\Upsilon\ A[\,]\ \Upsilon_1\ \otimes G[\beta\eta],a_{k+1}\dots a_n)\end{array}$$

$$\frac{\begin{array}{c}[A,h \mid F',j,\gamma,\to F,k,\eta \mid D,p,E,q]\mathbf{e} \\ [A,h \mid B,i,\gamma,\otimes C,j,\gamma \mid -,-,-,-]\mathbf{w}\end{array}}{[A,h \mid B,i,\gamma,\otimes G,k,\eta \mid D,p,E,q]\mathbf{e}} \quad \begin{array}{l} C[\circ\circ]\ \mathbf{w}\!\longmapsto\!\mathbf{w}\ C[\,] \to F'[\circ\circ] \\ C[\,] \to F[\circ\circ]\ \mathbf{e}\!\longmapsto\!\mathbf{e}\ G[\circ\circ]\end{array}$$

– a una derivación de puntos especiales:

$$\begin{array}{rl}(\mathbf{w},\Upsilon\ \otimes B[\,],a_{i+1}\dots a_n) & \overset{*}{\vdash} (\mathbf{w},\Upsilon\ \otimes C[\,],a_{j+1}\dots a_n) \\ & \vdash (\mathbf{w},\Upsilon\ \otimes C[\,] \to F'[\,],a_{j+1}\dots a_n) \\ & \overset{*}{\vdash} (\mathbf{e},\Upsilon\ \otimes C[\,] \to F[\,],a_{k+1}\dots a_n) \\ & \overset{*}{\vdash} (\mathbf{e},\Upsilon\ \otimes G[\,],a_{k+1}\dots a_n)\end{array}$$

$$\frac{\begin{array}{c}[-,- \mid F',j,-,\to F,k,- \mid -,-,-,-]\mathbf{e} \\ [-,- \mid B,i,-,\otimes C,j,- \mid -,-,-,-]\mathbf{w}\end{array}}{[-,- \mid B,i,-,\otimes G,k,- \mid -,-,-,-]\mathbf{e}} \quad \begin{array}{l} C[\circ\circ]\ \mathbf{w}\!\longmapsto\!\mathbf{w}\ C[\,] \to F'[\circ\circ] \\ C[\,] \to F[\circ\circ]\ \mathbf{e}\!\longmapsto\!\mathbf{e}\ G[\circ\circ]\end{array}$$

- Derivaciones que son el resultado de aplicar una transición $C[\,] \nearrow F[\circ\circ\eta]\ \mathbf{e}\!\longmapsto\!\mathbf{e}\ G[\circ\circ]$ a una derivación obtenida tras aplicar una transición $C[\circ\circ]\ \mathbf{w}\!\longmapsto\!\mathbf{w}\ C[\,] \nearrow F'[\circ\circ\gamma']$

– a una derivación de llamada:

$$\begin{array}{l}(\mathbf{w}, \Upsilon\ A[\delta], a_{h+1}\dots a_n)\\ \overset{*}{\vdash} (\mathbf{w}, \Upsilon\ A[\,]\ \Upsilon_1\ \otimes B[\delta\gamma], a_{i+1}\dots a_n)\\ \overset{*}{\vdash} (\mathbf{w}, \Upsilon\ A[\,]\ \Upsilon_1\ \otimes C[\delta\gamma], a_{j+1}\dots a_n)\\ \vdash (\mathbf{w}, \Upsilon\ A[\,]\ \Upsilon_1\ \otimes C[\,] \nearrow F'[\delta\gamma\gamma'], a_{j+1}\dots a_n)\\ \overset{*}{\vdash} (\mathbf{w}, \Upsilon\ A[\,]\ \Upsilon_1\ \otimes C[\,] \nearrow F'[\,]\ \Upsilon_2 \searrow D[\delta\gamma], a_{p+1}\dots a_n)\\ \overset{*}{\vdash} (\mathbf{w}, \Upsilon\ A[\,]\ \Upsilon_1\ \otimes C[\,] \nearrow F'[\,]\ \Upsilon_2 \searrow D[\,]\ \Upsilon_3 \searrow O[\delta], a_{u+1}\dots a_n)\\ \overset{*}{\vdash} (\mathbf{e}, \Upsilon\ A[\,]\ \Upsilon_1\ \otimes C[\,] \nearrow F'[\,]\ \Upsilon_2 \searrow D[\,]\ \Upsilon_3 \searrow P[\beta], a_{v+1}\dots a_n)\\ \overset{*}{\vdash} (\mathbf{e}, \Upsilon\ A[\,]\ \Upsilon_1\ \otimes C[\,] \nearrow F'[\,]\ \Upsilon_2 \searrow E[\beta\eta], a_{q+1}\dots a_n)\\ \overset{*}{\vdash} (\mathbf{e}, \Upsilon\ A[\,]\ \Upsilon_1\ \otimes C[\,] \nearrow F[\beta\eta\eta'], a_{k+1}\dots a_n)\\ \vdash (\mathbf{e}, \Upsilon\ A[\,]\ \Upsilon_1\ \otimes G[\beta\eta], a_{k+1}\dots a_n)\end{array}$$

$$\cfrac{\begin{array}{c}[C, j \mid F', j, \gamma', \nearrow F, k, \eta' \mid D, p, E, q]\mathbf{e}\\ [A, h \mid B, i, \gamma, \otimes C, j, \gamma \mid -, -, -, -]\mathbf{w}\\ [A, h \mid D, p, \gamma, \searrow E, q, \eta \mid O, u, P, v]\mathbf{e}\end{array}}{[A, h \mid B, i, \gamma, \otimes G, k, \eta \mid O, u, P, v]\mathbf{e}} \quad \begin{array}{l}C[\circ\circ]\ \mathbf{w}\longmapsto\mathbf{w}\ C[\,] \nearrow F'[\circ\circ\gamma']\\ C[\,] \nearrow F[\circ\circ\eta']\ \mathbf{e}\longmapsto\mathbf{e}\ G[\circ\circ]\end{array}$$

– a una derivación de puntos especiales:

$$\begin{array}{ll}(\mathbf{w}, \Upsilon\ \otimes B[\,], a_{i+1}\dots a_n) & \overset{*}{\vdash} (\mathbf{w}, \Upsilon\ \otimes C[\,], a_{j+1}\dots a_n)\\ & \vdash (\mathbf{w}, \Upsilon\ \otimes C[\,] \rightarrow F'[\gamma'], a_{j+1}\dots a_n)\\ & \overset{*}{\vdash} (\mathbf{w}, \Upsilon\ \otimes C[\,] \rightarrow F'[\,]\ \Upsilon_1 \searrow D[\,], a_{p+1}\dots a_n)\\ & \overset{*}{\vdash} (\mathbf{e}, \Upsilon\ \otimes C[\,] \rightarrow F'[\,]\ \Upsilon_1 \searrow E[\,], a_{q+1}\dots a_n)\\ & \overset{*}{\vdash} (\mathbf{e}, \Upsilon\ \otimes C[\,] \rightarrow F[\eta'], a_{k+1}\dots a_n)\\ & \vdash (\mathbf{e}, \Upsilon\ \otimes G[\,], a_{k+1}\dots a_n)\end{array}$$

$$\cfrac{\begin{array}{c}[C, j \mid F', j, \gamma', \nearrow F, k, \eta' \mid D, p, E, q]\mathbf{e}\\ [-, - \mid B, i, -, \otimes C, j, - \mid -, -, -, -]\mathbf{w}\\ [-, - \mid D, p, -, \searrow E, q, - \mid -, -, -, -]\mathbf{e}\end{array}}{[-, - \mid B, i, -, \otimes G, k, - \mid -, -, -, -]\mathbf{e}} \quad \begin{array}{l}C[\circ\circ]\ \mathbf{w}\longmapsto\mathbf{w}\ C[\,] \nearrow F'[\circ\circ\gamma']\\ C[\,] \nearrow F[\circ\circ\eta]\ \mathbf{e}\longmapsto\mathbf{e}\ G[\circ\circ]\end{array}$$

- Derivaciones que son el resultado de aplicar una transición $C[\,] \searrow F[\circ\circ]\ \mathbf{e}\longmapsto\mathbf{e}\ G[\circ\circ\eta]$ a una derivación obtenida tras aplicar una transición $C[\circ\circ\gamma]\ \mathbf{w}\longmapsto\mathbf{w}\ C[\,] \searrow F'[\circ\circ]$ a una derivación de llamada:

$$\begin{array}{l}(\mathbf{w}, \Upsilon\ M[\delta], a_{m+1}\dots a_n)\\ \overset{*}{\vdash} (\mathbf{w}, \Upsilon\ M[\,]\ \Upsilon_1\ \otimes_1 N[\delta\gamma'], a_{t+1}\dots a_n)\\ \overset{*}{\vdash} (\mathbf{w}, \Upsilon\ M[\,]\ \Upsilon_1\ \otimes_1 A[\delta\gamma'], a_{h+1}\dots a_n)\\ \overset{*}{\vdash} (\mathbf{w}, \Upsilon\ M[\,]\ \Upsilon_1\ \otimes_1 A[\,]\ \Upsilon_2\ \otimes_2 B[\delta\gamma'\gamma], a_{i+1}\dots a_n)\\ \overset{*}{\vdash} (\mathbf{w}, \Upsilon\ M[\,]\ \Upsilon_1\ \otimes_1 A[\,]\ \Upsilon_2\ \otimes_2 C[\delta\gamma'\gamma], a_{j+1}\dots a_n)\\ \vdash (\mathbf{w}, \Upsilon\ M[\,]\ \Upsilon_1\ \otimes_1 A[\,]\ \Upsilon_2\ \otimes_2 C[\,] \searrow F'[\delta\gamma'], a_{j+1}\dots a_n)\\ \overset{*}{\vdash} (\mathbf{w}, \Upsilon\ M[\,]\ \Upsilon_1\ \otimes_1 A[\,]\ \Upsilon_2\ \otimes_2 C[\,] \searrow F'[\,]\ \Upsilon_3 \searrow D[\delta], a_{p+1}\dots a_n)\\ \overset{*}{\vdash} (\mathbf{e}, \Upsilon\ M[\,]\ \Upsilon_1\ \otimes_1 A[\,]\ \Upsilon_2\ \otimes_2 C[\,] \searrow F'[\,]\ \Upsilon_3 \searrow E[\beta], a_{q+1}\dots a_n)\\ \overset{*}{\vdash} (\mathbf{e}, \Upsilon\ M[\,]\ \Upsilon_1\ \otimes_1 A[\,]\ \Upsilon_2\ \otimes_2 C[\,] \searrow F[\beta\eta'], a_{k+1}\dots a_n)\\ \vdash (\mathbf{e}, \Upsilon\ M[\,]\ \Upsilon_1\ \otimes_1 A[\,]\ \Upsilon_2\ \otimes_2 G[\beta\eta'\eta], a_{k+1}\dots a_n)\end{array}$$

$$\cfrac{\begin{array}{c}[M, m \mid F', j, \gamma', \searrow F, k, \eta \mid D, p, E, q]\mathbf{e}\\ [A, h \mid B, i, \gamma, \otimes_1 C, j, \gamma \mid -, -, -, -]\mathbf{w}\\ [M, m \mid N, t, \gamma', \otimes_2 A, h, \gamma' \mid -, -, -, -]\mathbf{w}\end{array}}{[A, h \mid B, i, \gamma, \otimes_1 G, k, \eta \mid F', j, F, k]\mathbf{e}} \quad \begin{array}{l}C[\circ\circ\gamma]\ \mathbf{w}\longmapsto\mathbf{w}\ C[\,] \searrow F'[\circ\circ]\\ C[\,] \searrow F[\circ\circ]\ \mathbf{e}\longmapsto\mathbf{e}\ G[\circ\circ\eta]\end{array}$$

A partir de esta lista se puede mostrar por inducción en la longitud de las derivaciones que tanto mediante la manipulación de configuraciones como mediante la manipulación de ítems se obtienen los mismos resultados. $\square$

La complejidad espacial de la técnica de tabulación propuesta es $\mathcal{O}(n^5)$ puesto que cada ítem almacena 5 posiciones de la cadena de entrada. La complejidad temporal en el peor caso es $\mathcal{O}(n^7)$ y viene dada por la siguiente regla:

$$\frac{\begin{array}{c}[C,j \mid F',j,\gamma',\nearrow F,k,\eta' \mid D,p,E,q]\mathbf{e}\\ [A,h \mid B,i,\gamma,\otimes C,j,\gamma \mid -,-,-,-]\mathbf{w}\\ [A,h \mid D,p,\gamma,\searrow E,q,\eta \mid O,u,P,v]\mathbf{e}\end{array}}{[A,h \mid B,i,\gamma,\otimes G,k,\eta \mid O,u,P,v]\mathbf{e}} \quad \begin{array}{l} C[\circ\circ]\ \mathbf{w}\longmapsto\mathbf{w}\ C[\,]\ \nearrow F'[\circ\circ\gamma'] \\ C[\,]\ \nearrow F[\circ\circ\eta']\ \mathbf{e}\longmapsto\mathbf{e}\ G[\circ\circ]\end{array}$$

Esta regla involucra la manipulación de 8 posiciones de la cadena de entrada, aunque mediante aplicación parcial sólo se necesita manipular simultáneamente 7 de dichas posiciones. La complejidad temporal puede reducirse utilizando la técnica propuesta en [53, 125]. Siguiendo dicha técnica, podemos descomponer la regla anterior en las dos reglas siguientes:

$$\frac{\begin{array}{c}[C,j \mid F',j,\gamma',\nearrow F,k,\eta' \mid D,p,E,q]\mathbf{e}\\ [A,h \mid D,p,\gamma,\searrow E,q,\eta \mid O,u,P,v]\mathbf{e}\end{array}}{[[C,j \mid F',j,\gamma',\nearrow F,k,\eta' \mid O,u,P,v]]\mathbf{e}} \quad \begin{array}{l} C[\circ\circ]\ \mathbf{w}\longmapsto\mathbf{w}\ C[\,]\ \nearrow F'[\circ\circ\gamma'] \\ C[\,]\ \nearrow F[\circ\circ\eta']\ \mathbf{e}\longmapsto\mathbf{e}\ G[\circ\circ]\end{array}$$

$$\frac{\begin{array}{c}[[C,j \mid F',j,\gamma',\nearrow F,k,\eta' \mid O,u,P,v]]\mathbf{e}\\ [A,h \mid B,i,\gamma,\otimes C,j,\gamma \mid -,-,-,-]\mathbf{w}\\ [A,h \mid D,p,\gamma,\searrow E,q,\eta \mid O,u,P,v]\mathbf{e}\end{array}}{[A,h \mid B,i,\gamma,\otimes G,k,\eta \mid O,u,P,v]\mathbf{e}} \quad \begin{array}{l} C[\circ\circ]\ \mathbf{w}\longmapsto\mathbf{w}\ C[\,]\ \nearrow F'[\circ\circ\gamma'] \\ C[\,]\ \nearrow F[\circ\circ\eta']\ \mathbf{e}\longmapsto\mathbf{e}\ G[\circ\circ]\end{array}$$

donde $[[C,j \mid F',j,\gamma',\nearrow F,k,\eta' \mid O,u,P,v]]\mathbf{e}$ es un pseudo-ítem que reparte equilibradamente la información entre ambas reglas y garantiza que la ejecución consecutiva de estas últimas sea equivalente a la ejecución de la regla original. La primera regla tiene complejidad $\mathcal{O}(n^6)$, puesto que la posición h no interviene en la combinación de posiciones de la cadena de entrada, al igual que la segunda, puesto que en esta última las posiciones p y q no intervienen.

Capítulo 10

Autómatas con dos pilas

En este capítulo se presenta un nuevo modelo de autómatas para el análisis de los lenguajes de adjunción de árboles, basado en la utilización de dos pilas que actúan coordinadamente. Las aportaciones de este capítulo se refieren a la definición de los autómatas con dos pilas fuertemente dirigidos y a la definición de los autómatas con dos pilas ascendentes, junto con las técnicas de tabulación para ambos. Este capítulo está basado en [53, 21, 54].

10.1 Introducción

En los capítulos anteriores se han definido diversas extensiones de los autómatas a pila que aceptan la clase de los lenguajes de adjunción de árboles. Básicamente, tales extensiones consisten en asociar a los elementos de la pila del autómata una pila de índices. En los autómatas lógicos a pila restringidos y en los autómatas lineales de índices esta característica está en la propia definición del modelo. En el caso de los autómatas a pila embebidos y su variante ascendente esta característica está también presente, aunque de modo implícito: la cima de cada una de las pilas individuales juega el papel de símbolo de pila mientras el resto de cada pila juega el papel de pila de índices asociada a dicho símbolo.

En este capítulo ofrecemos un enfoque diferente, puesto que en lugar de trabajar con una pila de símbolos de pila asociados a pilas de índices, trabajaremos con dos pilas, una equivalente a la pila de los autómatas a pila originales, y otra en la que se almacenan, en principio, índices, de tal modo que los elementos almacenados en esta segunda pila restringen los movimientos que se pueden realizar sobre la primera.

Una manera común de simular una máquina de Turing consiste en definir un autómata a pila que trabaja sobre dos pilas. Será, por tanto, necesario definir cuidadosamente el conjunto de transiciones permitido con el fin de diseñar modelos de autómatas con dos pilas que acepten exactamente los lenguajes de adjunción de árboles. En lo que resta de capítulo, se describen los autómatas con dos pilas tal y como fueron originalmente descritos por Becker [25], para continuar luego con la definición de los autómatas con dos pila fuertemente dirigidos [53] y los autómatas con dos pilas ascendentes [54].

10.2 Autómatas con dos pilas

Becker define en [25] los autómatas con dos pilas (*2–Stack automata*, 2–SA) como una extensión de los autómatas a pila en la que se permite realizar las siguientes operaciones:

- Leer un terminal de la cadena de entrada.

- Apilar un símbolo de pila en la primera pila.
- Eliminar de la primera pila el símbolo situado en su cima.
- Apilar un símbolo en la segunda pila.
- Situar en la cima de cada pila un símbolo especial denominado *separador*.
- Eliminar de la cima de las dos pilas el símbolo separador.
- En el caso de que en la cima de primera pila se encuentre un separador, eliminar de la segunda pila el símbolo situado en su cima

Formalmente, definiremos un autómata con dos pilas como una tupla $(Q, V_T, V_S, \delta, q_0, [\, , \$_0)$, donde:

- Q es un conjunto finito de estados.
- V_T es un conjunto finito de símbolos terminales.
- V_S es un conjunto finito de símbolos de pila.
- q_0 es el estado inicial.
- $[\notin V_S$ es el símbolo separador.
- $\$_0 \in V_S$ es el símbolo inicial de ambas pilas.
- δ es un conjunto finito de transiciones de alguno de los tres tipos siguientes:

$$(q', \alpha_1, Z_2\alpha_2) \in \delta(q, a, Z_1, Z_2)$$

$$(q', [\, , \alpha_3) \in \delta(q, a, [\, , Z_3)$$

$$(q', \epsilon, \epsilon) \in \delta(q, a, [\, , [\,)$$

  donde $q, q' \in Q$, $a \in V_T \cup \{\epsilon\}$, $Z_1, Z_3 \in V_S$, $Z_2 \in V_S \cup \{[\}$, $\alpha_1, \alpha_2 \in V_S^* \cup V_S [V_S^*$ cumpliéndose además que α_1 incluye un separador si y sólo si α_2 también lo incluye, y $\alpha_3 \in V_S^*$. El primer tipo de transiciones permite apilar elementos en la primera y/o segunda pila y extraer elementos de la primera pila. Las transiciones del segundo tipo permiten extraer símbolos de la segunda pila si y sólo si la cima de la primera pila está ocupada por un separador. El tercer tipo de transiciones permite eliminar los separadores de la cima de ambas pilas.

La *configuración* de un autómata con dos pilas en un momento dado viene definida por la tupla $(q, \Upsilon_1, \Upsilon_2, w)$, donde $q \in Q$ indica el estado en el que se encuentra, $\Upsilon_1, \Upsilon_2 \in ([V_S^*)^*$ es el contenido de la primera y segunda pila. El cambio de una configuración a otra viene determinado por la aplicación de una transición, de tal modo que si $(q, \Upsilon_1 Z_1, \Upsilon_2 Z_2, aw)$ es una configuración y $(q', \alpha_1, \alpha_2) \in \delta(q, a, Z_1, Z_2)$ es una transición, entonces el autómata con dos pilas pasa a la nueva configuración $(q', \Upsilon_1\alpha_1, \Upsilon_2\alpha_2, w)$. Este hecho se denota mediante

$$(q, \Upsilon_1 Z_1, \Upsilon_2 Z_2, aw) \vdash (q', \Upsilon_1\alpha_1, \Upsilon_2\alpha_2, w)$$

Denotamos por $\overset{*}{\vdash}$ el cierre reflexivo y transitivo de $\vdash$.

El *lenguaje aceptado por pila vacía* por un autómata con dos pilas viene determinado por el conjunto de cadenas $w \in V_T^*$ tal que $(q_0, [\$_0, [\$_0, w) \overset{*}{\vdash} (q, \epsilon, \epsilon, \epsilon)$ para cualquier $q \in Q$.

Los autómatas con dos pilas tal y como han sido definidos aceptan exactamente la clase de los lenguajes de adjunción de árboles. Para demostrarlo nos basaremos en la equivalencia entre los 2–SA y los autómatas a pila embebidos definidos en el capítulo 6.

Teorema 10.1 *Los lenguajes aceptados por los autómatas con dos pilas son un subconjunto de los lenguajes de adjunción de árboles.*

Demostración:

Siguiendo la demostración de Becker en [25], dado un autómata con dos pilas $\mathcal{A} = (Q, V_T, V_S, \delta, q_0, [\ , \$_0)$, construiremos un autómata a pila embebido con estados $\mathcal{E} = (Q', V_T, V_S', \delta', q_0', Q_F', \$_0)$ tal que $V_S' = V_S \cup \{\sharp\}$, los estados en Q' serán triples $\langle q, S_i, Z_2 \rangle$ con $q \in Q$, $Z_2 \in V_S \cup \{\sharp\}$, $q_0' = \langle q_0, S1, \$_0 \rangle$ y las transiciones en δ' se construirán a partir de las transiciones en δ de tal modo que:

$S_i = S_1$ si la cima de la primera pila del 2–SA no es un separador. La cima del EPDA equivaldrá a la cima de la primera pila del 2–SA y las pilas unitarias del EPDA situadas inmediatamente debajo de su cima representarán a los elementos de la segunda pila del 2-SA situados por encima del último separador. El resto de las pilas del EPDA repetirán esta representación para la siguiente porción de las pilas del 2–SA situada entre dos separadores.

$S_i = S_2$ si la cima de la primera pila del 2–SA es un separador. La cima del EPDA contendrá la cima de la segunda pila del 2–SA.

Para ello, transformaremos cada transición $(q', \alpha_1, Z_2\alpha_2) \in \delta(q, a, Z_1, Z_2)$, con $Z_1 \in V_S$, $\alpha_1 \in V_S^*$ y $\alpha_2 = Y_1 Y_2 \dots Y_m$, en la transición

$$(\langle q', S_1, Y_m \rangle, [Y_1 [Y_2 \dots [Y_m, \alpha_1, \epsilon) \in \delta'(\langle q, S_1, Z_2 \rangle, a, Z_1)$$

En el caso de que $\alpha_1 = \beta_1 [\beta_2$ y $\alpha_2 = Y_1 \dots Y_k [Y_{k+1} \dots Y_m$, la transición resultante del EPDA será

$$(\langle q', s_1, Y_m \rangle, [Y_1 \dots [Y_k, \beta_1, [[\sharp, Z_2][Y_{k+1} \dots [Y_m [\sharp\beta_2) \in \delta'(\langle q, S_1, Z_2 \rangle, a, Z_1)$$

donde $\sharp$ es un símbolo especial que representa el separador del 2–SA y $[\sharp, Z_2]$ es un símbolo especial que codifica la anterior cima de la segunda pila del 2–SA, con el fin de poder recuperarla posteriormente.

En el caso de que $Z_1 = [$ y $\alpha_2 = Y_1 \dots Y_m$, la transición del EPDA será

$$(\langle q', S_2, Y_m \rangle, \epsilon, \epsilon, [Y_1 \dots [Y_m) \in \delta'(\langle q, S_2, Z_2 \rangle, a, Z_2)$$

Adicionalmente, para todo $a \in Q$ crearemos las transiciones

$$(\langle q, S_2, Z_2 \rangle, \epsilon, \epsilon, \epsilon) \in \delta'(\langle q, S_1, Z_2 \rangle, \epsilon, \sharp)$$

que permiten comenzar a manipular la segunda del 2–SA cuando en la cima de la primera está un separador. También será preciso añadir, para todo $q \in Q$, las transiciones

$$(\langle q, S_1, Z' \rangle, \epsilon, \epsilon, \epsilon) \in \delta'(\langle q, S_2, Z_2 \rangle, \epsilon, [\sharp, Z'])$$

que permiten terminar la manipulación de la segunda pila del 2–SA cuando lleguemos a tener un separador en la cima de la misma. $\square$

Teorema 10.2 *Los lenguajes de adjunción de árboles son un subconjunto de los lenguajes aceptados por los autómatas con dos pilas.*

Demostración:

La demostración propuesta en [25] de que todo EPDA puede ser transformado en un 2–SA equivalente es errónea[1], por lo que plantearemos aquí una versión modificada de la misma.

Dado un autómata a pila embebido con estados $\mathcal{E} = (Q, V_T, V_S, \delta, q_0, Q_F, \$_0)$ construiremos un autómata con dos pilas $\mathcal{A} = (Q, V_T, V_S', \delta', q_0, [\,, \$_0)$, con $VS' = V_S \cup \{\sharp\}$. Cada transición del EPDA

$$(q', [Z_k' \ldots [Z_{i+1}', Z_m \ldots Z_1, [Z_i' \ldots [Z_1') \in \delta(q, a, Z)$$

donde $q, q' \in Q$, $a \in V_T \cup \{\epsilon\}$, $Z, Z_1, \ldots, Z_m, Z_1', \ldots, Z_k \in V_S$ y $m \leq 2$, se transformará en las siguientes transiciones del 2–SA, para todo $Z'' \in V_S$:

$$(q', Z_m \ldots Z_1 [Z_i' \ldots [Z_1', Z'' \sharp Z_k' \ldots Z_{i+1}' [\ldots [\,) \in \delta'(q, a, Z, Z'')$$

donde el número de separadores en ambas pilas es es el mismo y $\sharp$ es un nuevo símbolo de pila utilizado para separar las pilas unitarias que permanecen en espera en la segunda pila del 2–SA. Este tipo de transiciones no forman parte de las transiciones elementales que hemos propuesto para los autómatas con dos pilas puesto que se introduce más de un separador, pero pueden ser obtenidas mediante la aplicación sucesiva de dichas transiciones [25] y son utilizadas aquí para simplificar la prueba.

Adicionalmente, para todo $q \in Q$ deberemos añadir las transiciones

$$(q, \epsilon, \epsilon) \in \delta(q, a, [\,, [\,)$$

que eliminan los separadores, las transiciones

$$(q, [Z_2, \epsilon) \in \delta(q, \epsilon, [, Z_2)$$

que pasan elementos de la segunda pila a la primera para procesar aquellas pilas unitarias que habían sido dejadas en espera y las transiciones

$$(q, [, \epsilon) \in \delta(q, \epsilon, [, \sharp)$$

que finalizan el procesamiento de las pilas unitarias dejadas en espera. □

Los autómatas con dos pilas que acabamos de definir presentan un interés meramente teórico, puesto que la definición de esquemas de compilación para LIG y TAG dista de ser simple y porque no ha sido posible diseñar hasta el momento una técnica de tabulación que permite su ejecución en tiempo polinomial.

10.3 Autómatas con dos pilas fuertemente dirigidos

Puesto que los autómatas con dos pilas presentados en la sección precedente no son satisfactorios, procederemos a su redefinición. Los objetivos de la misma son, por una parte, permitir la descripción de forma simple de esquemas de compilación para LIG y TAG, y por otra parte posibilitar el desarrollo de una técnica de tabulación que permita ejecutar dichos autómatas en tiempo polinomial.

[1]Tilman Becker, comunicación personal, 1996.

Seguiremos suponiendo que dichos autómatas constan de dos pilas, una de las cuales recibirá el nombre de *pila maestra* (*master stack*, **MS**) mientras que la otra se denominará *pila auxiliar* (*auxiliary stack*, **AS**). En ambas pilas se almacenarán elementos de un alfabeto de pila y separadores. Denominaremos *sesión* a la parte de cada pila comprendida entre dos de dichos separadores. El número de sesiones debe ser el mismo en ambas pilas. Prescindiremos del control de estado finito, tal y como hemos hecho anteriormente en el caso de los autómatas a pila (capítulo 5) y los autómatas a pila embebidos (capítulos 6 y 7).

Si no se establece ninguna restricción, las dos pilas del autómata pueden ser utilizadas combinadamente para emular una máquina de Turing [85]. Con el fin de limitar la potencia expresiva a la clase de los lenguajes de adjunción de árboles, restringiremos las operaciones permitidas en cada pila en función de los dos modos siguientes de trabajo:

- Un *modo de escritura* **w** en el cual no se permite extraer elementos de la pila maestra.
- Un *modo de borrado* **e** en el cual no se permite apilar elementos en la pila maestra.

En caso de creación de una nueva sesión, el autómata pasa a modo de escritura. Las sesiones de ambas pilas podrán ser eliminadas simultáneamente (mediante el borrado de los separadores en ambas pilas) únicamente en modo de borrado y cuando la sesión de la pila auxiliar esté vacía. Cuando se eliminan sesiones, el autómata pasará al modo en que se encontraba antes de crear dichas sesiones.

Por convención, definimos una relación de orden entre modos, de tal modo que se cumple que $\mathbf{w} < \mathbf{e}$.

Para posibilitar el diseño de una técnica de tabulación, estableceremos una restricción adicional según la cual en los modos **w** y **e** en cada sesión de la pila auxiliar se aplicarán las mismas operaciones de apilamiento y extracción, pero en orden inverso. Para ello, cada transición de apilamiento en la pila maestra escribirá una marca de acción que indique la operación realizada sobre la pila auxiliar:

$\nearrow$ para indicar una operación de apilamiento.

$\rightarrow$ para indicar que no se ha realizado modificación alguna sobre una pila auxiliar.

$\searrow$ para indicar que se ha extraído el elemento en la cima de la pila auxiliar.

Las dos marcas de acción siguientes realizarán el papel de separadores de sesiones:

$\models^{\mathbf{w}}$ indica la creación de una nueva sesión a partir de una configuración en modo de escritura.

$\models^{\mathbf{e}}$ indica la creación de una nueva sesión a partir de una configuración en modo de borrado.

Las marcas de acción dirigirán, durante el modo de borrado, las operaciones que se pueden realizar sobre la pila auxiliar cuando se aplica una transición POP sobre la pila maestra. Puesto que las transiciones de apilamiento sobre la pila maestra escriben las marcas en el modo **w**, denominaremos $\otimes$**WRITE** a dichas transiciones. Las transiciones POP de la pila maestra son las encargadas de borrar las marcas en el modo **e**, por lo que recibirán el nombre de transiciones $\otimes$**ERASE**, donde $\otimes \in \{\models^{\mathbf{w}}, \models^{\mathbf{e}}, \nearrow, \rightarrow, \searrow\}$.

Una configuración (m, Ξ, ξ, w) del autómata vendrá determinada por el modo m en que se encuentre, el contenido Ξ de la pila maestra, el contenido ξ de la pila auxiliar y la parte w de la cadena de entrada que resta por leer.

En la nueva versión que acabamos de describir de los autómatas con dos pilas, los movimientos que se pueden realizar en un momento dado están restringidos en gran medida por movimientos realizados en algún momento anterior. Es por ello que recibirán el nombre de *autómatas con dos pilas fuertemente dirigidos* (*Strongly-Driven 2–Stack Automata*, SD–2SA).

Formalmente, definiremos un autómata con dos pilas fuertemente dirigido como una tupla $(V_T, V_S, \$_0, \$_f, V_I, \mathcal{D}, \Theta)$, donde:

- V_T es un conjunto finito de símbolos terminales.
- V_S es un conjunto finito de símbolos de la pila maestra.
- $\$_0 \in V_S$ es el símbolo inicial de la pila maestra.
- $\$_f \in V_S$ es el símbolo final de la pila maestra.
- V_I es un conjunto finito de símbolos de la pila auxiliar.
- $\mathcal{D} = \{\models^{\mathbf{w}}, \models^{\mathbf{e}}, \nearrow, \rightarrow, \searrow\}$ es el conjunto de marcas de acción.
- Θ es un conjunto finito de transiciones de los siguientes tipos:

SWAP1 : Transiciones de la forma $(m, C, \epsilon) \overset{a}{\longmapsto} (m, F, \epsilon)$, donde $m \in \{\mathbf{w}, \mathbf{e}\}$, $C, F \in V_S$ y $a \in V_T \cup \epsilon$. El resultado de aplicar una transición de este tipo a una configuración $(m, \Xi C, \xi, aw)$ es una configuración $(m, \Xi F, \xi, w)$.

SWAP2 : Transiciones de la forma $(\mathbf{w}, C, \models^m) \overset{a}{\longmapsto} (\mathbf{e}, F, \models^m)$. El resultado de aplicar una transición este tipo a una configuración $(m, \Xi C, \xi \models^m, aw)$ es una configuración $(m, \Xi F, \xi \models^m, w)$. Estas transiciones son las únicas que permiten pasar del modo de escritura al modo de borrado.

$\models$**WRITE** : Transiciones de la forma $(m, C, \epsilon) \longmapsto (\mathbf{w}, C \models^m F, \models^m)$ que al ser aplicadas a una configuración $(m, \Xi C, \xi, w)$ producen una configuración $(\mathbf{w}, \Xi C \models^m F, \xi \models^m)$.

$\rightarrow$**WRITE** Transiciones de la forma $(\mathbf{w}, C, \epsilon) \longmapsto (\mathbf{w}, C \rightarrow F, \epsilon)$ que al ser aplicadas a una transición $(\mathbf{w}, \Xi C, \xi, w)$ producen una configuración $(\mathbf{w}, \Xi C \rightarrow F, \xi, w)$.

$\nearrow$**WRITE** Transiciones de la forma $(\mathbf{w}, C, \epsilon) \longmapsto (\mathbf{w}, C \nearrow F, \gamma')$ que al ser aplicadas a una transición $(\mathbf{w}, \Xi C, \xi, w)$ producen una configuración $(\mathbf{w}, \Xi C \nearrow F, \xi \gamma', w)$.

$\searrow$**WRITE** Transiciones de la forma $(\mathbf{w}, C, \gamma) \longmapsto (\mathbf{w}, C \searrow F, \epsilon)$ que al ser aplicadas a una transición $(\mathbf{w}, \Xi C, \xi \gamma, w)$ producen una configuración $(\mathbf{w}, \Xi C \searrow F, \xi, w)$.

$\models$**ERASE** : Transiciones de la forma $(\mathbf{e}, C \models^m F, \models^m) \longmapsto (m, G, \epsilon)$ que al ser aplicadas a una configuración $(m, \Xi C \models^m F, \xi \models^m, w)$ producen una configuración $(m, \Xi G, \xi, w)$.

$\rightarrow$**ERASE** : Transiciones de la forma $(\mathbf{e}, C \rightarrow F, \epsilon) \longmapsto (\mathbf{e}, G, \epsilon)$ que al ser aplicadas a una configuración $(\mathbf{e}, \Xi C \rightarrow F, \xi, w)$ producen una configuración $(\mathbf{e}, \Xi G, \xi, w)$.

$\nearrow$**ERASE** : Transiciones de la forma $(\mathbf{e}, C \nearrow F, \eta') \longmapsto (\mathbf{e}, G, \epsilon)$ que al ser aplicadas a una configuración $(\mathbf{e}, \Xi C \nearrow F, \xi \eta', w)$ producen una configuración $(\mathbf{e}, \Xi G, \xi, w)$.

$\searrow$**ERASE** : Transiciones de la forma $(\mathbf{e}, C \searrow F, \epsilon) \longmapsto (\mathbf{e}, G, \eta)$ que al ser aplicadas a una configuración $(\mathbf{e}, \Xi C \rightarrow F, \xi, w)$ producen una configuración $(\mathbf{e}, \Xi G, \xi \eta, w)$.

Una *configuración* de un autómata con dos pilas fuertemente dirigido es una tupla (m, Ξ, ξ, w), donde $m \in \{\mathbf{w}, \mathbf{e}\}$, $\Xi \in (\mathcal{D} V_S)^*$, $\xi \in (\mathcal{D} V_I^*)^*$ y $w \in V_T^*$.

Una configuración (m, Ξ, ξ, aw) deriva una configuración (m', Ξ', ξ', w), denotado mediante $(m, \Xi, \xi, aw) \vdash (m', \Xi, \xi, w)$, si y sólo si existe una transición que aplicada en modo m transforma la pila maestra Ξ en Ξ' y la pila auxiliar ξ en ξ' al tiempo que lee $a \in V_T \cup \{\epsilon\}$ de la cadena de entrada, mientras el autómata pasa a modo m'. En caso de ser necesario identificar una derivación d concreta, utilizaremos la notación $\vdash_d$. Denotamos por $\overset{*}{\vdash}$ el cierre reflexivo y transitivo de $\vdash$. Decimos que una cadena de entrada w es aceptada por un autómata con dos pilas fuertemente dirigido si

$$(\mathbf{w}, \models^{\mathbf{w}} \$_0, \models^{\mathbf{w}}, w) \overset{*}{\vdash} (\mathbf{e}, \models^{\mathbf{w}} \$_0 \models^{\mathbf{w}} \$_f, \models^{\mathbf{w}} \models^{\mathbf{w}}, \epsilon)$$

El lenguaje aceptado por un autómata con dos pilas fuertemente dirigido es el conjunto

$$\{w \in V_T^* \mid (\mathbf{w}, \models^{\mathbf{w}} \$_0, \models^{\mathbf{w}}, w) \overset{*}{\vdash} (\mathbf{e}, \models^{\mathbf{w}} \$_0 \models^{\mathbf{w}} \$_f, \models^{\mathbf{w}} \models^{\mathbf{w}}, \epsilon)\}$$

Ejemplo 10.1 El autómata con dos pilas fuertemente dirigido de la tabla 10.1 acepta el lenguaje $\{a^n b^n c^n d^n \mid n > 0\}$. En la tabla 10.2 se muestra la derivación para la cadena de entrada *aaabbbcccddd* en dicho autómata. La primera columna indica la transición aplicada, la segunda señala el modo del autómata en ese momento, la tercera muestra el contenido de la pila maestra, la cuarta muestra el contenido de la pila auxiliar y la quinta muestra la parte de la cadena de entrada que resta por leer. ¶

10.3.1 Esquemas de compilación de gramáticas lineales de índices

Para la compilación de las gramáticas lineales de índices en autómatas con dos pilas fuertemente dirigidos utilizaremos la pila maestra para almacenar no-terminales de la gramática y la pila auxiliar para almacenar los índices. Cada sesión corresponderá a una espina en la derivación de la gramática con respecto a la cadena de entrada. Puesto que todos los no-terminales de una espina trabajan sobre la misma pila de índices, en todo momento se cumplirá que el contenido de una sesión de la pila auxiliar se corresponderá con el valor de la pila de índices asociada al no-terminal situado en la cima de la sesión correspondiente en la pila maestra.

Podemos definir un esquema de compilación genérico mediante la parametrización de la información predicha en la fase de llamada y la información propagada en la fase de retorno. Los parámetros a considerar son:

- $\overrightarrow{A}$, que se refiere a la predicción realizada sobre el no-terminal A durante la fase descendente de la estrategia de análisis.
- $\overrightarrow{\gamma}$, que se refiere a la predicción realizada sobre el índice γ.
- $\overleftarrow{A}$, que se refiere a la propagación de información respecto al no-terminal A durante la fase ascendente de la estrategia de análisis.
- $\overleftarrow{\gamma}$, que se refiere a la propagación de información respecto al índice γ.

Esquema de compilación 10.1 El esquema de compilación genérico de una gramática lineal de índices en un autómata con dos pilas fuertemente dirigido queda definido por el conjunto de reglas mostrado en la tabla 10.3 y por los elementos inicial $\$_0$ y final $\overleftarrow{S}$. §

El esquema de compilación genérico se puede convertir en esquemas de compilación que incorporan estrategias específicas. En la tabla 10.4 se muestran los valores que toman los diferentes parámetros para las estrategias de análisis de LIG más comunes. En dicha tabla, $\Box$ indica un símbolo de pila especial que no aparece inicialmente en V_S y $\Diamond$ indica un elemento especial que no aparece inicialmente en V_I.

(a)	$(\mathbf{w}, \$_0, \epsilon) \longmapsto (\mathbf{w}, \$_0 \models^{\mathbf{w}} A, \models^{\mathbf{w}})$
(b)	$(\mathbf{w}, A, \epsilon) \overset{a}{\longmapsto} (\mathbf{w}, A', \epsilon)$
(c)	$(\mathbf{w}, A', \epsilon) \longmapsto (\mathbf{w}, A' \nearrow A, \gamma)$
(d)	$(\mathbf{w}, A', \epsilon) \overset{b}{\longmapsto} (\mathbf{w}, B', \epsilon)$
(e)	$(\mathbf{w}, B', \gamma) \longmapsto (\mathbf{w}, B' \searrow B, \epsilon)$
(f)	$(\mathbf{w}, B, \epsilon) \overset{b}{\longmapsto} (\mathbf{w}, B', \epsilon)$
(g)	$(\mathbf{w}, B', \models^m) \overset{c}{\longmapsto} (\mathbf{e}, C', \models^m)$
(h)	$(\mathbf{e}, B' \searrow C', \epsilon) \longmapsto (\mathbf{e}, C, \eta)$
(i)	$(\mathbf{e}, C, \epsilon) \overset{c}{\longmapsto} (\mathbf{e}, C', \epsilon)$
(j)	$(\mathbf{e}, C', \epsilon) \overset{d}{\longmapsto} (\mathbf{e}, D', \epsilon)$
(k)	$(\mathbf{e}, A' \nearrow D') \longmapsto (\mathbf{e}, D, \epsilon)$
(l)	$(\mathbf{e}, D, \epsilon) \overset{d}{\longmapsto} (\mathbf{e}, D', \epsilon)$
(m)	$(\mathbf{e}, D', \epsilon) \longmapsto (\mathbf{e}, \$_f, \epsilon)$

Tabla 10.1: Transiciones del SD–2SA que acepta $\{a^n b^n c^n d^n \mid n > 0\}$

	$\mathbf{w}$	$\models^{\mathbf{w}}\$_0$	$\models^{\mathbf{w}}$	$aaabbbcccddd$
(a)	$\mathbf{w}$	$\models^{\mathbf{w}}\$_0 \ \models^{\mathbf{w}}A$	$\models^{\mathbf{w}}\models^{\mathbf{w}}$	$aaabbbcccddd$
(b)	$\mathbf{w}$	$\models^{\mathbf{w}}\$_0 \ \models^{\mathbf{w}}A'$	$\models^{\mathbf{w}}\models^{\mathbf{w}}$	$aabbbcccddd$
(c)	$\mathbf{w}$	$\models^{\mathbf{w}}\$_0 \ \models^{\mathbf{w}}A' \nearrow A$	$\models^{\mathbf{w}}\models^{\mathbf{w}} \gamma$	$aabbbcccddd$
(b)	$\mathbf{w}$	$\models^{\mathbf{w}}\$_0 \ \models^{\mathbf{w}}A' \nearrow A'$	$\models^{\mathbf{w}}\models^{\mathbf{w}} \gamma$	$abbbcccddd$
(c)	$\mathbf{w}$	$\models^{\mathbf{w}}\$_0 \ \models^{\mathbf{w}}A' \nearrow A' \nearrow A$	$\models^{\mathbf{w}}\models^{\mathbf{w}} \gamma\,\gamma$	$abbbcccddd$
(b)	$\mathbf{w}$	$\models^{\mathbf{w}}\$_0 \ \models^{\mathbf{w}}A' \nearrow A' \nearrow A'$	$\models^{\mathbf{w}}\models^{\mathbf{w}} \gamma\,\gamma$	$bbbcccddd$
(d)	$\mathbf{w}$	$\models^{\mathbf{w}}\$_0 \ \models^{\mathbf{w}}A' \nearrow A' \nearrow B'$	$\models^{\mathbf{w}}\models^{\mathbf{w}} \gamma\,\gamma$	$bbcccddd$
(e)	$\mathbf{w}$	$\models^{\mathbf{w}}\$_0 \ \models^{\mathbf{w}}A' \nearrow A' \nearrow B' \searrow B$	$\models^{\mathbf{w}}\models^{\mathbf{w}} \gamma$	$bbcccddd$
(f)	$\mathbf{w}$	$\models^{\mathbf{w}}\$_0 \ \models^{\mathbf{w}}A' \nearrow A' \nearrow B' \searrow B'$	$\models^{\mathbf{w}}\models^{\mathbf{w}} \gamma$	$bcccddd$
(e)	$\mathbf{w}$	$\models^{\mathbf{w}}\$_0 \ \models^{\mathbf{w}}A' \nearrow A' \nearrow B' \searrow B' \searrow B$	$\models^{\mathbf{w}}\models^{\mathbf{w}}$	$bcccddd$
(f)	$\mathbf{w}$	$\models^{\mathbf{w}}\$_0 \ \models^{\mathbf{w}}A' \nearrow A' \nearrow B' \searrow B' \searrow B'$	$\models^{\mathbf{w}}\models^{\mathbf{w}}$	$cccddd$
(g)	$\mathbf{e}$	$\models^{\mathbf{w}}\$_0 \ \models^{\mathbf{w}}A' \nearrow A' \nearrow B' \searrow B' \searrow C'$	$\models^{\mathbf{w}}\models^{\mathbf{w}}$	$ccddd$
(h)	$\mathbf{e}$	$\models^{\mathbf{w}}\$_0 \ \models^{\mathbf{w}}A' \nearrow A' \nearrow B' \searrow C$	$\models^{\mathbf{w}}\models^{\mathbf{w}} \eta$	$ccddd$
(i)	$\mathbf{e}$	$\models^{\mathbf{w}}\$_0 \ \models^{\mathbf{w}}A' \nearrow A' \nearrow B' \searrow C'$	$\models^{\mathbf{w}}\models^{\mathbf{w}} \eta$	$cddd$
(h)	$\mathbf{e}$	$\models^{\mathbf{w}}\$_0 \ \models^{\mathbf{w}}A' \nearrow A' \nearrow C$	$\models^{\mathbf{w}}\models^{\mathbf{w}} \eta\,\eta$	$cddd$
(i)	$\mathbf{e}$	$\models^{\mathbf{w}}\$_0 \ \models^{\mathbf{w}}A' \nearrow A' \nearrow C'$	$\models^{\mathbf{w}}\models^{\mathbf{w}} \eta\,\eta$	ddd
(j)	$\mathbf{e}$	$\models^{\mathbf{w}}\$_0 \ \models^{\mathbf{w}}A' \nearrow A' \nearrow D'$	$\models^{\mathbf{w}}\models^{\mathbf{w}} \eta\,\eta$	dd
(k)	$\mathbf{e}$	$\models^{\mathbf{w}}\$_0 \ \models^{\mathbf{w}}A' \nearrow D$	$\models^{\mathbf{w}}\models^{\mathbf{w}} \eta$	dd
(l)	$\mathbf{e}$	$\models^{\mathbf{w}}\$_0 \ \models^{\mathbf{w}}A' \nearrow D'$	$\models^{\mathbf{w}}\models^{\mathbf{w}} \eta$	d
(k)	$\mathbf{e}$	$\models^{\mathbf{w}}\$_0 \ \models^{\mathbf{w}}D$	$\models^{\mathbf{w}}\models^{\mathbf{w}}$	d
(l)	$\mathbf{e}$	$\models^{\mathbf{w}}\$_0 \ \models^{\mathbf{w}}D'$	$\models^{\mathbf{w}}\models^{\mathbf{w}}$	
(m)	$\mathbf{e}$	$\models^{\mathbf{w}}\$_0 \ \models^{\mathbf{w}}\$_f$	$\models^{\mathbf{w}}\models^{\mathbf{w}}$	

Tabla 10.2: Configuraciones del SD–2SA para la cadena de entrada $aaabbbcccddd$

[INIT]	$(\mathbf{w}, \$_0, \epsilon) \longmapsto (\mathbf{w}, \$_0 {\models^{\mathbf{w}}} \nabla_{0,0}, \models^{\mathbf{w}})$	
[CALL]	$(m, \nabla_{r,s}, \epsilon) \longmapsto (\mathbf{w}, \nabla_{r,s} {\models^m} \overrightarrow{A_{r,s+1}}, \models^m)$	$A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\,]\Upsilon_2$
[SCALL-1]	$(\mathbf{w}, \nabla_{r,s}, \epsilon) \longmapsto (\mathbf{w}, \nabla_{r,s} {\to} \overrightarrow{A_{r,s+1}}, \epsilon)$	$A_{r,0}[\circ\circ] \to \Upsilon_1 A_{r,s+1}[\circ\circ]\Upsilon_2$
[SCALL-2]	$(\mathbf{w}, \nabla_{r,s}, \epsilon) \longmapsto (\mathbf{w}, \nabla_{r,s} {\nearrow} \overrightarrow{A_{r,s+1}}, \overrightarrow{\gamma'})$	$A_{r,0}[\circ\circ] \to \Upsilon_1 A_{r,s+1}[\circ\circ\gamma']\Upsilon_2$
[SCALL-3]	$(\mathbf{w}, \nabla_{r,s}, \overrightarrow{\gamma}) \longmapsto (\mathbf{w}, \nabla_{r,s} {\searrow} \overrightarrow{A_{r,s+1}}, \epsilon)$	$A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\circ\circ]\Upsilon_2$
[SEL]	$(\mathbf{w}, \overrightarrow{A_{r,0}}, \epsilon) \longmapsto (\mathbf{w}, \nabla_{r,0}, \epsilon)$	$r \neq 0$
[PUB]	$(\mathbf{e}, \nabla_{r,n_r}, \epsilon) \longmapsto (\mathbf{e}, \overleftarrow{A_{r,0}}, \epsilon)$	
[RET]	$(\mathbf{e}, \nabla_{r,s} {\models^m} \overleftarrow{A_{r,s+1}}, \models^m) \longmapsto (m, \nabla_{r,s+1}, \epsilon)$	$A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\,]\Upsilon_2$
[SRET-1]	$(\mathbf{e}, \nabla_{r,s} {\to} \overleftarrow{A_{r,s+1}}, \epsilon) \longmapsto (\mathbf{e}, \nabla_{r,s+1}, \epsilon)$	$A_{r,0}[\circ\circ] \to \Upsilon_1 A_{r,s+1}[\circ\circ]\Upsilon_2$
[SRET-2]	$(\mathbf{e}, \nabla_{r,s} {\nearrow} \overleftarrow{A_{r,s+1}}, \overleftarrow{\gamma'}) \longmapsto (\mathbf{e}, \nabla_{r,s+1}, \epsilon)$	$A_{r,0}[\circ\circ] \to \Upsilon_1 A_{r,s+1}[\circ\circ\gamma']\Upsilon_2$
[SRET-3]	$(\mathbf{e}, \nabla_{r,s} {\searrow} \overleftarrow{A_{r,s+1}}, \epsilon) \longmapsto (\mathbf{e}, \nabla_{r,s+1}, \overleftarrow{\gamma})$	$A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\circ\circ]\Upsilon_2$
[SCAN]	$(\mathbf{w}, \overrightarrow{A_{r,0}}, \models^m) \overset{a}{\longmapsto} (\mathbf{e}, \overleftarrow{A_{r,0}}, \models^m)$	$A_{r,0}[\,] \to a$

Tabla 10.3: Reglas del esquema de compilación genérico de LIG en SD–2SA

estrategia-CF	estrategia-índices	$\overrightarrow{A_{r,s+1}}$	$\overrightarrow{\gamma}$	$\overleftarrow{A_{r,s+1}}$	$\overleftarrow{\gamma}$
Ascendente	ascendente	$\Box$	$\Diamond$	$A_{r,s+1}$	γ
Earley		$\overline{A_{r,s+1}}$	$\Diamond$	$\overline{\overline{A_{r,s+1}}}$	γ
Descendente		$A_{r,s+1}$	$\Diamond$	$\Box$	γ
Ascendente	Earley	$\Box$	γ	$A_{r,s+1}$	γ
Earley		$\overline{A_{r,s+1}}$	γ	$\overline{\overline{A_{r,s+1}}}$	γ
Descendente		$A_{r,s+1}$	γ	$\Box$	γ
Ascendente	descendente	$\Box$	γ	$A_{r,s+1}$	$\Diamond$
Earley		$\overline{A_{r,s+1}}$	γ	$\overline{\overline{A_{r,s+1}}}$	$\Diamond$
Descendente		$A_{r,s+1}$	γ	$\Box$	$\Diamond$

Tabla 10.4: Parámetros del esquema de compilación genérico de LIG en SD–2SA

10.3.2 Esquemas de compilación de gramáticas de adjunción de árboles

Para la compilación de gramáticas de adjunción de árboles en autómatas con dos pilas fuertemente dirigidos haremos uso de la pila maestra para almacenar los nodos de los árboles elementales según se van visitando. La pila auxiliar se utilizará para almacenar la pila de adjunciones pendiente en cada nodo, de tal modo que los valores almacenados en una sesión de la pila auxiliar se corresponden con la pila de adjunciones pendientes del nodo situado en la cima de la sesión correspondiente en la pila maestra.

Podemos definir un esquema de compilación genérico mediante la parametrización del flujo de información de las fases de llamada y retorno. Los parámetros a considerar son:

- $\overrightarrow{N^{\gamma}_{r,s}}$, la información predicha acerca del nodo $N^{\gamma}_{r,s}$.
- $\overleftarrow{N^{\gamma}_{r,s}}$, la información propagada acerca del nodo $N^{\gamma}_{r,s}$.

Esquema de compilación 10.2 El esquema de compilación genérico de una gramática de adjunción de árboles en un autómata con dos pilas fuertemente dirigido queda definido por el conjunto de reglas mostrado en la tabla 10.5 y los elementos inicial $\$_0$ y final $\overleftarrow{\top^{\alpha}}$, con $\alpha \in \boldsymbol{I}$.

En este esquema de análisis sintáctico el cambio de modo de escritura a modo de borrado se produce, además de en las transiciones producidas por la regla de compilación [**SCAN**], en las transiciones producidas por la regla de compilación [**PUB2**]. Ello se debe a que en las producciones de los árboles iniciales no son aplicables las reglas [**SCALL**] y [**SRET**], por lo cual todo cambio de modo realizado por una regla [**SCAN**] es eliminado por una regla [**RET**].

§

El esquema de compilación genérico puede convertirse en esquemas de compilación para diferentes estrategias de análisis según los valores que tomen los parámetros con respecto a su ubicación en las pilas maestra y auxiliar, tal y como se indica en la tabla 10.6.

10.3.3 SD–2SA y los lenguajes de adjunción de árboles

Los lenguajes aceptados por los autómatas con dos pilas fuertemente dirigidos coinciden con los lenguajes de adjunción de árboles. Para demostrar esta aseveración definiremos y demostraremos los dos teoremas siguientes.

Teorema 10.3 *Los lenguajes adjunción de árboles son un subconjunto de los lenguajes aceptados por la clase de los autómatas con dos pilas fuertemente dirigidos.*

<u>Demostración</u>:

Por el esquema de compilación de TAG en SD–2SA presentado anteriormente, a partir de cualquier gramática de adjunción de árboles es posible construir un SD–2SA que acepta el lenguaje reconocido por dicha gramática. Análogamente, por el esquema de compilación de LIG en SD–2SA, a partir de cualquier gramática lineal de índices es posible construir un SD–2SA que acepta el lenguaje reconocido por dicha gramática. □

Teorema 10.4 *La clase de los lenguajes aceptados por los autómatas con dos pilas fuertemente dirigidos es un subconjunto de los lenguajes de adjunción de árboles.*

$$
\begin{array}{llll}
\textbf{[INIT]} & (\mathbf{w}, \$_0, \epsilon) \longmapsto (\mathbf{w}, \$_0 \models^{\mathbf{w}} \nabla^{\alpha}_{0,0}, \models^{\mathbf{w}}) & & \alpha \in \boldsymbol{I} \\
\textbf{[CALL]} & (m, \nabla^{\gamma}_{r,s}, \epsilon) \longmapsto (\mathbf{w}, \nabla^{\gamma}_{r,s} \models^{m} \overrightarrow{N^{\gamma}_{r,s+1}}, \models^{m}) & & N^{\gamma}_{r,s+1} \not\in \text{espina}(\gamma),\ \mathbf{nil} \in \text{adj}(N_{r,s+1}) \\
\textbf{[SCALL]} & (\mathbf{w}, \nabla^{\gamma}_{r,s}, \epsilon) \longmapsto (\mathbf{w}, \nabla^{\gamma}_{r,s} \rightarrow \overrightarrow{N^{\gamma}_{r,s+1}}, \epsilon) & & N^{\gamma}_{r,s+1} \in \text{espina}(\gamma),\ \mathbf{nil} \in \text{adj}(N_{r,s+1}) \\
\textbf{[SEL]} & (\mathbf{w}, \overrightarrow{N^{\gamma}_{r,0}}, \epsilon) \longmapsto (\mathbf{w}, \nabla^{\gamma}_{r,0}, \epsilon) & & r \neq 0 \\
\textbf{[PUB1]} & (\mathbf{e}, \nabla^{\gamma}_{r,n_r}, \epsilon) \longmapsto (\mathbf{e}, \overleftarrow{N^{\gamma}_{r,0}}, \epsilon) & & \\
\textbf{[PUB2]} & (\mathbf{w}, \nabla^{\gamma}_{r,n_r}, \models^{m}) \longmapsto (\mathbf{e}, \overleftarrow{N^{\gamma}_{r,0}}, \models^{m}) & & \\
\textbf{[RET]} & (\mathbf{e}, \nabla^{\gamma}_{r,s} \models^{m} \overleftarrow{N^{\gamma}_{r,s+1}}, \models^{m}) \longmapsto (m, \nabla^{\gamma}_{r,s+1}, \epsilon) & & N^{\gamma}_{r,s+1} \not\in \text{espina}(\gamma),\ \mathbf{nil} \in \text{adj}(N_{r,s+1}) \\
\textbf{[SRET]} & (\mathbf{e}, \nabla^{\gamma}_{r,s} \rightarrow \overleftarrow{N^{\gamma}_{r,s+1}}, \epsilon) \longmapsto (\mathbf{e}, \nabla^{\gamma}_{r,s+1}, \epsilon) & & N^{\gamma}_{r,s+1} \in \text{espina}(\gamma),\ \mathbf{nil} \in \text{adj}(N_{r,s+1}) \\
\textbf{[SCAN]} & (\mathbf{w}, \overrightarrow{N^{\gamma}_{r,0}}, \models^{m}) \stackrel{a}{\longmapsto} (\mathbf{e}, \overleftarrow{N^{\gamma}_{r,0}}, \models^{m}) & & N^{\gamma}_{r,0}[\,] \rightarrow a \\
\textbf{[ACALL]} & (\mathbf{w}, \nabla^{\gamma}_{r,s}, \epsilon) \longmapsto (\mathbf{w}, \nabla^{\gamma}_{r,s} \nearrow \overrightarrow{\top^{\beta}}, \overrightarrow{N^{\gamma}_{r,s+1}}) & & \beta \in \text{adj}(N^{\gamma}_{r,s+1}) \\
\textbf{[ARET]} & (\mathbf{e}, \nabla^{\gamma}_{r,s} \nearrow \overleftarrow{\top^{\beta}}, \overleftarrow{N^{\gamma}_{r,s+1}}) \longmapsto (\mathbf{e}, \nabla^{\gamma}_{r,s+1}, \epsilon) & & \beta \in \text{adj}(N^{\gamma}_{r,s+1}) \\
\textbf{[FCALL]} & (\mathbf{w}, \nabla^{\beta}_{f,0}, \overrightarrow{N^{\gamma}_{r,s+1}}) \longmapsto (\mathbf{w}, \nabla^{\beta}_{f,0} \searrow \overrightarrow{N^{\gamma}_{r,s+1}}, \epsilon) & & N^{\beta}_{f,0} = \mathbf{F}^{\beta},\ \beta \in \text{adj}(N^{\gamma}_{r,s+1}) \\
\textbf{[FRET]} & (\mathbf{w}, \nabla^{\beta}_{f,0} \searrow \overleftarrow{N^{\gamma}_{r,s+1}}, \epsilon) \longmapsto (\mathbf{e}, \nabla^{\beta}_{f,1}, \overleftarrow{N^{\gamma}_{r,s+1}}) & & N^{\beta}_{f,0} = \mathbf{F}^{\beta},\ \beta \in \text{adj}(N^{\gamma}_{t,s+1})
\end{array}
$$

Tabla 10.5: Reglas del esquema de compilación genérico de TAG en SD–2SA

Estrategia-CF	**Estrategia-adjunción**	**MS** $\overrightarrow{N^{\gamma}_{r,s+1}}$	**AS** $\overrightarrow{N^{\gamma}_{r,s+1}}$	**MS** $\overleftarrow{N^{\gamma}_{r,s+1}}$	**AS** $\overleftarrow{N^{\gamma}_{r,s+1}}$
Ascendente	ascendente	$\Box$	$\Diamond$	$N^{\gamma}_{r,s+1}$	$N^{\gamma}_{r,s+1}$
Earley	ascendente	$\overline{N^{\gamma}_{r,s+1}}$	$\Diamond$	$\overline{\overline{N^{\gamma}_{r,s+1}}}$	$N^{\gamma}_{r,s+1}$
Descendente	ascendente	$N^{\gamma}_{r,s+1}$	$\Diamond$	$\Box$	$N^{\gamma}_{r,s+1}$
Ascendente	Earley	$\Box$	$N^{\gamma}_{r,s+1}$	$N^{\gamma}_{r,s+1}$	$N^{\gamma}_{r,s+1}$
Earley	Earley	$\overline{N^{\gamma}_{r,s+1}}$	$N^{\gamma}_{r,s+1}$	$\overline{\overline{N^{\gamma}_{r,s+1}}}$	$N^{\gamma}_{r,s+1}$
Descendente	Earley	$N^{\gamma}_{r,s+1}$	$N^{\gamma}_{r,s+1}$	$\Box$	$N^{\gamma}_{r,s+1}$
Ascendente	descendente	$\Box$	$N^{\gamma}_{r,s+1}$	$N^{\gamma}_{r,s+1}$	$\Diamond$
Earley	descendente	$\overline{N^{\gamma}_{r,s+1}}$	$N^{\gamma}_{r,s+1}$	$\overline{\overline{N^{\gamma}_{r,s+1}}}$	$\Diamond$
Descendente	descendente	$N^{\gamma}_{r,s+1}$	$N^{\gamma}_{r,s+1}$	$\Box$	$\Diamond$

Tabla 10.6: Parámetros del esquema de compilación genérico de TAG en SD–2SA

Demostración:

Mostraremos que para todo SD–2SA existe una gramática lineal de índices tal que el lenguaje reconocido por la gramática coincide con el lenguaje aceptado por el autómata.

Sea $\mathcal{A} = (V_T, V_S, \$_0, \$_f, V_I, \mathcal{D}, \Theta)$ un autómata lineal de índices fuertemente dirigido. Construiremos una gramática lineal de índices $\mathcal{L} = (V_T, V_N, V_I', S, P)$. El conjunto V_N de no-terminales estará formado por pares $\langle E, B\rangle$ tal que $A, B \in V_S^{\mathcal{D}}$, donde $V_S^{\mathcal{D}} = \{E^m \mid E \in V_S,\ m \in \mathcal{D}\}$. El conjunto V_I' estará formado por pares $\langle \gamma, \eta\rangle$ tal que $\gamma, \eta \in V_I$. Para que $\mathcal{L}$ reconozca el lenguaje aceptado por $\mathcal{A}$ el conjunto de producciones en P ha de construirse a partir de las transiciones en Θ de la siguiente manera:

- Para toda transición $(m, C, \epsilon) \overset{a}{\longmapsto} (m, F, \epsilon)$ y para todo $E^{m'} \in V_S^{\mathcal{D}}$ tal que $m' \leq m$ creamos una producción

$$\langle E^{m'}, F^m\rangle[\circ\circ] \to \langle E^{m'}, C^m\rangle[\circ\circ]\ a$$

- Para toda transición $(\mathbf{w}, C, \models^m) \overset{a}{\longmapsto} (\mathbf{e}, F, \models^m)$ y para todo $E^{\mathbf{w}} \in V_S^{\mathcal{D}}$ creamos la producción

$$\langle E^{\mathbf{w}}, F^{\mathbf{e}}\rangle[\,] \to \langle E^{\mathbf{w}}, C^{\mathbf{w}}\rangle[\,]\ a$$

- Para todo par de transiciones $(\mathbf{e}, C{\models^m}F, \models^m) \longmapsto (m, G, \epsilon)$ y $(m, C, \epsilon) \longmapsto (\mathbf{w}, C{\models^m}F', \models^m)$, y para todo $E^{m''} \in V_S^{\mathcal{D}}$ tal que $m'' \leq m$ creamos una producción

$$\langle E^{m''}, G^m\rangle[\circ\circ] \to \langle E^{m''}, C^m\rangle[\circ\circ]\ \ \langle F'^{\mathbf{w}}, F^{\mathbf{e}}\rangle[\,]$$

- Para todo par de transiciones $(\mathbf{e}, C{\to}F, \epsilon) \longmapsto (\mathbf{e}, G, \epsilon)$ y $(\mathbf{w}, C, \epsilon) \longmapsto (\mathbf{w}, C{\to}F', \epsilon)$, y para todo $E^{\mathbf{w}} \in V_S^{\mathcal{D}}$ creamos una producción

$$\langle E^{\mathbf{w}}, G^{\mathbf{e}}\rangle[\circ\circ] \to \langle E^{\mathbf{w}}, C^{\mathbf{w}}\rangle[\,]\ \ \langle F'^{\mathbf{w}}, F^{\mathbf{e}}\rangle[\circ\circ]$$

- Para todo par de transiciones $(\mathbf{e}, C{\nearrow}F, \eta') \longmapsto (\mathbf{e}, G, \epsilon)$ y $(\mathbf{w}, C, \epsilon) \longmapsto (\mathbf{w}, C{\nearrow}F', \gamma')$, y para todo $E^{\mathbf{w}} \in V_S^{\mathcal{D}}$ creamos una producción

$$\langle E^{\mathbf{w}}, G^{\mathbf{e}}\rangle[\circ\circ] \to \langle E^{\mathbf{w}}, C^{\mathbf{w}}\rangle[\,]\ \ \langle F'^{\mathbf{w}}, F^{\mathbf{e}}\rangle[\circ\circ\langle\gamma', \eta'\rangle]$$

- Para todo par de transiciones $(\mathbf{e}, C{\searrow}F, \epsilon) \longmapsto (\mathbf{e}, G, \eta)$ y $(\mathbf{w}, C, \gamma) \longmapsto (\mathbf{w}, C{\searrow}F', \epsilon)$, y para todo $E^{\mathbf{w}} \in V_S^{\mathcal{D}}$ creamos una producción

$$\langle E^{\mathbf{w}}, G^{\mathbf{e}}\rangle[\circ\circ\langle\gamma, \eta\rangle] \to \langle E^{\mathbf{w}}, C^{\mathbf{w}}\rangle[\,]\ \ \langle F'^{\mathbf{w}}, F^{\mathbf{e}}\rangle[\circ\circ]$$

- Para todo $E^m \in V_S^{\mathcal{D}}$ creamos una producción

$$\langle E^m, E^m\rangle[\,] \to \epsilon$$

- Para toda transición $(\mathbf{w}, \$_0, \epsilon) \longmapsto (\mathbf{w}, \$_0{\models^m}F, \models^m)$, donde $F \in V_S - \{\$_0\}$, creamos una producción

$$\langle \$_0^{\mathbf{w}}, \$_0^{\mathbf{w}}\rangle[\circ\circ] \to \langle F^{\mathbf{w}}, \$_f^{\mathbf{e}}\rangle[\circ\circ]$$

Con respecto al axioma de la gramática, tenemos que $S = \langle \$_0^{\mathbf{w}}, \$_0^{\mathbf{w}}\rangle$.

Dividiremos la demostración en tres casos, cada uno correspondiente a un tipo específico de derivación. Tratamos a continuación cada uno de los casos por separado.

Caso 1. Existe una derivación en el autómata $(\mathbf{w}, E, \xi_1, w) \overset{*}{\vdash} (\mathbf{w}, C, \xi_1, \epsilon)$ para algún $\xi_1 \in V_I^*$, en la que sólo se han aplicado transiciones de tipo **SWAP1**, si y sólo si existe una derivación en la gramática $\langle E^{\mathbf{w}}, C^{\mathbf{w}}\rangle[\,] \overset{*}{\Rightarrow} w$. La demostración se realiza por inducción. El caso base lo constituyen los dos casos siguientes:

- Si $(\mathbf{w}, E, \xi_1, \epsilon) \overset{0}{\vdash} (\mathbf{w}, E, \xi_1, \epsilon)$ entonces existe una producción $\langle E^{\mathbf{w}}, E^{\mathbf{w}}\rangle[\,] \rightarrow \epsilon$, por lo que $\langle E^{\mathbf{w}}, E^{\mathbf{w}}\rangle[\,] \overset{*}{\Rightarrow} \epsilon$.
- Si $\langle E^{\mathbf{w}}, E^{\mathbf{w}}\rangle[\,] \rightarrow \epsilon$ entonces existe una derivación $(\mathbf{w}, E, \xi_1, \epsilon) \overset{0}{\vdash} (\mathbf{w}, E, \xi_1, \epsilon)$ en el autómata.

Por hipótesis de inducción suponemos que se cumple para toda derivación de longitud inferior a s. El paso de inducción contempla los dos casos siguientes:

- Si $(\mathbf{w}, E, \xi_1, wa) \overset{s}{\vdash} (\mathbf{w}, C, \xi_1, a) \vdash (\mathbf{w}, F, \xi_1, \epsilon)$, entonces existe una producción $\langle E^{\mathbf{w}}, F^{\mathbf{w}}\rangle[\circ\circ] \rightarrow \langle E^{\mathbf{w}}, C^{\mathbf{w}}\rangle[\circ\circ]\, a$, por hipótesis de inducción tenemos que $\langle E^{\mathbf{w}}, C^{\mathbf{w}}\rangle[\,] \overset{*}{\Rightarrow} w$, por lo que $\langle E^{\mathbf{w}}, F^{\mathbf{w}}\rangle[\,] \overset{*}{\Rightarrow} wa$.
- Si $\langle E^{\mathbf{w}}, F^{\mathbf{w}}\rangle[\,] \Rightarrow \langle E^{\mathbf{w}}, C^{\mathbf{w}}\rangle[\,]\, a \overset{s}{\Rightarrow} wa$, entonces existe una transición $(\mathbf{w}, C, \epsilon) \overset{a}{\longmapsto} (\mathbf{w}, F, \epsilon)$, por hipótesis de inducción $(\mathbf{w}, E, \xi_1, w) \overset{s}{\vdash} (\mathbf{w}, C, \xi_1, \epsilon)$ para algún ξ_1 y en consecuencia $(\mathbf{w}, E, \xi_1, wa) \overset{s}{\vdash} (\mathbf{w}, F, \xi_1, \epsilon)$.

Caso 2. Existe una derivaci'on del autómata

$$\begin{aligned}
(\mathbf{w}, \Xi\, E, \xi_1, w_1 w_2) & \overset{*}{\vdash} (\mathbf{w}, \Xi\, C, \xi_1, w_2) \\
& \vdash (\mathbf{w}, \Xi\, C \models^{\mathbf{w}} F', \xi_1 \models^{\mathbf{w}}, w_2) \\
& \overset{*}{\vdash} (\mathbf{e}, \Xi\, C \models^{\mathbf{w}} F, \xi_1 \models^{\mathbf{w}}, \epsilon) \\
& \vdash (\mathbf{w}, \Xi\, G, \xi_1, \epsilon)
\end{aligned}$$

si y sólo si $\langle E^{\mathbf{w}}, G^{\mathbf{w}}\rangle[\,] \overset{*}{\Rightarrow} w_1 w_2$. La demostración se obtiene directamente a partir del caso 1 y de la existencia de una producción $\langle E^{\mathbf{w}}, G^{\mathbf{w}}\rangle[\circ\circ] \rightarrow \langle E^{\mathbf{w}}, C^{\mathbf{w}}\rangle[\circ\circ]\, \langle F'^{\mathbf{w}}, F^{\mathbf{e}}\rangle[\,]$.

Caso 3. El caso principal de esta demostración establece que $\langle E^{\mathbf{w}}, C^{\mathbf{e}}\rangle[\alpha] \overset{*}{\Rightarrow} w$ si y sólo si $(\mathbf{w}, E, \xi, w) \overset{*}{\vdash} (\mathbf{e}, C, \phi, \epsilon)$ y se cumple que $\xi = \gamma_1\gamma_2 \ldots \gamma_p$, $\phi = \eta_1\eta_2 \ldots \eta_p$ y $\alpha = \langle\gamma_1, \eta_1\rangle\langle\gamma_2, \eta_2\rangle \ldots \langle\gamma_p, \eta_p\rangle$, esto es, ξ es la pila obtenida como resultado de la proyección del primer componente de los elementos almacenados en α mientras que ϕ es la pila obtenida como resultado de la proyección del segundo componente de los elementos almacenados en la pila de índices α. Tratamos a continuación de demostrar cada una de las direcciones de la implicación:

- Si una derivación $(\mathbf{w}, E, \xi, w) \overset{*}{\vdash} (\mathbf{e}, C, \phi, \epsilon)$ es el resultado de aplicar la secuencia $t_1, \ldots, t_s$ de transiciones en Θ, entonces existe una secuencia $p_1, \ldots, p'_s$ de producciones en P tal que la derivación $\langle E^{\mathbf{w}}, C^{\mathbf{e}}\rangle[\alpha] \overset{*}{\Rightarrow} w$ resultado de aplicar $p_1, \ldots, p'_s$ reconoce w. La demostración se realiza por inducción en la longitud de la derivación del autómata. El caso base lo constituye la derivación $(\mathbf{w}, E, \models^m, \epsilon) \vdash (\mathbf{e}, C, \models^m, \epsilon)$, para la que existe la producción $\langle E^{\mathbf{w}}, C^{\mathbf{e}}\rangle[\,] \rightarrow \langle E^{\mathbf{w}}, E^{\mathbf{w}}\rangle[\,]\, a$ y la producción $\langle E^{\mathbf{w}}, E^{\mathbf{w}}\rangle[\,] \rightarrow \epsilon$, por lo que $\langle E^{\mathbf{w}}, C^{\mathbf{e}}\rangle[\,] \overset{*}{\Rightarrow} a$.

  Por hipótesis de inducción suponemos que la proposición se cumple para cualquier derivación del autómata de longitud s. En tal caso, durante el paso de inducción verificamos que se cumple para cualquier posible derivación de longitud mayor que s:

  - Si $(\mathbf{w}, E, \xi, wa) \overset{s}{\vdash} (\mathbf{e}, C, \phi, a) \vdash (\mathbf{e}, F, \phi, \epsilon)$, existe una producción $\langle E^{\mathbf{w}}, F^{\mathbf{e}}\rangle[\circ\circ] \rightarrow \langle E^{\mathbf{w}}, C^{\mathbf{e}}\rangle[\circ\circ]\, a$, por hipótesis de inducción se cumple que $\langle E^{\mathbf{w}}, C^{\mathbf{e}}\rangle[\alpha] \overset{*}{\Rightarrow} w$, y en consecuencia $\langle E^{\mathbf{w}}, F^{\mathbf{e}}\rangle[\alpha] \overset{*}{\Rightarrow} wa$.
  - Si $(\mathbf{w}, E, \xi, w_1 w_2) \overset{s_1}{\vdash} (\mathbf{e}, C, \phi, w_2) \vdash (\mathbf{w}, C \models^{\mathbf{e}} F', \phi \models^{\mathbf{e}}, w_2) \overset{s_2}{\vdash} (\mathbf{e}, C \models^{\mathbf{e}} F, \phi \models^{\mathbf{e}}, \epsilon) \vdash (\mathbf{e}, G, \phi, \epsilon)$, existe una producción $\langle E^{\mathbf{w}}, G^{\mathbf{e}}\rangle[\circ\circ] \rightarrow \langle E^{\mathbf{w}}, C^{\mathbf{e}}\rangle[\circ\circ]\, \langle F'^{\mathbf{w}}, F^{\mathbf{e}}\rangle[\,]$, por hipótesis de inducción se cumple que $\langle E^{\mathbf{w}}, C^{\mathbf{e}}\rangle[\alpha] \overset{*}{\Rightarrow} w_1$ y $\langle F'^{\mathbf{w}}, F^{\mathbf{e}}\rangle[\,] \overset{*}{\Rightarrow} w_2$, y en consecuencia $\langle E^{\mathbf{w}}, G^{\mathbf{e}}\rangle[\alpha] \overset{*}{\Rightarrow} w_1 w_2$.

  - Si $(\mathbf{w},E,\xi,w_1w_2) \overset{s_1}{\vdash} (\mathbf{w},C,\xi,w_2) \vdash (\mathbf{w},C\!\rightarrow\! F',\phi,w_2) \overset{s_2}{\vdash} (\mathbf{e},C\!\rightarrow\! F,\phi,\epsilon) \vdash (\mathbf{e},G,\phi,\epsilon)$, existe una producción $\langle E^{\mathbf{w}},G^{\mathbf{e}}\rangle[\circ\circ] \rightarrow \langle E^{\mathbf{w}},C^{\mathbf{w}}\rangle[\,]\ \langle F'^{\mathbf{w}},F^{\mathbf{e}}\rangle[\circ\circ]$, por los casos 1 y 2 se cumple que $\langle E^{\mathbf{w}},C^{\mathbf{w}}\rangle[\,] \overset{*}{\Rightarrow} w_1$ y $\langle F'^{\mathbf{w}},F^{\mathbf{e}}\rangle[\,] \overset{*}{\Rightarrow} w_2$, y en consecuencia $\langle E^{\mathbf{w}},G^{\mathbf{e}}\rangle[\alpha] \overset{*}{\Rightarrow} w_1w_2$.
  - Si $(\mathbf{w},E,\xi,w_1w_2) \overset{s_1}{\vdash} (\mathbf{w},C,\xi,w_2) \vdash (\mathbf{w},C\nearrow F',\phi\gamma',w_2) \overset{s_2}{\vdash} (\mathbf{e},C\nearrow F,\phi\eta',\epsilon) \vdash (\mathbf{e},G,\phi,\epsilon)$, existe una producción $\langle E^{\mathbf{w}},G^{\mathbf{e}}\rangle[\circ\circ] \rightarrow \langle E^{\mathbf{w}},C^{\mathbf{w}}\rangle[\,]\ \langle F'^{\mathbf{w}},F^{\mathbf{e}}\rangle[\circ\circ\langle\gamma',\eta'\rangle]$, por los casos 1 y 2 se cumple que $\langle E^{\mathbf{w}},C^{\mathbf{w}}\rangle[\,] \overset{*}{\Rightarrow} w_1$ y $\langle F'^{\mathbf{w}},F^{\mathbf{e}}\rangle[\alpha\langle\gamma',\eta\rangle] \overset{*}{\Rightarrow} w_2$, y en consecuencia $\langle E^{\mathbf{w}},G^{\mathbf{e}}\rangle[\alpha] \overset{*}{\Rightarrow} w_1w_2$.
  - Si $(\mathbf{w},E,\xi\gamma,w_1w_2) \overset{s_1}{\vdash} (\mathbf{w},C,\xi,w_2) \vdash (\mathbf{w},C\searrow F',\phi,w_2) \overset{s_2}{\vdash} (\mathbf{e},C\searrow F,\phi,\epsilon) \vdash (\mathbf{e},G,\phi\eta,\epsilon)$, existe una producción $\langle E^{\mathbf{w}},G^{\mathbf{e}}\rangle[\circ\circ\langle\gamma,\eta\rangle] \rightarrow \langle E^{\mathbf{w}},C^{\mathbf{w}}\rangle[\,]\ \langle F'^{\mathbf{w}},F^{\mathbf{e}}\rangle[\circ\circ]$, por los casos 1 y 2 se cumple que $\langle E^{\mathbf{w}},C^{\mathbf{w}}\rangle[\,] \overset{*}{\Rightarrow} w_1$ y $\langle F'^{\mathbf{w}},F^{\mathbf{e}}\rangle[\alpha] \overset{*}{\Rightarrow} w_2$, y en consecuencia $\langle E^{\mathbf{w}},G^{\mathbf{e}}\rangle[\alpha\langle\gamma',\eta\rangle] \overset{*}{\Rightarrow} w_1w_2$.

- Si una derivación izquierda $\langle C^{\mathbf{w}},E^{\mathbf{e}}\rangle[\alpha] \overset{*}{\Rightarrow} w$ reconoce la cadena w como resultado de aplicar la secuencia $p_1,\ldots,p_s$ de producciones en P, entonces existe una secuencia de transiciones $t_1,\ldots,t_s$ en Θ tal que la derivación $(\mathbf{w},C,\xi,w) \overset{*}{\vdash} (\mathbf{e},E,\phi,\epsilon)$ es el resultado de aplicar la secuencia de transiciones $t_1,\ldots,t_s$.

  La demostración se realiza por inducción en la longitud de las derivaciones de la gramática.

  El caso base de la demostración lo constituye la derivación $\langle E^{\mathbf{w}},C^{\mathbf{e}}\rangle[\,] \Rightarrow \langle E^{\mathbf{w}},E^{\mathbf{w}}\rangle[\,]\ a \Rightarrow a$, para la que existe una derivación $(\mathbf{w},E,\models^m,a) \vdash (\mathbf{e},C,\models^m,\epsilon)$ en el autómata.

  Por hipótesis de inducción suponemos que la proposición se cumple para cualquier derivación de la gramática de longitud s. En tal caso, durante el paso de inducción verificamos que se cumple para cualquier posible derivación de longitud mayor que s:

  - Si $\langle E^{\mathbf{w}},F^{\mathbf{e}}\rangle[\alpha] \Rightarrow \langle E^{\mathbf{w}},C^{\mathbf{e}}\rangle[\alpha]\ a \overset{s}{\Rightarrow} wa)$, existe una transición $(\mathbf{e},C,\epsilon) \overset{a}{\longmapsto} (\mathbf{e},F,\epsilon)$, por hipótesis de inducción se cumple que $(\mathbf{w},E,\xi,w) \overset{*}{\vdash} (\mathbf{e},C,\phi,\epsilon)$ y por consiguiente $(\mathbf{w},E,\xi,wa) \overset{*}{\vdash} (\mathbf{e},F,\phi,\epsilon)$.
  - Si $\langle E^{\mathbf{w}},G^{\mathbf{e}}\rangle[\alpha] \Rightarrow \langle E^{\mathbf{w}},C^{\mathbf{e}}\rangle[\alpha]\ \langle F'^{\mathbf{w}},F^{\mathbf{e}}\rangle[\,] \overset{s_1}{\Rightarrow} w_1\ \langle F'^{\mathbf{w}},F^{\mathbf{e}}\rangle[\,] \overset{s_2}{\Rightarrow} w_1w_2$, existe una transición $(\mathbf{e},C,\epsilon) \longmapsto (\mathbf{w},C{\models^{\mathbf{e}}}F',\models^{\mathbf{e}})$ y una transición $(C{\models^{\mathbf{e}}}F,\models^{\mathbf{e}}) \longmapsto (\mathbf{e},G,\epsilon)$, por hipótesis de inducción se cumple que $(\mathbf{w},E,\xi,w_1) \overset{*}{\vdash} (\mathbf{e},C,\phi,\epsilon)$ y que $(\mathbf{w},F',\models^{\mathbf{e}},w_2) \overset{*}{\vdash} (\mathbf{e},F,\models^{\mathbf{e}},\epsilon)$ y en consecuencia se cumple que $(\mathbf{w},E,\xi,w_1w_2) \overset{*}{\vdash} (\mathbf{e},C,\phi,w_2) \vdash (\mathbf{w},C{\models^{\mathbf{e}}}F',\phi{\models^{\mathbf{e}}},w_2) \overset{*}{\vdash} (\mathbf{e},C{\models^{\mathbf{e}}}F,\phi{\models^{\mathbf{e}}},\epsilon) \vdash (\mathbf{e},G,\phi,\epsilon)$.
  - Si $\langle E^{\mathbf{w}},G^{\mathbf{e}}\rangle[\alpha] \Rightarrow \langle E^{\mathbf{w}},C^{\mathbf{w}}\rangle[\,]\ \langle F'^{\mathbf{w}},F^{\mathbf{e}}\rangle[\alpha] \overset{s_1}{\Rightarrow} w_1\ \langle F'^{\mathbf{w}},F^{\mathbf{e}}\rangle[\alpha] \overset{s_2}{\Rightarrow} w_1w_2$, existe una transición $(\mathbf{w},C,\epsilon) \longmapsto (\mathbf{w},C\!\rightarrow\! F',\epsilon)$ y una transición $(\mathbf{e},C\!\rightarrow\! F,\epsilon) \longmapsto (\mathbf{e},G,\epsilon)$, por los casos 1 y 2 se cumple que $(\mathbf{w},E,\xi,w_1) \overset{*}{\vdash} (\mathbf{w},C,\xi,\epsilon)$ y por hipótesis de inducción $(\mathbf{w},F',\xi,w_2) \overset{*}{\vdash} (\mathbf{e},F,\phi,\epsilon)$ y en consecuencia se cumple que $(\mathbf{w},E,\xi,w_1w_2) \overset{*}{\vdash} (\mathbf{w},C,\xi,w_2) \vdash (\mathbf{w},C\!\rightarrow\! F',\xi,w_2) \overset{*}{\vdash} (\mathbf{e},C\!\rightarrow\! F,\phi,\epsilon) \vdash (\mathbf{e},G,\phi,\epsilon)$.
  - Si $\langle E^{\mathbf{w}},G^{\mathbf{e}}\rangle[\alpha] \Rightarrow \langle E^{\mathbf{w}},C^{\mathbf{w}}\rangle[\,]\ \langle F'^{\mathbf{w}},F^{\mathbf{e}}\rangle[\alpha\langle\gamma',\eta'\rangle] \overset{s_1}{\Rightarrow} w_1\ \langle F'^{\mathbf{w}},F^{\mathbf{e}}\rangle[\alpha\langle\gamma',\eta'\rangle] \overset{s_2}{\Rightarrow} w_1w_2$, existe una transición $(\mathbf{w},C,\epsilon) \longmapsto (C\nearrow F',\gamma')$ y una transición $(\mathbf{e},C\nearrow F,\eta') \longmapsto (\mathbf{e},G,\epsilon)$, por los casos 1 y 2 se cumple que $(\mathbf{w},E,\xi,w_1) \overset{*}{\vdash} (\mathbf{w},C,\xi,\epsilon)$ y por hipótesis de inducción $(\mathbf{w},F',\xi\gamma',w_2) \overset{*}{\vdash} (\mathbf{e},F,\phi\eta',\epsilon)$ y en consecuencia se cumple que $(\mathbf{w},E,\xi,w_1w_2) \overset{*}{\vdash} (\mathbf{w},C,\xi,w_2) \vdash (\mathbf{w},C\nearrow F',\xi\gamma',w_2) \overset{*}{\vdash} (\mathbf{e},C\nearrow F,\phi\eta',\epsilon) \vdash (\mathbf{e},G,\phi,\epsilon)$.

– Si $\langle E^{\mathbf{w}}, G^{\mathbf{e}}\rangle[\alpha\langle\gamma,\eta\rangle] \Rightarrow \langle E^{\mathbf{w}}, C^{\mathbf{w}}\rangle[\,]\ \langle F'^{\mathbf{w}}, F^{\mathbf{e}}\rangle[\alpha] \overset{s_1}{\Rightarrow} w_1\ \langle F'^{\mathbf{w}}, F^{\mathbf{e}}\rangle[\alpha] \overset{s_2}{\Rightarrow} w_1 w_2$, existe una transición $(\mathbf{w}, C, \gamma) \longmapsto (C\searrow F', \epsilon)$ y una transición $(\mathbf{e}, C\searrow F, \epsilon) \longmapsto (\mathbf{e}, G, \eta)$, por los casos 1 y 2 se cumple que $(\mathbf{w}, E, \xi\gamma, w_1) \overset{*}{\vdash} (\mathbf{w}, C, \xi\gamma, \epsilon)$ y por hipótesis de inducción $(\mathbf{w}, F', \xi, w_2) \overset{*}{\vdash} (\mathbf{e}, F, \phi, \epsilon)$ y en consecuencia se cumple que $(\mathbf{w}, E, \xi\gamma, w_1 w_2) \overset{*}{\vdash} (\mathbf{w}, C, \xi\gamma, w_2) \vdash (\mathbf{w}, C\searrow F', \xi, w_2) \overset{*}{\vdash} (\mathbf{e}, C\searrow F, \phi, \epsilon) \vdash (\mathbf{e}, G, \phi\eta, \epsilon)$.

□

10.3.4 Tabulación

A partir de las características propias de los autómatas con dos pilas fuertemente dirigidos, podemos observar que toda derivación puede clasificarse en uno de los tipos que se enuncian a continuación.

Derivaciones de llamada. Son aquellas que se inician y terminan en modo de escritura en la misma sesión. Presentan alguna de las siguientes formas, dependiendo de la última transición aplicada:

$$\begin{array}{rl} (\mathbf{w},\, \Xi A, \xi, a_h \ldots a_n) & \overset{*}{\vdash}_{d1} (\mathbf{w},\, \Xi A\, \Xi_1 B, \xi\gamma\gamma', a_i \ldots a_n) \\ & \overset{*}{\vdash}_{d2} (\mathbf{w},\, \Xi A\, \Xi_1 B\searrow C, \xi\gamma, a_j \ldots a_n) \end{array}$$

$$\begin{array}{rl} (\mathbf{w},\, \Xi A, \xi, a_h \ldots a_n) & \overset{*}{\vdash}_{d1} (\mathbf{w},\, \Xi A\, \Xi_1 B, \xi\gamma, a_i \ldots a_n) \\ & \overset{*}{\vdash}_{d2} (\mathbf{w},\, \Xi A\, \Xi_1 B\rightarrow C, \xi\gamma, a_j \ldots a_n) \end{array}$$

$$\begin{array}{rl} (\mathbf{w},\, \Xi B, \xi\gamma', a_i \ldots a_n) & \overset{0}{\vdash}_{d1} (\mathbf{w},\, \Xi B, \xi\gamma', a_i \ldots a_n) \\ & \overset{*}{\vdash}_{d2} (\mathbf{w},\, \Xi B\nearrow C, \xi\gamma'\gamma, a_j \ldots a_n) \end{array}$$

donde $\gamma, \gamma' \in V_I$ (en el tercer caso $\gamma' \in V_I \cup \{\models^{\mathbf{w}}, \models^{\mathbf{e}}\}$) y tanto A como B y C pertenecen a la misma sesión. A efectos de uniformizar las explicaciones que siguen, en las derivaciones de la última forma identificaremos ΞA con ΞB y h con i. La subderivación d_1 puede consultar A pero no puede alterar ese elemento de la pila maestra ni ninguno que quede por debajo. En d_1 también se permite la consulta de γ' aunque no su modificación ni la de ξ. La subderivación d_2 puede consultar B pero no puede alterar ni B ni los elementos de la pila maestra que quedan por debajo. Esta subderivación tampoco puede modificar aquella parte de la pila auxiliar que finalmente queda por debajo de γ. En la figura 10.1 se muestra una representación gráfica de este tipo de derivaciones.

Para cualquier $\Xi' \in (\mathcal{D}V_S)^*$ y $\xi' \in (\models^x V_I^*)^*$ tal que $x \in \{\mathbf{w}, \mathbf{e}\}$ y el número de sesiones en Ξ' y ξ' coincide, se cumple

$$\begin{array}{rl} (\mathbf{w},\, \Xi' A, \xi', a_h \ldots a_n) & \overset{*}{\vdash}_{d1} (\mathbf{w},\, \Xi' A\, \Xi_1 B, \xi'\gamma\gamma', a_i \ldots a_n) \\ & \overset{*}{\vdash}_{d2} (\mathbf{w},\, \Xi' A\, \Xi_1 B\searrow C, \xi'\gamma, a_j \ldots a_n) \end{array}$$

$$\begin{array}{rl} (\mathbf{w},\, \Xi' A, \xi', a_h \ldots a_n) & \overset{*}{\vdash}_{d1} (\mathbf{w},\, \Xi' A\, \Xi_1 B, \xi'\gamma, a_i \ldots a_n) \\ & \overset{*}{\vdash}_{d2} (\mathbf{w},\, \Xi' A\, \Xi_1 B\rightarrow C, \xi'\gamma, a_j \ldots a_n) \end{array}$$

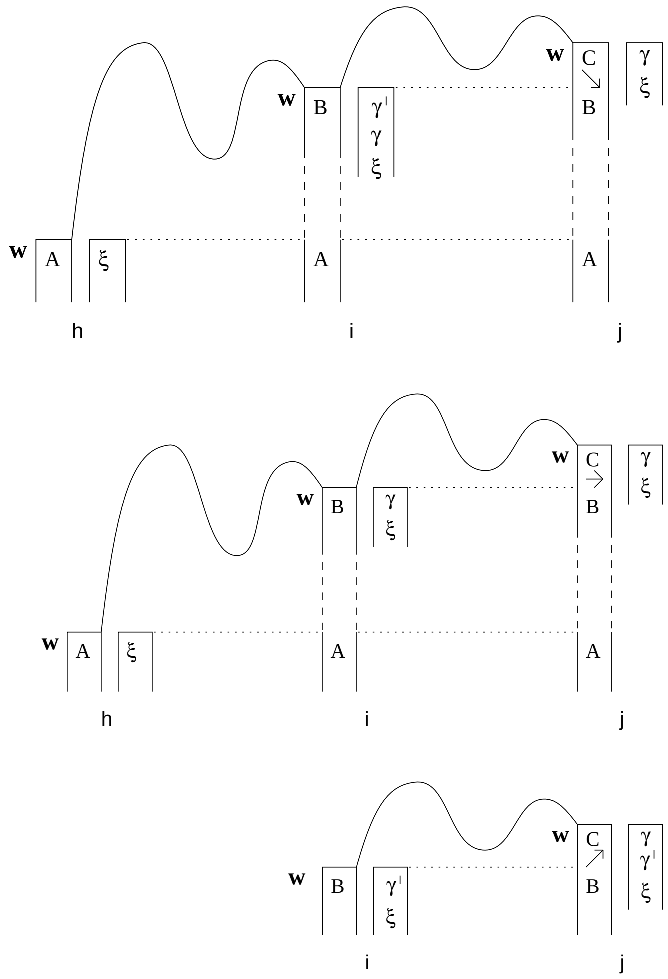

Figura 10.1: Derivaciones de llamada en SD–2SA

$$
\begin{aligned}
(\mathbf{w},\, \Xi' B, \xi'\gamma', a_i \ldots a_n) &\overset{0}{\vdash}_{d1} (\mathbf{w},\, \Xi' B, \xi'\gamma', a_i \ldots a_n) \\
&\overset{*}{\vdash}_{d2} (\mathbf{w},\, \Xi' B \nearrow C, \xi'\gamma'\gamma, a_j \ldots a_n)
\end{aligned}
$$

por lo que este tipo de derivaciones puede ser representado de manera compacta por los correspondientes ítems de la forma

$$[A, h \mid B, i, \gamma', \searrow C, j, \gamma \mid -, -, -, -, -]\mathbf{w}$$

$$[A, h \mid B, i, \gamma, \rightarrow C, j, \gamma \mid -, -, -, -, -]\mathbf{w}$$

$$[B, i \mid B, i, \gamma', \nearrow C, j, \gamma \mid -, -, -, -, -]\mathbf{w}$$

Derivaciones de retorno. Son aquellas que se inician en una sesión en modo de escritura y terminan en la misma sesión en modo de borrado. Estas derivaciones pueden ser de alguna de las tres formas siguientes:

$$
\begin{aligned}
(\mathbf{w},\, \Xi A, \xi, a_h \ldots a_n) &\overset{*}{\vdash}_{d1} (\mathbf{w},\, \Xi A\, \Xi_1 B, \xi\gamma\gamma', a_i \ldots a_n) \\
&\overset{*}{\vdash}_{d2} (\mathbf{w},\, \Xi A\, \Xi_1 B\, \Xi_2 D, \xi\gamma, a_p \ldots a_n) \\
&\overset{*}{\vdash}_{d3} (\mathbf{e},\, \Xi A\, \Xi_1 B\, \Xi_2 D \searrow E, \phi, a_q \ldots a_n) \\
&\overset{*}{\vdash}_{d4} (\mathbf{e},\, \Xi A\, \Xi_1 B \searrow C, \phi\eta, a_j \ldots a_n)
\end{aligned}
$$

$$\begin{aligned}(\mathbf{w},\ \Xi A, \xi, a_h \ldots a_n) \ &\overset{*}{\vdash}_{d1} (\mathbf{w},\ \Xi A\, \Xi_1 B, \xi\gamma, a_i \ldots a_n)\\ &\overset{*}{\vdash}_{d2} (\mathbf{w},\ \Xi A\, \Xi_1 B\, \Xi_2 D, \xi\gamma, a_p \ldots a_n)\\ &\overset{*}{\vdash}_{d3} (\mathbf{e},\ \Xi A\, \Xi_1 B\, \Xi_2 D{\searrow}E, \phi, a_q \ldots a_n)\\ &\overset{*}{\vdash}_{d4} (\mathbf{e},\ \Xi A\, \Xi_1 B{\rightarrow}C, \phi\eta, a_j \ldots a_n)\end{aligned}$$

$$\begin{aligned}(\mathbf{w},\ \Xi B, \xi\gamma', a_i \ldots a_n) \ &\overset{0}{\vdash}_{d1} (\mathbf{w},\ \Xi B, \xi\gamma', a_i \ldots a_n)\\ &\overset{*}{\vdash}_{d2} (\mathbf{w},\ \Xi B\, \Xi_2 D, \xi\gamma'\gamma, a_p \ldots a_n)\\ &\overset{*}{\vdash}_{d3} (\mathbf{e},\ \Xi B\, \Xi_2 D{\searrow}E, \phi\eta', a_q \ldots a_n)\\ &\overset{*}{\vdash}_{d4} (\mathbf{e},\ \Xi B{\nearrow}C, \phi\eta'\eta, a_j \ldots a_n)\end{aligned}$$

donde $\gamma, \gamma', \gamma'', \eta, \eta' \in V_I$ (en el tercer caso $\gamma' \in V_I \cup \{\models^{\mathbf{w}}, \models^{\mathbf{e}}\}$) y tanto A como B, D, E y C pertenecen a la misma sesión. A efectos de uniformizar las explicaciones que siguen, en las derivaciones de la última forma identificaremos ΞA con ΞB y h con i. La subderivación d_1 puede consultar A pero no puede alterar ese elemento de la pila maestra ni ninguno que quede por debajo. En d_1 también se permite la consulta de γ' aunque no su modificación ni la de ξ. Las subderivaciones d_2 y d_4 no pueden modificar B ni ningún elemento de la pila maestra que queda por debajo, aunque la subderivación d_2 puede consultar el propio B. La subderivación d_3 puede consultar D pero no puede alterar dicho elemento ni ningún otro de la pila maestra que quede por debajo. Las distintas apariciones de las pilas $\xi\gamma$, $\xi\gamma'$, ϕ y $\phi\eta'$ que aparecen a lo largo de la derivación se refieren a la misma pila y no a pilas que eventualmente resulten con los mismos contenidos después de apilar y extraer diversos elementos. En la figura 10.2 se muestra una representación gráfica de este tipo de derivaciones.

Para cualquier $\Xi' \in (\mathcal{D}V_S)^*$ y $\xi', \phi' \in (\models^x V_I^*)^*$ tal que $x \in \{\mathbf{w}, \mathbf{e}\}$, el número de sesiones en Ξ', ξ' y ϕ' coincide y, según el caso, existe una derivación

$$(\mathbf{w}, D, \xi\gamma, a_p \ldots a_n) \overset{*}{\vdash} (\mathbf{e}, D{\searrow}E, \phi, a_q \ldots a_n)$$

$$(\mathbf{w}, D, \xi\gamma, a_p \ldots a_n) \overset{*}{\vdash} (\mathbf{e}, D{\searrow}E, \phi, a_q \ldots a_n)$$

$$(\mathbf{w}, D, \xi\gamma'\gamma, a_p \ldots a_n) \overset{*}{\vdash} (\mathbf{e}, D{\searrow}E, \phi\eta', a_q \ldots a_n)$$

se cumple que

$$\begin{aligned}(\mathbf{w},\ \Xi' A, \xi', a_h \ldots a_n) \ &\overset{*}{\vdash}_{d1} (\mathbf{w},\ \Xi' A\, \Xi_1 B, \xi'\gamma\gamma', a_i \ldots a_n)\\ &\overset{*}{\vdash}_{d2} (\mathbf{w},\ \Xi' A\, \Xi_1 B\, \Xi_2 D, \xi'\gamma, a_p \ldots a_n)\\ &\overset{*}{\vdash}_{d3} (\mathbf{e},\ \Xi' A\, \Xi_1 B\, \Xi_2 D{\searrow}E, \phi', a_q \ldots a_n)\\ &\overset{*}{\vdash}_{d4} (\mathbf{e},\ \Xi' A\, \Xi_1 B{\searrow}C, \phi'\eta, a_j \ldots a_n)\end{aligned}$$

$$\begin{aligned}(\mathbf{w},\ \Xi' A, \xi', a_h \ldots a_n) \ &\overset{*}{\vdash}_{d1} (\mathbf{w},\ \Xi' A\, \Xi_1 B, \xi'\gamma, a_i \ldots a_n)\\ &\overset{*}{\vdash}_{d2} (\mathbf{w},\ \Xi' A\, \Xi_1 B\, \Xi_2 D, \xi'\gamma, a_p \ldots a_n)\\ &\overset{*}{\vdash}_{d3} (\mathbf{e},\ \Xi' A\, \Xi_1 B\, \Xi_2 D{\searrow}E, \phi', a_q \ldots a_n)\\ &\overset{*}{\vdash}_{d4} (\mathbf{e},\ \Xi' A\, \Xi_1 B{\rightarrow}C, \phi'\eta, a_j \ldots a_n)\end{aligned}$$

$$
\begin{aligned}
(\mathbf{w},\ \Xi' B, \xi'\gamma', a_i \dots a_n) & \overset{0}{\vdash}_{d1} (\mathbf{w},\ \Xi' B, \xi'\gamma', a_i \dots a_n) \\
& \overset{*}{\vdash}_{d2} (\mathbf{w},\ \Xi' B\, \Xi_2 D, \xi'\gamma'\gamma, a_p \dots a_n) \\
& \overset{*}{\vdash}_{d3} (\mathbf{e},\ \Xi' B\, \Xi_2 D \searrow E, \phi'\eta', a_q \dots a_n) \\
& \overset{*}{\vdash}_{d4} (\mathbf{e},\ \Xi' B \nearrow C, \phi'\eta'\eta, a_j \dots a_n)
\end{aligned}
$$

Ello posibilita que las derivaciones de retorno puedan ser representadas por ítems de la forma

$$[A, h \mid B, i, \gamma', \searrow C, j, \eta \mid D, p, \gamma, E, q]\mathbf{e}$$

$$[A, h \mid B, i, \gamma, \rightarrow C, j, \eta \mid D, p, \gamma, E, q]\mathbf{e}$$

$$[A, h \mid B, i, \gamma', \nearrow C, j, \eta \mid D, p, \gamma, E, q]\mathbf{e}$$

Derivaciones de puntos especiales. Son aquellas derivaciones que involucran una sesión vacía en la pila auxiliar, por lo que presentan la forma

$$(m,\ \Xi B, \xi\gamma', a_i \dots a_n) \overset{*}{\vdash} (m',\ \Xi B \models^m C, \xi\gamma' \models^m, a_j \dots a_n)$$

$$(\mathbf{w},\ \Xi B, \xi \models^m \gamma, a_i \dots a_n) \overset{*}{\vdash} (m',\ \Xi B \searrow C, \xi \models^m, a_j \dots a_n)$$

$$(\mathbf{w},\ \Xi B, \xi \models^m, a_i \dots a_n) \overset{*}{\vdash} (m',\ \Xi B \rightarrow C, \xi \models^m, a_j \dots a_n)$$

donde $\mathbf{w} \leq m'$, $\gamma' \in V_I \cup \{\models^{\mathbf{w}}, \models^{\mathbf{e}}\}$ y $\gamma \in V_I$. Las derivaciones de puntos especiales se muestran gráficamente en la figura 10.3.

Para cualquier $\Xi' \in (\mathcal{D}V_S)^*$ y $\xi' \in (\models^x V_I^*)^*$ tal que $x \in \{\mathbf{w}, \mathbf{e}\}$ y el número de sesiones en Ξ' y ξ' coincide, se cumple

$$(m,\ \Xi' B, \xi'\gamma', a_i \dots a_n) \overset{*}{\vdash} (m',\ \Xi' B \models^m C, \xi'\gamma' \models^m, a_j \dots a_n)$$

$$(\mathbf{w},\ \Xi' B, \xi' \models^m \gamma, a_i \dots a_n) \overset{*}{\vdash} (m',\ \Xi' B \searrow C, \xi' \models^m, a_j \dots a_n)$$

$$(\mathbf{w},\ \Xi' B, \xi' \models^m, a_i \dots a_n) \overset{*}{\vdash} (m',\ \Xi' B \rightarrow C, \xi' \models^m, a_j \dots a_n)$$

por lo que podemos utilizar ítems de la siguientes formas para representar este tipo de derivaciones:

$$[-, - \mid B, i, \gamma', \models^m C, j, \models^m \mid -, -, -, -, -]m'$$

$$[-, - \mid B, i, \gamma, \searrow C, j, \models^m \mid -, -, -, -, -]m'$$

$$[-, - \mid B, i, \models^m, \rightarrow C, j, \models^m \mid -, -, -, -, -]m'$$

Los ítems se combinan mediante las reglas descritas en la tabla 10.7, a partir de un ítem inicial

$$[-, - \mid -, -, -, \models^{\mathbf{w}} \$_0, 0, \models^{\mathbf{w}} \mid -, -, -, -, -]\mathbf{w}$$

La aceptación de la cadena se entrada $a_1 \dots a_n$ se indica mediante la presencia de ítems de la forma

$$[-, - \mid \$_0, 0, \models^{\mathbf{w}}, \models^{\mathbf{w}} \$_f, n, \models^{\mathbf{w}} \mid -, -, -, -, -]\mathbf{e}$$

Teorema 10.5 *La manipulación de configuraciones mediante la aplicación de transiciones en los autómatas con dos pilas fuertemente dirigidos es equivalente a la manipulación de ítems mediante las reglas de combinación de la tabla 10.7.*

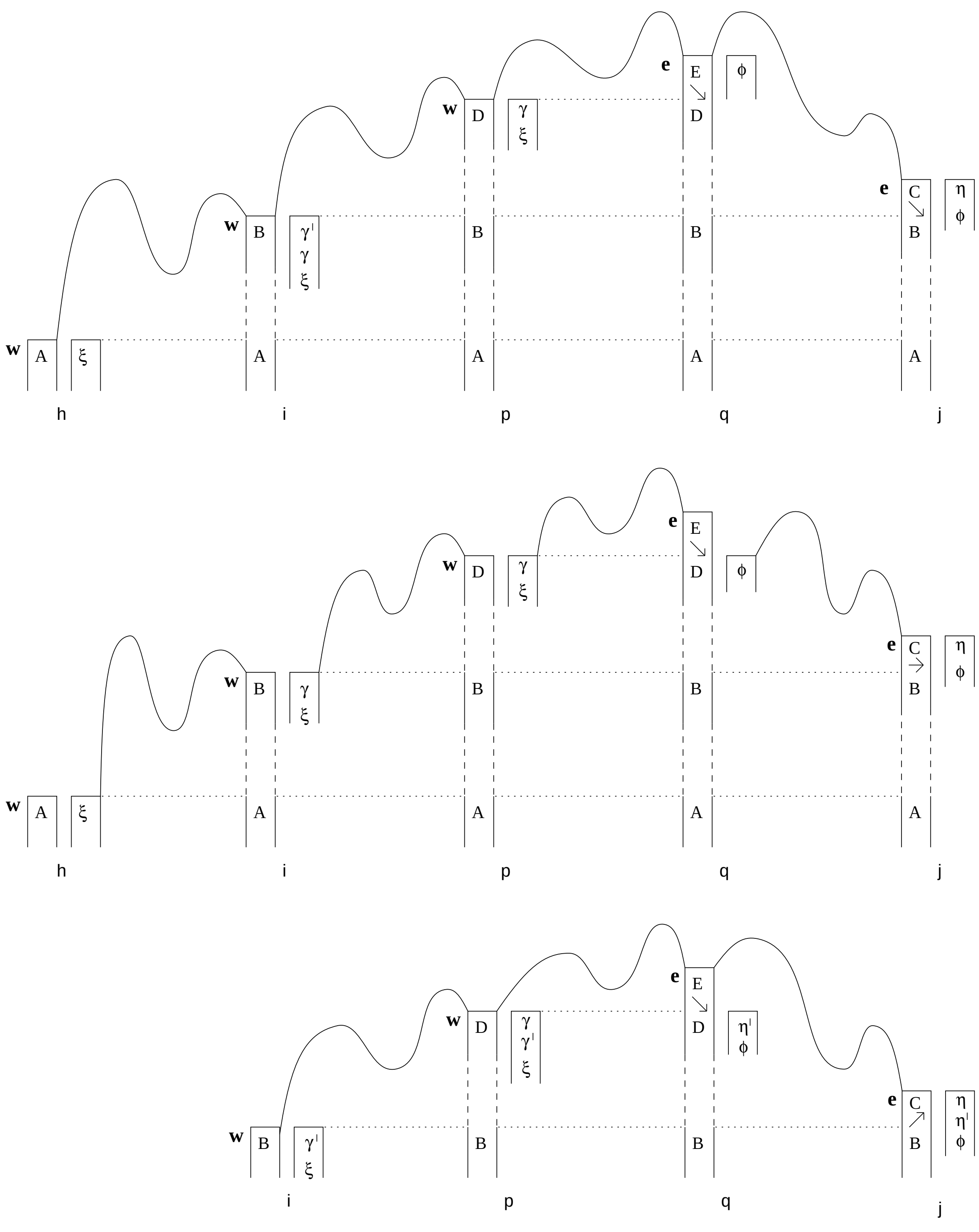

Figura 10.2: Derivaciones de retorno en SD–2SA

$$\frac{[A,h \mid B,i,\gamma',\otimes C,j,\eta \mid D,p,\gamma,E,q]m}{[A,h \mid B,i,\gamma',\otimes F,k,\eta \mid D,p,\gamma,E,q]m} \quad \begin{array}{l}(m,C,\epsilon) \overset{a}{\longmapsto} (m,F,\epsilon),\\ k=j \text{ si } a=\epsilon,\ \ k=j+1 \text{ si } a \in V_T\end{array}$$

$$\frac{[A,h \mid B,i,\gamma',\otimes C,j,\eta \mid D,p,\gamma,E,q]m}{[-,- \mid C,j,\eta,\models^m F,j,\models^m \mid -,-,-,-,-]\mathbf{w}} \quad (m,C,\epsilon) \longmapsto (\mathbf{w},C{\models^m}F,\models^m)$$

$$\frac{[A,h \mid B,i,\gamma',\otimes C,j,\gamma \mid -,-,-,-,-]\mathbf{w}}{[A,h \mid C,j,\gamma,\to F,j,\gamma \mid -,-,-,-,-]\mathbf{w}} \quad (\mathbf{w},C,\epsilon) \longmapsto (\mathbf{w},C{\to}F,\epsilon)$$

$$\frac{[A,h \mid B,i,\gamma'',\otimes C,j,\gamma \mid -,-,-,-,-]\mathbf{w}}{[C,j \mid C,j,\gamma,\nearrow F,j,\gamma' \mid -,-,-,-,-]\mathbf{w}} \quad (\mathbf{w},C,\epsilon) \longmapsto (\mathbf{w},C{\nearrow}F,\gamma')$$

$$\frac{\begin{array}{c}[A,h \mid B,i,\gamma'',\otimes_1 C,j,\gamma \mid -,-,-,-,-]\mathbf{w}\\ [M,m \mid N,t,\gamma''',\otimes_2 A,h,\gamma' \mid -,-,-,-,-]\mathbf{w}\end{array}}{[M,m \mid C,j,\gamma,\searrow F,j,\gamma' \mid -,-,-,-,-]\mathbf{w}} \quad (\mathbf{w},C,\gamma) \longmapsto (\mathbf{w},C{\searrow}F,\epsilon)$$

$$\frac{[-,- \mid B,i,\gamma',\otimes C,j,\models^m \mid -,-,-,-,-]\mathbf{w}}{[-,- \mid B,i,\gamma',\otimes F,k,\models^m \mid -,-,-,-,-]\mathbf{e}} \quad \begin{array}{l}(\mathbf{w},C,\models^m) \overset{a}{\longmapsto} (\mathbf{e},F,\models^m),\\ k=j \text{ si } a=\epsilon,\ \ k=j+1 \text{ si } a \in V_T\end{array}$$

$$\frac{\begin{array}{c}[-,- \mid C,j,\eta,\models^m F,k,\models^m \mid -,-,-,-,-]\mathbf{e}\\ [A,h \mid B,i,\gamma',\otimes C,j,\eta \mid D,p,\gamma,E,q]m\end{array}}{[A,h \mid B,i,\gamma',\otimes G,k,\eta \mid D,p,\gamma,E,q]m} \quad (\mathbf{e},C{\models^m}F,\models^m) \longmapsto (m,G,\epsilon)$$

$$\frac{\begin{array}{c}[A,h \mid C,j,\gamma,\to F,k,\eta \mid D,p,\gamma,E,q]\mathbf{e}\\ [A,h \mid B,i,\gamma',\otimes C,j,\gamma \mid -,-,-,-,-]\mathbf{w}\end{array}}{[A,h \mid B,i,\gamma',\otimes G,k,\eta \mid D,p,\gamma,E,q]\mathbf{e}} \quad (\mathbf{e},C{\to}F,\epsilon) \longmapsto (\mathbf{e},G,\epsilon)$$

$$\frac{\begin{array}{c}[C,j \mid C,j,\gamma,\nearrow F,k,\eta' \mid D,p,\gamma',E,q]\mathbf{e}\\ [A,h \mid B,i,\gamma'',\otimes C,j,\gamma \mid -,-,-,-,-]\mathbf{w}\\ [A,h \mid D,p,\gamma',\searrow E,q,\eta \mid O,u,\gamma,P,v]\mathbf{e}\end{array}}{[A,h \mid B,i,\gamma'',\otimes G,k,\eta \mid O,u,\gamma,P,v]\mathbf{e}} \quad (\mathbf{e},C{\nearrow}F,\eta') \longmapsto (\mathbf{e},G,\epsilon)$$

$$\frac{\begin{array}{c}[M,m \mid C,j,\gamma,\searrow F,k,\eta' \mid D,p,\gamma',E,q]\mathbf{e}\\ [A,h \mid B,i,\gamma'',\otimes_1 C,j,\gamma \mid -,-,-,-,-]\mathbf{w}\\ [M,m \mid N,t,\gamma''',\otimes_2 A,h,\gamma' \mid -,-,-,-,-]\mathbf{w}\end{array}}{[A,h \mid B,i,\gamma'',\otimes_1 G,k,\eta \mid C,j,\gamma,\searrow F,k]\mathbf{e}} \quad (\mathbf{e},C{\searrow}F,\epsilon) \longmapsto (\mathbf{e},G,\eta)$$

Tabla 10.7: Reglas de combinación de ítems en SD–2SA

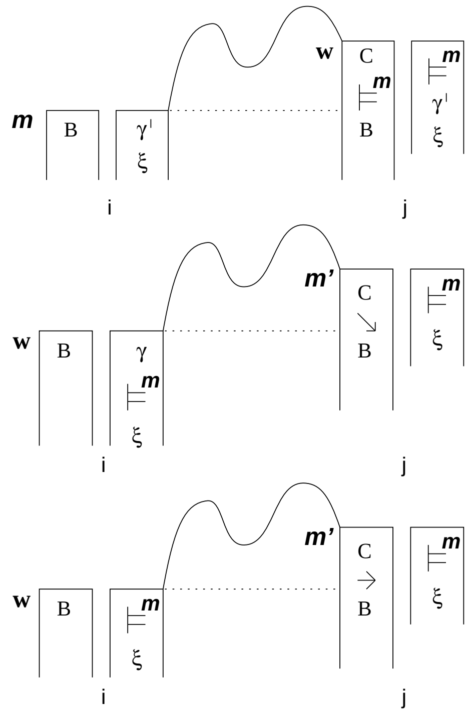

Figura 10.3: Derivaciones de puntos especiales en SD–2SA

<u>Demostración</u>:

Mostraremos que para toda derivación existe una regla de combinación que produce un ítem que representa de forma compacta dicha derivación y que para toda regla de combinación se corresponde con una derivación válida del autómata. Para ello detallaremos todas las posibles derivaciones, junto con las reglas de combinación de ítems correspondientes, en la siguiente lista.

- Derivaciones que son el resultado de aplicar una transición $(m, C, \epsilon) \overset{a}{\longmapsto} (m, F, \epsilon)$
  - a una derivación de llamada, con las tres posibilidades siguientes, en las cuales $k = j$ si $a = \epsilon$ y $k = j + 1$ si $a \in V_T$:

$$\begin{array}{rl} (\mathbf{w},\, \Xi A, \xi, a_h \ldots a_n) & \overset{*}{\vdash} (\mathbf{w},\, \Xi A\,\Xi_1 B, \xi\gamma\gamma', a_i \ldots a_n) \\ & \overset{*}{\vdash} (\mathbf{w},\, \Xi A\,\Xi_1 B {\searrow} C, \xi\gamma, a_j \ldots a_n) \\ & \vdash (\mathbf{w},\, \Xi A\,\Xi_1 B {\searrow} F, \xi\gamma, a_k \ldots a_n) \end{array}$$

$$\frac{[A, h \mid B, i, \gamma', {\searrow}C, j, \gamma \mid -,-,-,-,-]\mathbf{w}}{[A, h \mid B, i, \gamma', {\searrow}F, k, \gamma \mid -,-,-,-,-]\mathbf{w}} \quad (\mathbf{w}, C, \epsilon) \overset{a}{\longmapsto} (\mathbf{w}, F, \epsilon)$$

$$\begin{array}{rl} (\mathbf{w},\, \Xi A, \xi, a_h \ldots a_n) & \overset{*}{\vdash} (\mathbf{w},\, \Xi A\,\Xi_1 B, \xi\gamma, a_i \ldots a_n) \\ & \overset{*}{\vdash} (\mathbf{w},\, \Xi A\,\Xi_1 B {\rightarrow} C, \xi\gamma, a_j \ldots a_n) \\ & \vdash (\mathbf{w},\, \Xi A\,\Xi_1 B {\rightarrow} F, \xi\gamma, a_k \ldots a_n) \end{array}$$

$$\frac{[A, h \mid B, i, \gamma, {\rightarrow}C, j, \gamma \mid -,-,-,-,-]\mathbf{w}}{[A, h \mid B, i, \gamma, {\rightarrow}F, k, \gamma \mid -,-,-,-,-]\mathbf{w}} \quad (\mathbf{w}, C, \epsilon) \overset{a}{\longmapsto} (\mathbf{w}, F, \epsilon)$$

$$\begin{array}{ll} (\mathbf{w},\, \Xi B, \xi\gamma', a_i \dots a_n) & \overset{0}{\vdash} (\mathbf{w},\, \Xi B, \xi\gamma', a_i \dots a_n) \\ & \overset{*}{\vdash} (\mathbf{w},\, \Xi B \nearrow C, \xi\gamma'\gamma, a_j \dots a_n) \\ & \vdash (\mathbf{w},\, \Xi B \nearrow F, \xi\gamma'\gamma, a_k \dots a_n) \end{array}$$

$$\frac{[B, i \mid B, i, \gamma', \nearrow C, j, \gamma \mid -, -, -, -, -]\mathbf{w}}{[B, i \mid B, i, \gamma', \nearrow F, k, \gamma \mid -, -, -, -, -]\mathbf{w}} \quad (\mathbf{w}, C, \epsilon) \overset{a}{\longmapsto} (\mathbf{w}, F, \epsilon)$$

– a una derivación de retorno, con las tres posibilidades siguientes, en las cuales $k = j$ si $a = \epsilon$ y $k = j + 1$ si $a \in V_T$:

$$\begin{array}{ll} (\mathbf{w},\, \Xi A, \xi, a_h \dots a_n) & \overset{*}{\vdash} (\mathbf{w},\, \Xi A\, \Xi_1 B, \xi\gamma\gamma', a_i \dots a_n) \\ & \overset{*}{\vdash} (\mathbf{w},\, \Xi A\, \Xi_1 B\, \Xi_2 D, \xi\gamma, a_p \dots a_n) \\ & \overset{*}{\vdash} (\mathbf{e},\, \Xi A\, \Xi_1 B\, \Xi_2 D \searrow E, \phi, a_q \dots a_n) \\ & \overset{*}{\vdash} (\mathbf{e},\, \Xi A\, \Xi_1 B \searrow C, \phi\eta, a_j \dots a_n) \\ & \vdash (\mathbf{e},\, \Xi A\, \Xi_1 B \searrow F, \phi\eta, a_k \dots a_n) \end{array}$$

$$\frac{[A, h \mid B, i, \gamma', \searrow C, j, \eta \mid D, p, \gamma, E, q]\mathbf{e}}{[A, h \mid B, i, \gamma', \searrow F, k, \eta \mid D, p, \gamma, E, q]\mathbf{e}} \quad (\mathbf{e}, C, \epsilon) \overset{a}{\longmapsto} (\mathbf{e}, F, \epsilon)$$

$$\begin{array}{ll} (\mathbf{w},\, \Xi A, \xi, a_h \dots a_n) & \overset{*}{\vdash} (\mathbf{w},\, \Xi A\, \Xi_1 B, \xi\gamma, a_i \dots a_n) \\ & \overset{*}{\vdash} (\mathbf{w},\, \Xi A\, \Xi_1 B\, \Xi_2 D, \xi\gamma, a_p \dots a_n) \\ & \overset{*}{\vdash} (\mathbf{e},\, \Xi A\, \Xi_1 B\, \Xi_2 D \searrow E, \phi, a_q \dots a_n) \\ & \overset{*}{\vdash} (\mathbf{e},\, \Xi A\, \Xi_1 B \rightarrow C, \phi\eta, a_j \dots a_n) \\ & \vdash (\mathbf{e},\, \Xi A\, \Xi_1 B \rightarrow F, \phi\eta, a_k \dots a_n) \end{array}$$

$$\frac{[A, h \mid B, i, \gamma, \rightarrow C, j, \eta \mid D, p, \gamma, E, q]\mathbf{e}}{[A, h \mid B, i, \gamma, \rightarrow F, k, \eta \mid D, p, \gamma, E, q]\mathbf{e}} \quad (\mathbf{e}, C, \epsilon) \overset{a}{\longmapsto} (\mathbf{e}, F, \epsilon)$$

$$\begin{array}{ll} (\mathbf{w},\, \Xi B, \xi\gamma', a_i \dots a_n) & \overset{0}{\vdash} (\mathbf{w},\, \Xi B, \xi\gamma', a_i \dots a_n) \\ & \overset{*}{\vdash} (\mathbf{w},\, \Xi B\, \Xi_2 D, \xi\gamma'\gamma, a_p \dots a_n) \\ & \overset{*}{\vdash} (\mathbf{e},\, \Xi B\, \Xi_2 D \searrow E, \phi\eta', a_q \dots a_n) \\ & \overset{*}{\vdash} (\mathbf{e},\, \Xi B \nearrow C, \phi\eta'\eta, a_j \dots a_n) \\ & \vdash (\mathbf{e},\, \Xi B \nearrow F, \phi\eta'\eta, a_k \dots a_n) \end{array}$$

$$\frac{[B, i \mid B, i, \gamma', \nearrow C, j, \eta \mid D, p, \gamma, E, q]\mathbf{e}}{[B, i \mid B, i, \gamma', \nearrow F, k, \eta \mid D, p, \gamma, E, q]\mathbf{e}} \quad (\mathbf{e}, C, \epsilon) \overset{a}{\longmapsto} (\mathbf{e}, F, \epsilon)$$

– a una derivación de puntos especiales, con las tres posibilidades siguientes, en las cuales $k = j$ si $a = \epsilon$ y $k = j + 1$ si $a \in V_T$:

$$\begin{array}{ll} (m,\, \Xi B, \xi\gamma', a_i \dots a_n) & \overset{*}{\vdash} (m',\, \Xi B {\models^m} C, \xi\gamma' {\models^m}, a_j \dots a_n) \\ & \vdash (m',\, \Xi B {\models^m} F, \xi\gamma' {\models^m}, a_k \dots a_n) \end{array}$$

$$\frac{[-, - \mid B, i, \gamma', {\models^m} C, j, {\models^m} \mid -, -, -, -, -]m'}{[-, - \mid B, i, \gamma', {\models^m} F, k, {\models^m} \mid -, -, -, -, -]m'} \quad (\mathbf{w}, C, \epsilon) \overset{a}{\longmapsto} (\mathbf{w}, F, \epsilon)$$

$$\begin{array}{ll} (\mathbf{w},\, \Xi B, \xi {\models^m} \gamma, a_i \dots a_n) & \overset{*}{\vdash} (m',\, \Xi B \searrow C, \xi {\models^m}, a_j \dots a_n) \\ & \vdash (m',\, \Xi B \searrow F, \xi {\models^m}, a_k \dots a_n) \end{array}$$

$$\frac{[-,-\mid B,i,\gamma,\searrow C,j,\models^m \mid -,-,-,-,-]m'}{[-,-\mid B,i,\gamma,\searrow F,k,\models^m \mid -,-,-,-,-]m'} \quad (m',C,\epsilon)\overset{a}{\longmapsto}(m',F,\epsilon)$$

$$\begin{aligned}(\mathbf{w},\,\Xi B,\xi{\models^m},a_i\ldots a_n) &\overset{*}{\vdash} (m',\,\Xi B{\to}C,\xi{\models^m},a_j\ldots a_n)\\ &\vdash (m',\,\Xi B{\to}F,\xi{\models^m},a_k\ldots a_n)\end{aligned}$$

$$\frac{[-,-\mid B,i,\models^m,\to C,j,\models^m \mid -,-,-,-,-]m'}{[-,-\mid B,i,\models^m,\to F,k,\models^m \mid -,-,-,-,-]m'} \quad (m',C,\epsilon)\overset{a}{\longmapsto}(m',F,\epsilon)$$

- Derivaciones que son el resultado de aplicar una transición $(m,C,\epsilon)\longmapsto(\mathbf{w},C{\models^m}F,\models^m)$
  - a una derivación de llamada, con las tres posibilidades siguientes:

$$\begin{aligned}(\mathbf{w},\,\Xi A,\xi,a_h\ldots a_n) &\overset{*}{\vdash} (\mathbf{w},\,\Xi A\,\Xi_1 B,\xi\gamma\gamma',a_i\ldots a_n)\\ &\overset{*}{\vdash} (\mathbf{w},\,\Xi A\,\Xi_1 B{\searrow}C,\xi\gamma,a_j\ldots a_n)\\ &\vdash (\mathbf{w},\,\Xi A\,\Xi_1 B{\searrow}C{\models^{\mathbf{w}}}F,\xi\gamma{\models^{\mathbf{w}}},a_j\ldots a_n)\end{aligned}$$

$$\frac{[A,h\mid B,i,\gamma',\searrow C,j,\gamma \mid -,-,-,-,-]\mathbf{w}}{[-,-\mid C,j,\gamma,\models^{\mathbf{w}}F,j,\models^{\mathbf{w}} \mid -,-,-,-,-]\mathbf{w}} \quad (\mathbf{w},C,\epsilon)\longmapsto(\mathbf{w},C{\models^{\mathbf{w}}}F,\models^{\mathbf{w}})$$

$$\begin{aligned}(\mathbf{w},\,\Xi A,\xi,a_h\ldots a_n) &\overset{*}{\vdash} (\mathbf{w},\,\Xi A\,\Xi_1 B,\xi\gamma,a_i\ldots a_n)\\ &\overset{*}{\vdash} (\mathbf{w},\,\Xi A\,\Xi_1 B{\to}C,\xi\gamma,a_j\ldots a_n)\\ &\vdash (\mathbf{w},\,\Xi A\,\Xi_1 B{\to}C{\models^{\mathbf{w}}}F,\xi\gamma{\models^{\mathbf{w}}},a_j\ldots a_n)\end{aligned}$$

$$\frac{[A,h\mid B,i,\gamma,\searrow C,j,\gamma \mid -,-,-,-,-]\mathbf{w}}{[-,-\mid C,j,\gamma,\models^{\mathbf{w}}F,j,\models^{\mathbf{w}} \mid -,-,-,-,-]\mathbf{w}} \quad (\mathbf{w},C,\epsilon)\longmapsto(\mathbf{w},C{\models^{\mathbf{w}}}F,\models^{\mathbf{w}})$$

$$\begin{aligned}(\mathbf{w},\,\Xi B,\xi\gamma',a_i\ldots a_n) &\overset{0}{\vdash} (\mathbf{w},\,\Xi B,\xi\gamma',a_i\ldots a_n)\\ &\overset{*}{\vdash} (\mathbf{w},\,\Xi B{\nearrow}C,\xi\gamma'\gamma,a_j\ldots a_n)\\ &\vdash (\mathbf{w},\,\Xi B{\nearrow}C{\models^{\mathbf{w}}}F,\xi\gamma'\gamma{\models^{\mathbf{w}}},a_j\ldots a_n)\end{aligned}$$

$$\frac{[B,i\mid B,i,\gamma',\searrow C,j,\gamma \mid -,-,-,-,-]\mathbf{w}}{[-,-\mid C,j,\gamma,\models^{\mathbf{w}}F,j,\models^{\mathbf{w}} \mid -,-,-,-,-]\mathbf{w}} \quad (\mathbf{w},C,\epsilon)\longmapsto(\mathbf{w},C{\models^{\mathbf{w}}}F,\models^{\mathbf{w}})$$

  - a una derivación de retorno, con las tres posibilidades siguientes:

$$\begin{aligned}(\mathbf{w},\,\Xi A,\xi,a_h\ldots a_n) &\overset{*}{\vdash} (\mathbf{w},\,\Xi A\,\Xi_1 B,\xi\gamma\gamma',a_i\ldots a_n)\\ &\overset{*}{\vdash} (\mathbf{w},\,\Xi A\,\Xi_1 B\,\Xi_2 D,\xi\gamma,a_p\ldots a_n)\\ &\overset{*}{\vdash} (\mathbf{e},\,\Xi A\,\Xi_1 B\,\Xi_2 D{\searrow}E,\phi,a_q\ldots a_n)\\ &\overset{*}{\vdash} (\mathbf{e},\,\Xi A\,\Xi_1 B{\searrow}C,\phi\eta,a_j\ldots a_n)\\ &\vdash (\mathbf{w},\,\Xi A\,\Xi_1 B{\searrow}C{\models^{\mathbf{e}}}F,\phi\eta{\models^{\mathbf{e}}},a_j\ldots a_n)\end{aligned}$$

$$\frac{[A,h\mid B,i,\gamma',\searrow C,j,\eta \mid D,p,\gamma,E,q]\mathbf{e}}{[-,-\mid C,j,\eta,\models^{\mathbf{e}}F,j,\models^{\mathbf{e}} \mid -,-,-,-,-]\mathbf{w}} \quad (\mathbf{e},C,\epsilon)\longmapsto(\mathbf{w},C{\models^{\mathbf{e}}}F,\models^{\mathbf{e}})$$

$$\begin{aligned}(\mathbf{w},\,\Xi A,\xi,a_h\ldots a_n) &\overset{*}{\vdash} (\mathbf{w},\,\Xi A\,\Xi_1 B,\xi\gamma,a_i\ldots a_n)\\ &\overset{*}{\vdash} (\mathbf{w},\,\Xi A\,\Xi_1 B\,\Xi_2 D,\xi\gamma,a_p\ldots a_n)\\ &\overset{*}{\vdash} (\mathbf{e},\,\Xi A\,\Xi_1 B\,\Xi_2 D{\searrow}E,\phi,a_q\ldots a_n)\\ &\overset{*}{\vdash} (\mathbf{e},\,\Xi A\,\Xi_1 B{\to}C,\phi\eta,a_j\ldots a_n)\\ &\vdash (\mathbf{w},\,\Xi A\,\Xi_1 B{\to}C{\models^{\mathbf{e}}}F,\phi\eta{\models^{\mathbf{e}}},a_j\ldots a_n)\end{aligned}$$

$$\frac{[A,h \mid B,i,\gamma,{\rightarrow}C,j,\eta \mid D,p,\gamma,E,q]\mathbf{e}}{[-,- \mid C,j,\eta,\models^{\mathbf{e}}F,j,\models^{\mathbf{e}} \mid -,-,-,-,-]\mathbf{w}} \quad (\mathbf{e},C,\epsilon) \longmapsto (\mathbf{w},C{\models^{\mathbf{e}}}F,\models^{\mathbf{e}})$$

$$\begin{aligned}(\mathbf{w},\,\Xi B,\xi\gamma',a_i\dots a_n) &\overset{0}{\vdash} (\mathbf{w},\,\Xi B,\xi\gamma',a_i\dots a_n)\\ &\overset{*}{\vdash} (\mathbf{w},\,\Xi B\,\Xi_2 D,\xi\gamma'\gamma,a_p\dots a_n)\\ &\overset{*}{\vdash} (\mathbf{e},\,\Xi B\,\Xi_2 D{\searrow}E,\phi\eta',a_q\dots a_n)\\ &\overset{*}{\vdash} (\mathbf{e},\,\Xi B{\nearrow}C,\phi\eta'\eta,a_j\dots a_n)\\ &\vdash (\mathbf{w},\,\Xi B{\nearrow}C{\models^{\mathbf{e}}}F,\phi\eta'\eta{\models^{\mathbf{e}}},a_j\dots a_n)\end{aligned}$$

$$\frac{[B,i \mid B,i,\gamma',{\nearrow}C,j,\eta \mid D,p,\gamma,E,q]\mathbf{e}}{[-,- \mid C,j,\eta,\models^{\mathbf{e}}F,j,\models^{\mathbf{e}} \mid -,-,-,-,-]\mathbf{w}} \quad (\mathbf{e},C,\epsilon) \longmapsto (\mathbf{w},C{\models^{\mathbf{e}}}F,\models^{\mathbf{e}})$$

– a una derivación de puntos especiales, con las tres posibilidades siguientes:

$$\begin{aligned}(m,\,\Xi B,\xi\gamma',a_i\dots a_n) &\overset{*}{\vdash} (m',\,\Xi B{\models^m}C,\xi\gamma'{\models^m},a_j\dots a_n)\\ &\vdash (\mathbf{w},\,\Xi B{\models^m}C{\models^{m'}}F,\xi\gamma'{\models^m}{\models^{m'}},a_j\dots a_n)\end{aligned}$$

$$\frac{[-,- \mid B,i,\gamma',\models^m C,j,\models^m \mid -,-,-,-,-]m'}{[-,- \mid C,j,\models^m,\models^{m'}F,j,\models^{m'} \mid -,-,-,-,-]\mathbf{w}} \quad (m',C,\epsilon) \longmapsto (\mathbf{w},C{\models^{m'}}F,\models^{m'})$$

$$\begin{aligned}(\mathbf{w},\,\Xi B,\xi{\models^m}\gamma,a_i\dots a_n) &\overset{*}{\vdash} (m',\,\Xi B{\searrow}C,\xi{\models^m},a_j\dots a_n)\\ &\vdash (\mathbf{w},\,\Xi B{\searrow}C{\models^{m'}}F,\xi{\models^m}{\models^{m'}},a_j\dots a_n)\end{aligned}$$

$$\frac{[-,- \mid B,i,\gamma,{\searrow}C,j,\models^m \mid -,-,-,-,-]m'}{[-,- \mid C,j,\models^m,\models^{m'}F,j,\models^{m'} \mid -,-,-,-,-]\mathbf{w}} \quad (m',C,\epsilon) \longmapsto (\mathbf{w},C{\models^{m'}}F,\models^{m'})$$

$$\begin{aligned}(\mathbf{w},\,\Xi B,\xi{\models^m},a_i\dots a_n) &\overset{*}{\vdash} (m',\,\Xi B{\rightarrow}C,\xi{\models^m},a_j\dots a_n)\\ &\vdash (\mathbf{w},\,\Xi B{\rightarrow}C{\models^{m'}}F,\xi{\models^m}{\models^{m'}},a_j\dots a_n)\end{aligned}$$

$$\frac{[-,- \mid B,i,\models^m,{\rightarrow}C,j,\models^m \mid -,-,-,-,-]m'}{[-,- \mid C,j,\models^m,\models^{m'}F,j,\models^{m'} \mid -,-,-,-,-]\mathbf{w}} \quad (m',C,\epsilon) \longmapsto (\mathbf{w},C{\models^{m'}}F,\models^{m'})$$

- Derivaciones que son el resultado de aplicar una transición $(\mathbf{w},C,\epsilon) \longmapsto (\mathbf{w},C{\rightarrow}F,\epsilon)$
  – a una derivación de llamada, con las tres posibilidades siguientes:

$$\begin{aligned}(\mathbf{w},\,\Xi A,\xi,a_h\dots a_n) &\overset{*}{\vdash} (\mathbf{w},\,\Xi A\,\Xi_1 B,\xi\gamma\gamma',a_i\dots a_n)\\ &\overset{*}{\vdash} (\mathbf{w},\,\Xi A\,\Xi_1 B{\searrow}C,\xi\gamma,a_j\dots a_n)\\ &\vdash (\mathbf{w},\,\Xi A\,\Xi_1 B{\searrow}C{\rightarrow}F,\xi\gamma,a_j\dots a_n)\end{aligned}$$

$$\frac{[A,h \mid B,i,\gamma',{\searrow}C,j,\gamma \mid -,-,-,-,-]\mathbf{w}}{[A,h \mid C,j,\gamma,{\rightarrow}F,j,\gamma \mid -,-,-,-,-]\mathbf{w}} \quad (\mathbf{w},C,\epsilon) \longmapsto (\mathbf{w},C{\rightarrow}F,\epsilon)$$

$$\begin{aligned}(\mathbf{w},\,\Xi A,\xi,a_h\dots a_n) &\overset{*}{\vdash} (\mathbf{w},\,\Xi A\,\Xi_1 B,\xi\gamma,a_i\dots a_n)\\ &\overset{*}{\vdash} (\mathbf{w},\,\Xi A\,\Xi_1 B{\rightarrow}C,\xi\gamma,a_j\dots a_n)\\ &\vdash (\mathbf{w},\,\Xi A\,\Xi_1 B{\rightarrow}C{\rightarrow}F,\xi\gamma,a_j\dots a_n)\end{aligned}$$

$$\frac{[A,h \mid B,i,\gamma,{\rightarrow}C,j,\gamma \mid -,-,-,-,-]\mathbf{w}}{[A,h \mid C,j,\gamma,{\rightarrow}F,j,\gamma \mid -,-,-,-,-]\mathbf{w}} \quad (\mathbf{w},C,\epsilon) \longmapsto (\mathbf{w},C{\rightarrow}F,\epsilon)$$

$$\begin{aligned}(\mathbf{w},\,\Xi B,\xi\gamma',a_i\dots a_n) &\overset{0}{\vdash} (\mathbf{w},\,\Xi B,\xi\gamma',a_i\dots a_n)\\ &\overset{*}{\vdash} (\mathbf{w},\,\Xi B{\nearrow}C,\xi\gamma'\gamma,a_j\dots a_n)\\ &\vdash (\mathbf{w},\,\Xi B{\nearrow}C{\rightarrow}F,\xi\gamma'\gamma,a_j\dots a_n)\end{aligned}$$

$$\frac{[B,i \mid B,i,\gamma',{\nearrow}C,j,\gamma \mid -,-,-,-,-]\mathbf{w}}{[B,i \mid C,j,\gamma,{\rightarrow}F,j,\gamma \mid -,-,-,-,-]\mathbf{w}} \quad (\mathbf{w},C,\epsilon) \longmapsto (\mathbf{w},C{\rightarrow}F,\epsilon)$$

– a una derivación de puntos especiales, con las tres posibilidades siguientes:

$$\begin{array}{ll}(m,\ \Xi B, \xi\gamma', a_i \ldots a_n) & \overset{*}{\vdash} (\mathbf{w},\ \Xi B {\models^m} C, \xi\gamma'{\models^m}, a_j \ldots a_n) \\ & \vdash (\mathbf{w},\ \Xi B{\models^m} C{\to}F, \xi\gamma'{\models^m}, a_j \ldots a_n)\end{array}$$

$$\frac{[-,- \mid B,i,\gamma',{\models^m}C,j,{\models^m} \mid -,-,-,-,-]\mathbf{w}}{[-,- \mid C,j,{\models^m},{\to}F,j,{\models^m} \mid -,-,-,-,-]\mathbf{w}}\ (\mathbf{w},C,\epsilon) \longmapsto (\mathbf{w},C{\to}F,\epsilon)$$

$$\begin{array}{ll}(\mathbf{w},\ \Xi B, \xi{\models^m}\gamma, a_i \ldots a_n) & \overset{*}{\vdash} (\mathbf{w},\ \Xi B{\searrow}C, \xi{\models^m}, a_j \ldots a_n) \\ & \vdash (\mathbf{w},\ \Xi B{\searrow}C{\to}F, \xi{\models^m}, a_j \ldots a_n)\end{array}$$

$$\frac{[-,- \mid B,i,\gamma,{\searrow}C,j,{\models^m} \mid -,-,-,-,-]\mathbf{w}}{[-,- \mid C,j,{\models^m},{\to}F,j,{\models^m} \mid -,-,-,-,-]\mathbf{w}}\ (\mathbf{w},C,\epsilon) \longmapsto (\mathbf{w},C{\to}F,\epsilon)$$

$$\begin{array}{ll}(\mathbf{w},\ \Xi B, \xi{\models^m}, a_i \ldots a_n) & \overset{*}{\vdash} (\mathbf{w},\ \Xi B{\to}C, \xi{\models^m}, a_j \ldots a_n) \\ & \vdash (\mathbf{w},\ \Xi B{\to}C{\to}F, \xi{\models^m}, a_j \ldots a_n)\end{array}$$

$$\frac{[-,- \mid B,i,{\models^m},{\to}C,j,{\models^m} \mid -,-,-,-,-]\mathbf{w}}{[-,- \mid C,j,{\models^m},{\to}F,j,{\models^m} \mid -,-,-,-,-]\mathbf{w}}\ (\mathbf{w},C,\epsilon) \longmapsto (\mathbf{w},C{\to}F,\epsilon)$$

- Derivaciones que son el resultado de aplicar una transición $(\mathbf{w},C,\epsilon) \longmapsto (\mathbf{w},C{\nearrow}F,\gamma')$
  – a una derivación de llamada, con las tres posibilidades siguientes:

$$\begin{array}{ll}(\mathbf{w},\ \Xi A, \xi, a_h \ldots a_n) & \overset{*}{\vdash} (\mathbf{w},\ \Xi A\,\Xi_1 B, \xi\gamma\gamma'', a_i \ldots a_n) \\ & \overset{*}{\vdash} (\mathbf{w},\ \Xi A\,\Xi_1 B{\searrow}C, \xi\gamma, a_j \ldots a_n) \\ & \vdash (\mathbf{w},\ \Xi A\,\Xi_1 B{\searrow}C{\nearrow}F, \xi\gamma\gamma', a_j \ldots a_n)\end{array}$$

$$\frac{[A,h \mid B,i,\gamma'',{\searrow}C,j,\gamma \mid -,-,-,-,-]\mathbf{w}}{[C,j \mid C,j,\gamma,{\nearrow}F,j,\gamma' \mid -,-,-,-,-]\mathbf{w}}\ (\mathbf{w},C,\epsilon) \longmapsto (\mathbf{w},C{\nearrow}F,\gamma')$$

$$\begin{array}{ll}(\mathbf{w},\ \Xi A, \xi, a_h \ldots a_n) & \overset{*}{\vdash} (\mathbf{w},\ \Xi A\,\Xi_1 B, \xi\gamma, a_i \ldots a_n) \\ & \overset{*}{\vdash} (\mathbf{w},\ \Xi A\,\Xi_1 B{\to}C, \xi\gamma, a_j \ldots a_n) \\ & \vdash (\mathbf{w},\ \Xi A\,\Xi_1 B{\to}C{\nearrow}F, \xi\gamma\gamma', a_j \ldots a_n)\end{array}$$

$$\frac{[A,h \mid B,i,\gamma,{\to}C,j,\gamma \mid -,-,-,-,-]\mathbf{w}}{[C,j \mid C,j,\gamma,{\nearrow}F,j,\gamma' \mid -,-,-,-,-]\mathbf{w}}\ (\mathbf{w},C,\epsilon) \longmapsto (\mathbf{w},C{\nearrow}F,\gamma')$$

$$\begin{array}{ll}(\mathbf{w},\ \Xi B, \xi\gamma'', a_i \ldots a_n) & \overset{0}{\vdash} (\mathbf{w},\ \Xi B, \xi\gamma'', a_i \ldots a_n) \\ & \overset{*}{\vdash} (\mathbf{w},\ \Xi B{\nearrow}C, \xi\gamma''\gamma, a_j \ldots a_n) \\ & \vdash (\mathbf{w},\ \Xi B{\nearrow}C{\nearrow}F, \xi\gamma''\gamma\gamma', a_j \ldots a_n)\end{array}$$

$$\frac{[B,i \mid B,i,\gamma'',{\nearrow}C,j,\gamma \mid -,-,-,-,-]\mathbf{w}}{[C,j \mid C,j,\gamma,{\nearrow}F,j,\gamma' \mid -,-,-,-,-]\mathbf{w}}\ (\mathbf{w},C,\epsilon) \longmapsto (\mathbf{w},C{\nearrow}F,\gamma')$$

  – a una derivación de puntos especiales, con las tres posibilidades siguientes:

$$\begin{array}{ll}(m,\ \Xi B, \xi\gamma'', a_i \ldots a_n) & \overset{*}{\vdash} (\mathbf{w},\ \Xi B{\models^m}C, \xi\gamma''{\models^m}, a_j \ldots a_n) \\ & \vdash (\mathbf{w},\ \Xi B{\models^m}C{\nearrow}F, \xi\gamma''{\models^m}\gamma', a_j \ldots a_n)\end{array}$$

$$\frac{[-,- \mid B,i,\gamma'',{\models^m}C,j,{\models^m} \mid -,-,-,-,-]\mathbf{w}}{[C,j \mid C,j,{\models^m},{\nearrow}F,j,\gamma' \mid -,-,-,-,-]\mathbf{w}}\ (\mathbf{w},C,\epsilon) \longmapsto (\mathbf{w},C{\nearrow}F,\gamma')$$

$$\begin{array}{ll}(\mathbf{w},\,\Xi B,\xi{\models^m}\gamma,a_i\dots a_n) & \overset{*}{\vdash}(\mathbf{w},\,\Xi B{\searrow}C,\xi{\models^m},a_j\dots a_n)\\ & \vdash(\mathbf{w},\,\Xi B{\searrow}C{\nearrow}F,\xi{\models^m}\gamma',a_j\dots a_n)\end{array}$$

$$\frac{[-,-\mid B,i,\gamma,{\searrow}C,j,{\models^m}\mid -,-,-,-,-]\mathbf{w}}{[C,j\mid C,j,{\models^m},{\nearrow}F,j,\gamma'\mid -,-,-,-,-]\mathbf{w}}\;\;(\mathbf{w},C,\epsilon)\longmapsto(\mathbf{w},C{\nearrow}F,\gamma')$$

$$\begin{array}{ll}(\mathbf{w},\,\Xi B,\xi{\models^m},a_i\dots a_n) & \overset{*}{\vdash}(\mathbf{w},\,\Xi B{\rightarrow}C,\xi{\models^m},a_j\dots a_n)\\ & \vdash(\mathbf{w},\,\Xi B{\rightarrow}C{\nearrow}F,\xi{\models^m}\gamma',a_j\dots a_n)\end{array}$$

$$\frac{[-,-\mid B,i,{\models^m},{\rightarrow}C,j,{\models^m}\mid -,-,-,-,-]\mathbf{w}}{[C,j\mid C,j,{\models^m},{\nearrow}F,j,\gamma'\mid -,-,-,-,-]\mathbf{w}}\;\;(\mathbf{w},C,\epsilon)\longmapsto(\mathbf{w},C{\nearrow}F,\gamma')$$

- Derivaciones que son el resultado de aplicar una transición $(\mathbf{w},C,\gamma)\longmapsto(\mathbf{w},C{\searrow}F,\epsilon)$ a una derivación de llamada, con las tres posibilidades siguientes:

$$\begin{array}{ll}(\mathbf{w},\,\Xi M,\xi,a_m\dots a_n) & \overset{*}{\vdash}(\mathbf{w},\,\Xi M\,\Xi_1 N,\xi'\gamma''',a_t\dots a_n)\\ & \overset{*}{\vdash}(\mathbf{w},\,\Xi M\,\Xi_1 N\otimes A,\xi\gamma',a_h\dots a_n)\\ & \overset{*}{\vdash}(\mathbf{w},\,\Xi M\,\Xi_1 N\otimes A\,\Xi_2 B,\xi\gamma'\gamma\gamma'',a_i\dots a_n)\\ & \overset{*}{\vdash}(\mathbf{w},\,\Xi M\,\Xi_1 N\otimes A\,\Xi_2 B{\searrow}C,\xi\gamma'\gamma,a_j\dots a_n)\\ & \vdash(\mathbf{w},\,\Xi M\,\Xi_1 N\otimes A\,\Xi_2 B{\searrow}C{\searrow}F,\xi\gamma',a_j\dots a_n)\end{array}$$

$$\frac{\begin{array}{c}[A,h\mid B,i,\gamma'',{\searrow}C,j,\gamma\mid -,-,-,-,-]\mathbf{w}\\ [M,m\mid N,t,\gamma''',{\otimes}A,h,\gamma'\mid -,-,-,-,-]\mathbf{w}\end{array}}{[M,m\mid C,j,\gamma,{\searrow}F,j,\gamma'\mid -,-,-,-,-]\mathbf{w}}\;\;(\mathbf{w},C,\gamma)\longmapsto(\mathbf{w},C{\searrow}F,\epsilon)$$

$$\begin{array}{ll}(\mathbf{w},\,\Xi M,\xi,a_m\dots a_n) & \overset{*}{\vdash}(\mathbf{w},\,\Xi M\,\Xi_1 N,\xi'\gamma''',a_t\dots a_n)\\ & \overset{*}{\vdash}(\mathbf{w},\,\Xi M\,\Xi_1 N\otimes A,\xi\gamma',a_h\dots a_n)\\ & \overset{*}{\vdash}(\mathbf{w},\,\Xi M\,\Xi_1 N\otimes A\,\Xi_2 B,\xi\gamma'\gamma,a_i\dots a_n)\\ & \overset{*}{\vdash}(\mathbf{w},\,\Xi M\,\Xi_1 N\otimes A\,\Xi_2 B{\rightarrow}C,\xi\gamma'\gamma,a_j\dots a_n)\\ & \vdash(\mathbf{w},\,\Xi M\,\Xi_1 N\otimes A\,\Xi_2 B{\rightarrow}C{\searrow}F,\xi\gamma',a_j\dots a_n)\end{array}$$

$$\frac{\begin{array}{c}[A,h\mid B,i,\gamma'',{\rightarrow}C,j,\gamma\mid -,-,-,-,-]\mathbf{w}\\ [M,m\mid N,t,\gamma''',{\otimes}A,h,\gamma'\mid -,-,-,-,-]\mathbf{w}\end{array}}{[M,m\mid C,j,\gamma,{\searrow}F,j,\gamma'\mid -,-,-,-,-]\mathbf{w}}\;\;(\mathbf{w},C,\gamma)\longmapsto(\mathbf{w},C{\searrow}F,\epsilon)$$

$$\begin{array}{ll}(\mathbf{w},\,\Xi M,\xi,a_m\dots a_n) & \overset{*}{\vdash}(\mathbf{w},\,\Xi M\,\Xi_1 N,\xi'\gamma''',a_t\dots a_n)\\ & \overset{*}{\vdash}(\mathbf{w},\,\Xi M\,\Xi_1 N\otimes B,\xi\gamma',a_h\dots a_n)\\ & \overset{0}{\vdash}(\mathbf{w},\,\Xi M\,\Xi_1 N\otimes B,\xi\gamma',a_i\dots a_n)\\ & \overset{*}{\vdash}(\mathbf{w},\,\Xi M\,\Xi_1 N\otimes B{\nearrow}C,\xi\gamma'\gamma,a_j\dots a_n)\\ & \vdash(\mathbf{w},\,\Xi M\,\Xi_1 N\otimes B{\nearrow}C{\searrow}F,\xi\gamma',a_j\dots a_n)\end{array}$$

$$\frac{\begin{array}{c}[B,i\mid B,i,\gamma',{\nearrow}C,j,\gamma\mid -,-,-,-,-]\mathbf{w}\\ [M,m\mid N,t,\gamma''',{\otimes}B,i,\gamma'\mid -,-,-,-,-]\mathbf{w}\end{array}}{[M,m\mid C,j,\gamma,{\searrow}F,j,\gamma'\mid -,-,-,-,-]\mathbf{w}}\;\;(\mathbf{w},C,\gamma)\longmapsto(\mathbf{w},C{\searrow}F,\epsilon)$$

- Derivaciones que son el resultado de aplicar una transición $(\mathbf{w}, C, \models^m) \overset{a}{\longmapsto} (\mathbf{e}, F, \models^m)$ a una derivación de puntos especiales, con los tres casos siguientes, en los cuales $k = j$ si $a = \epsilon$ y $k = j+1$ si $a \in V_T$:

$$\begin{array}{ll}(m,\, \Xi B, \xi\gamma', a_i \ldots a_n) & \overset{*}{\vdash} (\mathbf{w},\, \Xi B{\models^m}C, \xi\gamma'{\models^m}, a_j \ldots a_n) \\ & \vdash (\mathbf{e},\, \Xi B{\models^m}F, \xi\gamma'{\models^m}, a_j \ldots a_n)\end{array}$$

$$\frac{[-,- \mid B, i, \gamma', {\models^m}C, j, {\models^m} \mid -,-,-,-,-]\mathbf{w}}{[-,- \mid B, i, \gamma', {\models^m}F, k, {\models^m} \mid -,-,-,-,-]\mathbf{e}} \quad (\mathbf{w}, C, \models^m) \overset{a}{\longmapsto} (\mathbf{e}, F, \models^m)$$

$$\begin{array}{ll}(\mathbf{w},\, \Xi B, \xi{\models^m}\gamma, a_i \ldots a_n) & \overset{*}{\vdash} (\mathbf{w},\, \Xi B{\searrow}C, \xi{\models^m}, a_j \ldots a_n) \\ & \vdash (\mathbf{e},\, \Xi B{\searrow}F, \xi{\models^m}, a_j \ldots a_n)\end{array}$$

$$\frac{[-,- \mid B, i, \gamma, {\searrow}C, j, {\models^m} \mid -,-,-,-,-]\mathbf{w}}{[-,- \mid B, i, \gamma, {\searrow}F, k, {\models^m} \mid -,-,-,-,-]\mathbf{e}} \quad (\mathbf{w}, C, \models^m) \overset{a}{\longmapsto} (\mathbf{e}, F, \models^m)$$

$$\begin{array}{ll}(\mathbf{w},\, \Xi B, \xi{\models^m}, a_i \ldots a_n) & \overset{*}{\vdash} (\mathbf{w},\, \Xi B{\rightarrow}C, \xi{\models^m}, a_j \ldots a_n) \\ & \vdash (\mathbf{e},\, \Xi B{\rightarrow}F, \xi{\models^m}, a_j \ldots a_n)\end{array}$$

$$\frac{[-,- \mid B, i, {\models^m}, {\rightarrow}C, j, {\models^m} \mid -,-,-,-,-]\mathbf{w}}{[-,- \mid B, i, {\models^m}, {\rightarrow}F, k, {\models^m} \mid -,-,-,-,-]\mathbf{e}} \quad (\mathbf{w}, C, \models^m) \overset{a}{\longmapsto} (\mathbf{e}, F, \models^m)$$

- Derivaciones que son el resultado de aplicar una transición $(\mathbf{e}, C{\models^m}F, \models^m) \longmapsto (m, G, \epsilon)$ a una derivación obtenida tras aplicar una transición $(m, C, \epsilon) \longmapsto (\mathbf{w}, C{\models^m}F', \models^m)$
  - a una derivación de llamada, con los tres casos siguientes:

$$\begin{array}{ll}(\mathbf{w},\, \Xi A, \xi, a_h \ldots a_n) & \overset{*}{\vdash} (\mathbf{w},\, \Xi A\, \Xi_1 B, \xi\gamma\gamma', a_i \ldots a_n) \\ & \overset{*}{\vdash} (\mathbf{w},\, \Xi A\, \Xi_1 B{\searrow}C, \xi\gamma, a_j \ldots a_n) \\ & \vdash (\mathbf{w},\, \Xi A\, \Xi_1 B{\searrow}C{\models^{\mathbf{w}}}F', \xi\gamma{\models^{\mathbf{w}}}, a_j \ldots a_n) \\ & \overset{*}{\vdash} (\mathbf{e},\, \Xi A\, \Xi_1 B{\searrow}C{\models^{\mathbf{w}}}F, \xi\gamma{\models^{\mathbf{w}}}, a_k \ldots a_n) \\ & \vdash (\mathbf{w},\, \Xi A\, \Xi_1 B{\searrow}G, \xi\gamma, a_k \ldots a_n)\end{array}$$

$$\frac{\begin{array}{c}[-,- \mid C, j, \gamma, {\models^{\mathbf{w}}}F, k, {\models^{\mathbf{w}}} \mid -,-,-,-,-]\mathbf{e} \\ [A, h \mid B, i, \gamma', {\searrow}C, j, \gamma \mid -,-,-,-,-]\mathbf{w}\end{array}}{[A, h \mid B, i, \gamma', {\searrow}G, k, \gamma \mid -,-,-,-,-]\mathbf{w}} \quad (\mathbf{e}, C{\models^{\mathbf{w}}}F, \models^{\mathbf{w}}) \longmapsto (\mathbf{w}, G, \epsilon)$$

$$\begin{array}{ll}(\mathbf{w},\, \Xi A, \xi, a_h \ldots a_n) & \overset{*}{\vdash} (\mathbf{w},\, \Xi A\, \Xi_1 B, \xi\gamma, a_i \ldots a_n) \\ & \overset{*}{\vdash} (\mathbf{w},\, \Xi A\, \Xi_1 B{\rightarrow}C, \xi\gamma, a_j \ldots a_n) \\ & \vdash (\mathbf{w},\, \Xi A\, \Xi_1 B{\rightarrow}C{\models^{\mathbf{w}}}F', \xi\gamma{\models^{\mathbf{w}}}, a_j \ldots a_n) \\ & \overset{*}{\vdash} (\mathbf{e},\, \Xi A\, \Xi_1 B{\rightarrow}C{\models^{\mathbf{w}}}F, \xi\gamma{\models^{\mathbf{w}}}, a_k \ldots a_n) \\ & \vdash (\mathbf{w},\, \Xi A\, \Xi_1 B{\rightarrow}G, \xi\gamma, a_k \ldots a_n)\end{array}$$

$$\frac{\begin{array}{c}[-,- \mid C, j, \gamma, {\models^{\mathbf{w}}}F, k, {\models^{\mathbf{w}}} \mid -,-,-,-,-]\mathbf{e} \\ [A, h \mid B, i, \gamma, {\rightarrow}C, j, \gamma \mid -,-,-,-,-]\mathbf{w}\end{array}}{[A, h \mid B, i, \gamma, {\rightarrow}G, k, \gamma \mid -,-,-,-,-]\mathbf{w}} \quad (\mathbf{e}, C{\models^{\mathbf{w}}}F, \models^{\mathbf{w}}) \longmapsto (\mathbf{w}, G, \epsilon)$$

$$\begin{array}{ll}(\mathbf{w},\, \Xi B, \xi\gamma', a_i \ldots a_n) & \overset{0}{\vdash} (\mathbf{w},\, \Xi B, \xi\gamma', a_i \ldots a_n) \\ & \overset{*}{\vdash} (\mathbf{w},\, \Xi B{\nearrow}C, \xi\gamma'\gamma, a_j \ldots a_n) \\ & \vdash (\mathbf{w},\, \Xi B{\nearrow}C{\models^{\mathbf{w}}}F', \xi\gamma'\gamma{\models^{\mathbf{w}}}, a_j \ldots a_n) \\ & \overset{*}{\vdash} (\mathbf{e},\, \Xi B{\nearrow}C{\models^{\mathbf{w}}}F, \xi\gamma'\gamma{\models^{\mathbf{w}}}, a_k \ldots a_n) \\ & \vdash (\mathbf{w},\, \Xi B{\nearrow}G, \xi\gamma'\gamma, a_k \ldots a_n)\end{array}$$

$$\frac{\begin{array}{c}[-,- \mid C,j,\gamma,\models^{\mathbf{w}}F,k,\models^{\mathbf{w}} \mid -,-,-,-,-]\mathbf{e}\\ [B,i \mid B,i,\gamma',\nearrow C,j,\gamma \mid -,-,-,-,-]\mathbf{w}\end{array}}{[B,i \mid B,i,\gamma',\nearrow G,k,\gamma \mid -,-,-,-,-]\mathbf{w}} \quad (\mathbf{e},C\models^{\mathbf{w}}F,\models^{\mathbf{w}}) \longmapsto (\mathbf{w},G,\epsilon)$$

– a una derivación de retorno, con las tres posibilidades siguientes:

$$\begin{array}{rl}(\mathbf{w},\,\Xi A,\xi,a_h\ldots a_n) & \overset{*}{\vdash} (\mathbf{w},\,\Xi A\,\Xi_1 B,\xi\gamma\gamma',a_i\ldots a_n)\\ & \overset{*}{\vdash} (\mathbf{w},\,\Xi A\,\Xi_1 B\,\Xi_2 D,\xi\gamma,a_p\ldots a_n)\\ & \overset{*}{\vdash} (\mathbf{e},\,\Xi A\,\Xi_1 B\,\Xi_2 D\searrow E,\phi,a_q\ldots a_n)\\ & \overset{*}{\vdash} (\mathbf{e},\,\Xi A\,\Xi_1 B\searrow C,\phi\eta,a_j\ldots a_n)\\ & \vdash (\mathbf{w},\,\Xi A\,\Xi_1 B\searrow C\models^{\mathbf{e}}F',\phi\eta\models^{\mathbf{e}},a_j\ldots a_n)\\ & \overset{*}{\vdash} (\mathbf{e},\,\Xi A\,\Xi_1 B\searrow C\models^{\mathbf{e}}F,\phi\eta\models^{\mathbf{e}},a_k\ldots a_n)\\ & \vdash (\mathbf{e},\,\Xi A\,\Xi_1 B\searrow G,\phi\eta,a_k\ldots a_n)\end{array}$$

$$\frac{\begin{array}{c}[-,- \mid C,j,\eta,\models^{\mathbf{e}}F,k,\models^{\mathbf{e}} \mid -,-,-,-,-]\mathbf{e}\\ [A,h \mid B,i,\gamma',\searrow C,j,\eta \mid D,p,\gamma,E,q]\mathbf{e}\end{array}}{[A,h \mid B,i,\gamma',\searrow G,k,\eta \mid D,p,\gamma,E,q]\mathbf{e}} \quad (\mathbf{e},C\models^{\mathbf{e}}F,\models^{\mathbf{e}}) \longmapsto (\mathbf{e},G,\epsilon)$$

$$\begin{array}{rl}(\mathbf{w},\,\Xi A,\xi,a_h\ldots a_n) & \overset{*}{\vdash} (\mathbf{w},\,\Xi A\,\Xi_1 B,\xi\gamma,a_i\ldots a_n)\\ & \overset{*}{\vdash} (\mathbf{w},\,\Xi A\,\Xi_1 B\,\Xi_2 D,\xi\gamma,a_p\ldots a_n)\\ & \overset{*}{\vdash} (\mathbf{e},\,\Xi A\,\Xi_1 B\,\Xi_2 D\searrow E,\phi,a_q\ldots a_n)\\ & \overset{*}{\vdash} (\mathbf{e},\,\Xi A\,\Xi_1 B\rightarrow C,\phi\eta,a_j\ldots a_n)\\ & \vdash (\mathbf{w},\,\Xi A\,\Xi_1 B\rightarrow C\models^{\mathbf{e}}F',\phi\eta\models^{\mathbf{e}},a_j\ldots a_n)\\ & \overset{*}{\vdash} (\mathbf{e},\,\Xi A\,\Xi_1 B\rightarrow C\models^{\mathbf{e}}F,\phi\eta\models^{\mathbf{e}},a_k\ldots a_n)\\ & \vdash (\mathbf{e},\,\Xi A\,\Xi_1 B\rightarrow G,\phi\eta,a_k\ldots a_n)\end{array}$$

$$\frac{\begin{array}{c}[-,- \mid C,j,\eta,\models^{\mathbf{e}}F,k,\models^{\mathbf{e}} \mid -,-,-,-,-]\mathbf{e}\\ [A,h \mid B,i,\gamma,\rightarrow C,j,\eta \mid D,p,\gamma,E,q]\mathbf{e}\end{array}}{[A,h \mid B,i,\gamma,\rightarrow G,k,\eta \mid D,p,\gamma,E,q]\mathbf{e}} \quad (\mathbf{e},C\models^{\mathbf{e}}F,\models^{\mathbf{e}}) \longmapsto (\mathbf{e},G,\epsilon)$$

$$\begin{array}{rl}(\mathbf{w},\,\Xi B,\xi\gamma',a_i\ldots a_n) & \overset{0}{\vdash} (\mathbf{w},\,\Xi B,\xi\gamma',a_i\ldots a_n)\\ & \overset{*}{\vdash} (\mathbf{w},\,\Xi B\,\Xi_2 D,\xi\gamma'\gamma,a_p\ldots a_n)\\ & \overset{*}{\vdash} (\mathbf{e},\,\Xi B\,\Xi_2 D\searrow E,\phi\eta',a_q\ldots a_n)\\ & \overset{*}{\vdash} (\mathbf{e},\,\Xi B\nearrow C,\phi\eta'\eta,a_j\ldots a_n)\\ & \vdash (\mathbf{w},\,\Xi B\nearrow C\models^{\mathbf{e}}F',\phi\eta'\eta\models^{\mathbf{e}},a_j\ldots a_n)\\ & \overset{*}{\vdash} (\mathbf{e},\,\Xi B\nearrow C\models^{\mathbf{e}}F,\phi\eta'\eta\models^{\mathbf{e}},a_k\ldots a_n)\\ & \vdash (\mathbf{e},\,\Xi B\nearrow G,\phi\eta'\eta,a_k\ldots a_n)\end{array}$$

$$\frac{\begin{array}{c}[-,- \mid C,j,\eta,\models^{\mathbf{e}}F,k,\models^{\mathbf{e}} \mid -,-,-,-,-]\mathbf{e}\\ [B,i \mid B,i,\gamma',\nearrow C,j,\eta \mid D,p,\gamma,E,q]\mathbf{e}\end{array}}{[B,i \mid B,i,\gamma',\nearrow G,k,\eta \mid D,p,\gamma,E,q]\mathbf{e}} \quad (\mathbf{e},C\models^{\mathbf{e}}F,\models^{\mathbf{e}}) \longmapsto (\mathbf{e},G,\epsilon)$$

– a una derivación de puntos especiales, con las tres posibilidades siguientes:

$$\begin{array}{rl}(m,\,\Xi B,\xi\gamma',a_i\ldots a_n) & \overset{*}{\vdash} (m',\,\Xi B\models^{m}C,\xi\gamma'\models^{m},a_j\ldots a_n)\\ & \vdash (\mathbf{w},\,\Xi B\models^{m}C\models^{m'}F',\xi\gamma'\models^{m}\models^{m'},a_j\ldots a_n)\\ & \overset{*}{\vdash} (\mathbf{e},\,\Xi B\models^{m}C\models^{m'}F,\xi\gamma'\models^{m}\models^{m'},a_k\ldots a_n)\\ & \vdash (m',\,\Xi B\models^{m}G,\xi\gamma'\models^{m},a_k\ldots a_n)\end{array}$$

$$\frac{\begin{array}{c}[-,- \mid C,j,\models^m,\models^{m'}F,k,\models^{m'} \mid -,-,-,-,-]\mathbf{e}\\ [A,h \mid B,i,\gamma',\models^m C,j,\models^m \mid -,-,-,-,-]m'\end{array}}{[A,h \mid B,i,\gamma',\models^m G,k,\models^m \mid -,-,-,-,-]m'}\ (\mathbf{e},C\models^{m'}F,\models^{m'}) \longmapsto (m',G,\epsilon)$$

$$\begin{array}{ll}(\mathbf{w},\,\Xi B,\xi\models^m\gamma,a_i\dots a_n) & \overset{*}{\vdash} (m',\,\Xi B\searrow C,\xi\models^m,a_j\dots a_n)\\ & \vdash (\mathbf{w},\,\Xi B\searrow C\models^{m'}F',\xi\models^m\models^{m'},a_j\dots a_n)\\ & \overset{*}{\vdash} (\mathbf{e},\,\Xi B\searrow C\models^{m'}F,\xi\models^m\models^{m'},a_k\dots a_n)\\ & \vdash (m',\,\Xi B\searrow C\models^m F,\xi\models^m\models^m,a_k\dots a_n)\end{array}$$

$$\frac{\begin{array}{c}[-,- \mid C,j,\models^m,\models^{m'}F,k,\models^{m'} \mid -,-,-,-,-]\mathbf{e}\\ [A,h \mid B,i,\gamma,\searrow C,j,\models^m \mid -,-,-,-,-]m'\end{array}}{[A,h \mid B,i,\gamma,\searrow G,k,\models^m \mid -,-,-,-,-]m'}\ (\mathbf{e},C\models^{m'}F,\models^{m'}) \longmapsto (m',G,\epsilon)$$

$$\begin{array}{ll}(\mathbf{w},\,\Xi B,\xi\models^m,a_i\dots a_n) & \overset{*}{\vdash} (m',\,\Xi B\rightarrow C,\xi\models^m,a_j\dots a_n)\\ & \vdash (\mathbf{w},\,\Xi B\rightarrow C\models^{m'}F',\xi\models^m\models^{m'},a_j\dots a_n)\\ & \overset{*}{\vdash} (\mathbf{e},\,\Xi B\rightarrow C\models^{m'}F,\xi\models^m\models^{m'},a_k\dots a_n)\\ & \vdash (m',\,\Xi B\rightarrow C\models^m F,\xi\models^m\models^m,a_k\dots a_n)\end{array}$$

$$\frac{\begin{array}{c}[-,- \mid C,j,\models^m,\models^{m'}F,k,\models^{m'} \mid -,-,-,-,-]\mathbf{e}\\ [A,h \mid B,i,\models^m,\rightarrow C,j,\models^m \mid -,-,-,-,-]m'\end{array}}{[A,h \mid B,i,\models^m,\rightarrow G,k,\models^m \mid -,-,-,-,-]m'}\ (\mathbf{e},C\models^{m'}F,\models^{m'}) \longmapsto (m',G,\epsilon)$$

- Derivaciones que son el resultado de aplicar una transición $(\mathbf{e},C\rightarrow F,\epsilon) \longmapsto (\mathbf{e},G,\epsilon)$ a una derivación obtenida tras aplicar una transición $(\mathbf{w},C,\epsilon) \longmapsto (\mathbf{w},C\rightarrow F',\epsilon)$
  - a una derivación de llamada, con las tres posibilidades siguientes:

$$\begin{array}{ll}(\mathbf{w},\,\Xi A,\xi,a_h\dots a_n) & \overset{*}{\vdash} (\mathbf{w},\,\Xi A\,\Xi_1 B,\xi\gamma\gamma',a_i\dots a_n)\\ & \overset{*}{\vdash} (\mathbf{w},\,\Xi A\,\Xi_1 B\searrow C,\xi\gamma,a_j\dots a_n)\\ & \vdash (\mathbf{w},\,\Xi A\,\Xi_1 B\searrow C\rightarrow F',\xi\gamma,a_j\dots a_n)\\ & \overset{*}{\vdash} (\mathbf{w},\,\Xi A\,\Xi_1 B\searrow C\rightarrow F'\,\Xi_2 D,\xi\gamma,a_p\dots a_n)\\ & \overset{*}{\vdash} (\mathbf{e},\,\Xi A\,\Xi_1 B\searrow C\rightarrow F'\,\Xi_2 D\searrow E,\phi,a_q\dots a_n)\\ & \overset{*}{\vdash} (\mathbf{e},\,\Xi A\,\Xi_1 B\searrow C\rightarrow F,\phi\eta,a_k\dots a_n)\\ & \vdash (\mathbf{e},\,\Xi A\,\Xi_1 B\searrow G,\phi\eta,a_k\dots a_n)\end{array}$$

$$\frac{\begin{array}{c}[A,h \mid C,j,\gamma,\rightarrow F,k,\eta \mid D,p,\gamma,E,q]\mathbf{e}\\ [A,h \mid B,i,\gamma',\searrow C,j,\gamma \mid -,-,-,-,-]\mathbf{w}\end{array}}{[A,h \mid B,i,\gamma',\searrow G,k,\eta \mid D,p,\gamma,E,q]\mathbf{e}}\ (\mathbf{e},C\rightarrow F,\epsilon) \longmapsto (\mathbf{e},G,\epsilon)$$

$$\begin{array}{ll}(\mathbf{w},\,\Xi A,\xi,a_h\dots a_n) & \overset{*}{\vdash} (\mathbf{w},\,\Xi A\,\Xi_1 B,\xi\gamma,a_i\dots a_n)\\ & \overset{*}{\vdash} (\mathbf{w},\,\Xi A\,\Xi_1 B\rightarrow C,\xi\gamma,a_j\dots a_n)\\ & \vdash (\mathbf{w},\,\Xi A\,\Xi_1 B\rightarrow C\rightarrow F',\xi\gamma,a_j\dots a_n)\\ & \overset{*}{\vdash} (\mathbf{w},\,\Xi A\,\Xi_1 B\rightarrow C\rightarrow F'\Xi_2 D,\xi\gamma,a_p\dots a_n)\\ & \overset{*}{\vdash} (\mathbf{e},\,\Xi A\,\Xi_1 B\rightarrow C\rightarrow F'\,\Xi_2 D\searrow E,\phi,a_j\dots a_n)\\ & \overset{*}{\vdash} (\mathbf{e},\,\Xi A\,\Xi_1 B\rightarrow C\rightarrow F,\phi\eta,a_k\dots a_n)\\ & \vdash (\mathbf{e},\,\Xi A\,\Xi_1 B\rightarrow G,\phi\eta,a_k\dots a_n)\end{array}$$

$$\frac{\begin{array}{c}[A,h \mid C,j,\gamma,\rightarrow F,k,\eta \mid D,p,\gamma,E,q]\mathbf{e}\\ [A,h \mid B,i,\gamma,\rightarrow C,j,\gamma \mid -,-,-,-,-]\mathbf{w}\end{array}}{[A,h \mid B,i,\gamma,\rightarrow G,k,\eta \mid D,p,\gamma,E,q]\mathbf{e}}\ (\mathbf{e},C\rightarrow F,\epsilon) \longmapsto (\mathbf{e},G,\epsilon)$$

$$
\begin{array}{rl}
(\mathbf{w},\, \Xi B, \xi\gamma', a_i \ldots a_n) & \overset{0}{\vdash} (\mathbf{w},\, \Xi B, \xi\gamma', a_i \ldots a_n) \\
& \overset{*}{\vdash} (\mathbf{w},\, \Xi B \nearrow C, \xi\gamma'\gamma, a_j \ldots a_n) \\
& \vdash (\mathbf{w},\, \Xi B \nearrow C \rightarrow F', \xi\gamma'\gamma, a_j \ldots a_n) \\
& \overset{*}{\vdash} (\mathbf{w},\, \Xi B \nearrow C \rightarrow F'\, \Xi_2 D, \xi\gamma'\gamma, a_p \ldots a_n) \\
& \overset{*}{\vdash} (\mathbf{e},\, \Xi B \nearrow C \rightarrow F'\, \Xi_2 D \searrow E, \phi\eta', a_q \ldots a_n) \\
& \overset{*}{\vdash} (\mathbf{e},\, \Xi B \nearrow C \rightarrow F, \phi\eta'\eta, a_k \ldots a_n) \\
& \vdash (\mathbf{e},\, \Xi B \nearrow G, \phi\eta'\eta, a_k \ldots a_n)
\end{array}
$$

$$
\frac{\begin{array}{c}[B,i \mid C,j,\gamma,\rightarrow F,k,\eta \mid D,p,\gamma,E,q]\mathbf{e} \\ [B,i \mid B,i,\gamma',\nearrow C,j,\gamma \mid -,-,-,-,-]\mathbf{w}\end{array}}{[B,i \mid B,i,\gamma',\nearrow G,k,\eta \mid D,p,\gamma,E,q]\mathbf{e}} \quad (\mathbf{e}, C \rightarrow F, \epsilon) \longmapsto (\mathbf{e}, G, \epsilon)
$$

– a una derivación de puntos especiales, con las tres posibilidades siguientes:

$$
\begin{array}{rl}
(m,\, \Xi B, \xi\gamma', a_i \ldots a_n) & \overset{*}{\vdash} (\mathbf{w},\, \Xi B {\models^m} C, \xi\gamma' {\models^m}, a_j \ldots a_n) \\
& \vdash (\mathbf{w},\, \Xi B {\models^m} C \rightarrow F', \xi\gamma' {\models^m}, a_j \ldots a_n) \\
& \overset{*}{\vdash} (\mathbf{e},\, \Xi B {\models^m} C \rightarrow F, \xi\gamma' {\models^m}, a_k \ldots a_n) \\
& \vdash (\mathbf{e},\, \Xi B {\models^m} G, \xi\gamma' {\models^m}, a_k \ldots a_n)
\end{array}
$$

$$
\frac{\begin{array}{c}[-,- \mid C,j,{\models^m},\rightarrow F,k,{\models^m} \mid -,-,-,-,-]\mathbf{e} \\ [-,- \mid B,i,\gamma',{\models^m} C,j,{\models^m} \mid -,-,-,-,-]\mathbf{w}\end{array}}{[-,- \mid B,i,\gamma',{\models^m} G,k,{\models^m} \mid -,-,-,-,-]\mathbf{e}} \quad (\mathbf{e}, C \rightarrow F, \epsilon) \longmapsto (\mathbf{e}, G, \epsilon)
$$

$$
\begin{array}{rl}
(\mathbf{w},\, \Xi B, \xi {\models^m} \gamma, a_i \ldots a_n) & \overset{*}{\vdash} (\mathbf{w},\, \Xi B \searrow C, \xi {\models^m}, a_j \ldots a_n) \\
& \vdash (\mathbf{w},\, \Xi B \searrow C \rightarrow F', \xi {\models^m}, a_j \ldots a_n) \\
& \overset{*}{\vdash} (\mathbf{e},\, \Xi B \searrow C \rightarrow F, \xi {\models^m}, a_k \ldots a_n) \\
& \vdash (\mathbf{e},\, \Xi B \searrow G, \xi {\models^m}, a_k \ldots a_n)
\end{array}
$$

$$
\frac{\begin{array}{c}[-,- \mid C,j,{\models^m},\rightarrow F,k,{\models^m} \mid -,-,-,-,-]\mathbf{e} \\ [-,- \mid B,i,\gamma,\searrow C,j,{\models^m} \mid -,-,-,-,-]\mathbf{w}\end{array}}{[-,- \mid B,i,\gamma,\searrow G,k,{\models^m} \mid -,-,-,-,-]\mathbf{e}} \quad (\mathbf{e}, C \rightarrow F, \epsilon) \longmapsto (\mathbf{e}, G, \epsilon)
$$

$$
\begin{array}{rl}
(\mathbf{w},\, \Xi B, \xi {\models^m}, a_i \ldots a_n) & \overset{*}{\vdash} (\mathbf{w},\, \Xi B \rightarrow C, \xi {\models^m}, a_j \ldots a_n) \\
& \vdash (\mathbf{w},\, \Xi B \rightarrow C \rightarrow F', \xi {\models^m}, a_j \ldots a_n) \\
& \overset{*}{\vdash} (\mathbf{e},\, \Xi B \rightarrow C \rightarrow F, \xi {\models^m}, a_k \ldots a_n) \\
& \vdash (\mathbf{e},\, \Xi B \rightarrow G, \xi {\models^m}, a_k \ldots a_n)
\end{array}
$$

$$
\frac{\begin{array}{c}[-,- \mid C,j,{\models^m},\rightarrow F,k,{\models^m} \mid -,-,-,-,-]\mathbf{e} \\ [-,- \mid B,i,{\models^m},\rightarrow C,j,{\models^m} \mid -,-,-,-,-]\mathbf{w}\end{array}}{[-,- \mid B,i,{\models^m},\rightarrow G,k,{\models^m} \mid -,-,-,-,-]\mathbf{e}} \quad (\mathbf{e}, C \rightarrow F, \epsilon) \longmapsto (\mathbf{e}, G, \epsilon)
$$

- Derivaciones que son el resultado de aplicar una transición $(\mathbf{e}, C \nearrow F, \eta') \longmapsto (\mathbf{e}, G, \epsilon)$ a una derivación obtenida tras aplicar una transición $(\mathbf{w}, C, \epsilon) \longmapsto (\mathbf{w}, C \nearrow F', \gamma')$

– a una derivación de llamada, con los tres casos siguientes:

$$
\begin{array}{ll}
(\mathbf{w},\, \Xi A, \xi, a_h \dots a_n) & \overset{*}{\vdash} (\mathbf{w},\, \Xi A\, \Xi_1 B, \xi\gamma\gamma'', a_i \dots a_n) \\
& \overset{*}{\vdash} (\mathbf{w},\, \Xi A\, \Xi_1 B {\searrow} C, \xi\gamma, a_j \dots a_n) \\
& \vdash (\mathbf{w},\, \Xi A\, \Xi_1 B {\searrow} C {\nearrow} F', \xi\gamma\gamma', a_j \dots a_n) \\
& \overset{*}{\vdash} (\mathbf{w},\, \Xi A\, \Xi_1 B {\searrow} C {\nearrow} F'\, \Xi_2 D, \xi\gamma\gamma', a_p \dots a_n) \\
& \overset{*}{\vdash} (\mathbf{w},\, \Xi A\, \Xi_1 B {\searrow} C {\nearrow} F'\, \Xi_2 D\, \Xi_3 O, \xi\gamma, u \dots a_n) \\
& \overset{*}{\vdash} (\mathbf{e},\, \Xi A\, \Xi_1 B {\searrow} C {\nearrow} F'\, \Xi_2 D\, \Xi_3 O {\searrow} P, \phi, v \dots a_n) \\
& \overset{*}{\vdash} (\mathbf{e},\, \Xi A\, \Xi_1 B {\searrow} C {\nearrow} F'\, \Xi_2 D {\searrow} E, \phi\eta, a_q \dots a_n) \\
& \overset{*}{\vdash} (\mathbf{e},\, \Xi A\, \Xi_1 B {\searrow} C {\nearrow} F, \phi\eta\eta', a_k \dots a_n) \\
& \vdash (\mathbf{e},\, \Xi A\, \Xi_1 B {\searrow} G, \phi\eta, a_k \dots a_n)
\end{array}
$$

$$
\frac{\begin{array}{c}[C, j \mid C, j, \gamma, {\nearrow} F, k, \eta' \mid D, p, \gamma', E, q]\mathbf{e} \\ [A, h \mid B, i, \gamma'', {\searrow} C, j, \gamma \mid -, -, -, -, -]\mathbf{w} \\ [A, h \mid D, p, \gamma', {\searrow} E, q, \eta \mid O, u, \gamma, P, v]\mathbf{e}\end{array}}{[A, h \mid B, i, \gamma'', {\searrow} G, k, \eta \mid O, u, \gamma, P, v]\mathbf{e}} \quad (\mathbf{e}, C {\nearrow} F, \eta') \longmapsto (\mathbf{e}, G, \epsilon)
$$

$$
\begin{array}{ll}
(\mathbf{w},\, \Xi A, \xi, a_h \dots a_n) & \overset{*}{\vdash} (\mathbf{w},\, \Xi A\, \Xi_1 B, \xi\gamma, a_i \dots a_n) \\
& \overset{*}{\vdash} (\mathbf{w},\, \Xi A\, \Xi_1 B {\rightarrow} C, \xi\gamma, a_j \dots a_n) \\
& \vdash (\mathbf{w},\, \Xi A\, \Xi_1 B {\rightarrow} C {\nearrow} F', \xi\gamma\gamma', a_j \dots a_n) \\
& \overset{*}{\vdash} (\mathbf{w},\, \Xi A\, \Xi_1 B {\rightarrow} C {\nearrow} F'\, \Xi_2 D, \xi\gamma\gamma', a_j \dots a_n) \\
& \overset{*}{\vdash} (\mathbf{w},\, \Xi A\, \Xi_1 B {\rightarrow} C {\nearrow} F'\, \Xi_2 D\, \Xi_2 O, \xi\gamma, a_u \dots a_n) \\
& \overset{*}{\vdash} (\mathbf{e},\, \Xi A\, \Xi_1 B {\rightarrow} C {\nearrow} F'\, \Xi_2 D\, \Xi_2 O {\searrow} P, \phi, a_v \dots a_n) \\
& \overset{*}{\vdash} (\mathbf{e},\, \Xi A\, \Xi_1 B {\rightarrow} C {\nearrow} F'\, \Xi_2 D {\searrow} E, \phi\eta, a_j \dots a_n) \\
& \overset{*}{\vdash} (\mathbf{e},\, \Xi A\, \Xi_1 B {\rightarrow} C {\nearrow} F, \phi\eta\eta', a_k \dots a_n) \\
& \vdash (\mathbf{e},\, \Xi A\, \Xi_1 B {\rightarrow} G, \phi\eta, a_k \dots a_n)
\end{array}
$$

$$
\frac{\begin{array}{c}[C, j \mid C, j, \gamma, {\nearrow} F, k, \eta' \mid D, p, \gamma', E, q]\mathbf{e} \\ [A, h \mid B, i, \gamma, {\rightarrow} C, j, \gamma \mid -, -, -, -, -]\mathbf{w} \\ [A, h \mid D, p, \gamma', {\searrow} E, q, \eta \mid O, u, \gamma, P, v]\mathbf{e}\end{array}}{[A, h \mid B, i, \gamma, {\rightarrow} G, k, \eta \mid O, u, \gamma, P, v]\mathbf{e}} \quad (\mathbf{e}, C {\nearrow} F, \eta') \longmapsto (\mathbf{e}, G, \epsilon)
$$

$$
\begin{array}{ll}
(\mathbf{w},\, \Xi B, \xi\gamma'', a_i \dots a_n) & \overset{0}{\vdash} (\mathbf{w},\, \Xi B, \xi\gamma'', a_i \dots a_n) \\
& \overset{*}{\vdash} (\mathbf{w},\, \Xi B {\nearrow} C, \xi\gamma''\gamma, a_j \dots a_n) \\
& \vdash (\mathbf{w},\, \Xi B {\nearrow} C {\nearrow} F', \xi\gamma''\gamma\gamma', a_j \dots a_n) \\
& \overset{*}{\vdash} (\mathbf{w},\, \Xi B {\nearrow} C {\nearrow} F'\, \Xi_1 D, \xi\gamma''\gamma\gamma', a_p \dots a_n) \\
& \overset{*}{\vdash} (\mathbf{w},\, \Xi B {\nearrow} C {\nearrow} F'\, \Xi_1 D\, \Xi_2 O, \xi\gamma''\gamma, a_u \dots a_n) \\
& \overset{*}{\vdash} (\mathbf{e},\, \Xi B {\nearrow} C {\nearrow} F'\, \Xi_1 D\, \Xi_2 O {\searrow} P, \phi\eta'', a_v \dots a_n) \\
& \overset{*}{\vdash} (\mathbf{e},\, \Xi B {\nearrow} C {\nearrow} F'\, \Xi_1 D {\searrow} E, \phi\eta''\eta, a_q \dots a_n) \\
& \overset{*}{\vdash} (\mathbf{e},\, \Xi B {\nearrow} C {\nearrow} F, \phi\eta''\eta\eta', a_k \dots a_n) \\
& \vdash (\mathbf{e},\, \Xi B {\nearrow} G, \phi\eta''\eta, a_k \dots a_n)
\end{array}
$$

$$
\frac{\begin{array}{c}[C, j \mid C, j, \gamma, {\nearrow} F, k, \eta' \mid D, p, \gamma', E, q]\mathbf{e} \\ [B, i \mid B, i, \gamma'', {\nearrow} C, j, \gamma \mid -, -, -, -, -]\mathbf{w} \\ [B, i \mid D, p, \gamma', {\searrow} E, q, \eta \mid O, u, \gamma, P, v]\mathbf{e}\end{array}}{[B, i \mid B, i, \gamma'', {\nearrow} G, k, \eta \mid O, u, \gamma, P, v]\mathbf{e}} \quad (\mathbf{e}, C {\nearrow} F, \eta') \longmapsto (\mathbf{e}, G, \epsilon)
$$

– a una derivación de puntos especiales, con los dos casos siguientes:

$$\begin{array}{ll}(m,\,\Xi B, \xi\gamma'', a_i \dots a_n) & \overset{*}{\vdash} (\mathbf{w},\,\Xi B{\models^m} C, \xi\gamma''{\models^m}, a_j \dots a_n)\\ & \vdash (\mathbf{w},\,\Xi B{\models^m} C{\nearrow}F', \xi\gamma''{\models^m}\gamma', a_j \dots a_n)\\ & \overset{*}{\vdash} (\mathbf{w},\,\Xi B{\models^m} C{\nearrow}F'\,\Xi_1 D, \xi\gamma''{\models^m}\gamma', a_p \dots a_n)\\ & \overset{*}{\vdash} (\mathbf{e},\,\Xi B{\models^m} C{\nearrow}F'\,\Xi_1 D{\searrow}E, \xi\gamma''{\models^m}, a_q \dots a_n)\\ & \overset{*}{\vdash} (\mathbf{e},\,\Xi B{\models^m} C{\nearrow}F, \xi\gamma''{\models^m}\eta', a_k \dots a_n)\\ & \vdash (\mathbf{e},\,\Xi B{\models^m} G, \xi\gamma''{\models^m}, a_k \dots a_n)\end{array}$$

$$\cfrac{\begin{array}{l}[C,j \mid C,j,\models^m,\nearrow F,k,\eta' \mid D,p,\gamma',E,q]\mathbf{e}\\ [-,- \mid B,i,\gamma'',\models^m C,j,\models^m \mid -,-,-,-,-]\mathbf{w}\\ [-,- \mid D,p,\gamma',\searrow E,q,\models^m \mid -,-,-,-,-]\mathbf{e}\end{array}}{[-,- \mid B,i,\gamma'',\models^m G,k,\models^m \mid -,-,-,-,-]\mathbf{e}}\ (\mathbf{e}, C{\nearrow}F, \eta') \longmapsto (\mathbf{e}, G, \epsilon)$$

$$\begin{array}{ll}(\mathbf{w},\,\Xi B, \xi{\models^m}\gamma, a_i \dots a_n) & \overset{*}{\vdash} (\mathbf{w},\,\Xi B{\searrow}C, \xi{\models^m}, a_j \dots a_n)\\ & \vdash (\mathbf{w},\,\Xi B{\searrow}C{\nearrow}F', \xi{\models^m}\gamma', a_j \dots a_n)\\ & \overset{*}{\vdash} (\mathbf{w},\,\Xi B{\searrow}C{\nearrow}F'\,\Xi_1 D, \xi{\models^m}\gamma', a_p \dots a_n)\\ & \overset{*}{\vdash} (\mathbf{e},\,\Xi B{\searrow}C{\nearrow}F'\,\Xi_1 D{\searrow}E, \xi{\models^m}, a_q \dots a_n)\\ & \overset{*}{\vdash} (\mathbf{e},\,\Xi B{\searrow}C{\nearrow}F, \xi{\models^m}\eta', a_k \dots a_n)\\ & \vdash (\mathbf{e},\,\Xi B{\searrow}G, \xi{\models^m}, a_k \dots a_n)\end{array}$$

$$\cfrac{\begin{array}{l}[C,j \mid C,j,\models^m,\nearrow F,k,\eta' \mid D,p,\gamma',E,q]\mathbf{e}\\ [-,- \mid B,i,\gamma,\searrow C,j,\models^m \mid -,-,-,-,-]\mathbf{w}\\ [-,- \mid D,p,\gamma',\searrow E,q,\models^m \mid -,-,-,-,-]\mathbf{e}\end{array}}{[-,- \mid B,i,\gamma,\searrow G,k,\models^m \mid -,-,-,-,-]\mathbf{e}}\ (\mathbf{e}, C{\nearrow}F, \eta') \longmapsto (\mathbf{e}, G, \epsilon)$$

$$\begin{array}{ll}(\mathbf{w},\,\Xi B, \xi{\models^m}, a_i \dots a_n) & \overset{*}{\vdash} (\mathbf{w},\,\Xi B{\rightarrow}C, \xi{\models^m}, a_j \dots a_n)\\ & \vdash (\mathbf{w},\,\Xi B{\rightarrow}C{\nearrow}F', \xi{\models^m}\gamma', a_j \dots a_n)\\ & \overset{*}{\vdash} (\mathbf{w},\,\Xi B{\rightarrow}C{\nearrow}F'\,\Xi_1 D, \xi{\models^m}\gamma', a_p \dots a_n)\\ & \overset{*}{\vdash} (\mathbf{e},\,\Xi B{\rightarrow}C{\nearrow}F'\,\Xi_1 D{\searrow}E, \xi{\models^m}, a_q \dots a_n)\\ & \overset{*}{\vdash} (\mathbf{e},\,\Xi B{\rightarrow}C{\nearrow}F, \xi{\models^m}\eta', a_k \dots a_n)\\ & \vdash (\mathbf{e},\,\Xi B{\rightarrow}G, \xi{\models^m}, a_k \dots a_n)\end{array}$$

$$\cfrac{\begin{array}{l}[C,j \mid C,j,\models^m,\nearrow F,k,\eta' \mid D,p,\gamma',E,q]\mathbf{e}\\ [-,- \mid B,i,\models^m,\rightarrow C,j,\models^m \mid -,-,-,-,-]\mathbf{w}\\ [-,- \mid D,p,\gamma',\searrow E,q,\models^m \mid -,-,-,-,-]\mathbf{e}\end{array}}{[-,- \mid B,i,\models^m,\rightarrow G,k,\models^m \mid -,-,-,-,-]\mathbf{e}}\ (\mathbf{e}, C{\nearrow}F, \eta') \longmapsto (\mathbf{e}, G, \epsilon)$$

- Derivaciones que son el resultado de aplicar una transición $(\mathbf{e}, C{\searrow}F, \epsilon) \longmapsto (\mathbf{e}, G, \eta)$ a una derivación obtenida tras aplicar una transición $(\mathbf{w}, C, \gamma) \longmapsto (\mathbf{w}, C{\searrow}F, \epsilon)$ a una derivación de llamada, con los tres casos siguientes:

$$\begin{array}{ll}(\mathbf{w},\,\Xi M, \xi, a_m \dots a_n) & \overset{*}{\vdash} (\mathbf{w},\,\Xi M\,\Xi_1 N, \xi'\gamma''', a_t \dots a_n)\\ & \overset{*}{\vdash} (\mathbf{w},\,\Xi M\,\Xi_1 N \otimes A, \xi\gamma', a_h \dots a_n)\\ & \overset{*}{\vdash} (\mathbf{w},\,\Xi M\,\Xi_1 N \otimes A\,\Xi_2 B, \xi\gamma'\gamma\gamma'', a_i \dots a_n)\\ & \overset{*}{\vdash} (\mathbf{w},\,\Xi M\,\Xi_1 N \otimes A\,\Xi_2 B{\searrow}C, \xi\gamma'\gamma, a_j \dots a_n)\\ & \vdash (\mathbf{w},\,\Xi M\,\Xi_1 N \otimes A\,\Xi_2 B{\searrow}C{\searrow}F', \xi\gamma', a_j \dots a_n)\\ & \overset{*}{\vdash} (\mathbf{w},\,\Xi M\,\Xi_1 N \otimes A\,\Xi_2 B{\searrow}C{\searrow}F'\,\Xi_4 D, \xi\gamma', a_p \dots a_n)\\ & \overset{*}{\vdash} (\mathbf{e},\,\Xi M\,\Xi_1 N \otimes A\,\Xi_2 B{\searrow}C{\searrow}F'\,\Xi_4 D{\searrow}E, \phi, a_q \dots a_n)\\ & \overset{*}{\vdash} (\mathbf{e},\,\Xi M\,\Xi_1 N \otimes A\,\Xi_2 B{\searrow}C{\searrow}F, \phi\eta', a_k \dots a_n)\\ & \vdash (\mathbf{e},\,\Xi M\,\Xi_1 N \otimes A\,\Xi_2 B{\searrow}G, \phi\eta'\eta, a_k \dots a_n)\end{array}$$

$$\frac{\begin{array}{c}[M, m \mid C, j, \gamma, \searrow F, k, \eta' \mid D, p, \gamma', E, q]\mathbf{e} \\ [A, h \mid B, i, \gamma'', \searrow C, j, \gamma \mid -, -, -, -, -]\mathbf{w} \\ [M, m \mid N, t, \gamma''', \otimes A, h, \gamma' \mid -, -, -, -, -]\mathbf{w}\end{array}}{[A, h \mid B, i, \gamma'', \searrow G, k, \eta \mid C, j, \gamma, \searrow F, k]\mathbf{e}} \quad (\mathbf{e}, C\searrow F, \epsilon) \longmapsto (\mathbf{e}, G, \eta)$$

$$\begin{array}{rl}(\mathbf{w}, \Xi M, \xi, a_m \dots a_n) & \overset{*}{\vdash} (\mathbf{w}, \Xi M\, \Xi_1 N, \xi'\gamma''', a_t \dots a_n) \\ & \overset{*}{\vdash} (\mathbf{w}, \Xi M\, \Xi_1 N \otimes A, \xi\gamma', a_h \dots a_n) \\ & \overset{*}{\vdash} (\mathbf{w}, \Xi M\, \Xi_1 N \otimes A\, \Xi_2 B, \xi\gamma'\gamma, a_i \dots a_n) \\ & \overset{*}{\vdash} (\mathbf{w}, \Xi M\, \Xi_1 N \otimes A\, \Xi_2 B{\rightarrow}C, \xi\gamma'\gamma, a_j \dots a_n) \\ & \vdash (\mathbf{w}, \Xi M\, \Xi_1 N \otimes A\, \Xi_2 B{\rightarrow}C\searrow F', \xi\gamma', a_j \dots a_n) \\ & \overset{*}{\vdash} (\mathbf{w}, \Xi M\, \Xi_1 N \otimes A\, \Xi_2 B{\rightarrow}C\searrow F'\, \Xi_3 D, \xi\gamma', a_p \dots a_n) \\ & \overset{*}{\vdash} (\mathbf{e}, \Xi M\, \Xi_1 N \otimes A\, \Xi_2 B{\rightarrow}C\searrow F'\, \Xi_3 D\searrow E, \phi, a_q \dots a_n) \\ & \overset{*}{\vdash} (\mathbf{e}, \Xi M\, \Xi_1 N \otimes A\, \Xi_2 B{\rightarrow}C\searrow F, \phi\eta', a_k \dots a_n) \\ & \vdash (\mathbf{e}, \Xi M\, \Xi_1 N \otimes A\, \Xi_2 B{\rightarrow}G, \phi\eta'\eta, a_k \dots a_n)\end{array}$$

$$\frac{\begin{array}{c}[M, m \mid C, j, \gamma, \searrow F, k, \eta' \mid D, p, \gamma', E, q]\mathbf{e} \\ [A, h \mid B, i, \gamma, \rightarrow C, j, \gamma \mid -, -, -, -, -]\mathbf{w} \\ [M, m \mid N, t, \gamma''', \otimes A, h, \gamma' \mid -, -, -, -, -]\mathbf{w}\end{array}}{[A, h \mid B, i, \gamma, \rightarrow G, k, \eta \mid C, j, \gamma, \searrow F, k]\mathbf{e}} \quad (\mathbf{e}, C\searrow F, \epsilon) \longmapsto (\mathbf{e}, G, \eta)$$

$$\begin{array}{rl}(\mathbf{w}, \Xi M, \xi, a_m \dots a_n) & \overset{*}{\vdash} (\mathbf{w}, \Xi M\, \Xi_1 N, \xi'\gamma''', a_t \dots a_n) \\ & \overset{*}{\vdash} (\mathbf{w}, \Xi M\, \Xi_1 N \otimes B, \xi\gamma', a_h \dots a_n) \\ & \overset{0}{\vdash} (\mathbf{w}, \Xi M\, \Xi_1 N \otimes B, \xi\gamma', a_i \dots a_n) \\ & \overset{*}{\vdash} (\mathbf{w}, \Xi M\, \Xi_1 N \otimes B\nearrow C, \xi\gamma'\gamma, a_j \dots a_n) \\ & \vdash (\mathbf{w}, \Xi M\, \Xi_1 N \otimes B\nearrow C\searrow F', \xi\gamma', a_j \dots a_n) \\ & \overset{*}{\vdash} (\mathbf{w}, \Xi M\, \Xi_1 N \otimes B\nearrow C\searrow F'\, \Xi_2 D, \xi\gamma', a_p \dots a_n) \\ & \overset{*}{\vdash} (\mathbf{e}, \Xi M\, \Xi_1 N \otimes B\nearrow C\searrow F'\, \Xi_2 D\searrow E, \phi, a_q \dots a_n) \\ & \overset{*}{\vdash} (\mathbf{e}, \Xi M\, \Xi_1 N \otimes B\nearrow C\searrow F, \phi\eta', a_k \dots a_n) \\ & \vdash (\mathbf{e}, \Xi M\, \Xi_1 N \otimes B\nearrow G, \phi\eta'\eta, a_k \dots a_n)\end{array}$$

$$\frac{\begin{array}{c}[M, m \mid C, j, \gamma, \searrow F, k, \eta' \mid D, p, \gamma', E, q]\mathbf{e} \\ [B, i \mid B, i, \gamma', \nearrow C, j, \gamma \mid -, -, -, -, -]\mathbf{w} \\ [M, m \mid N, t, \gamma''', \otimes A, h, \gamma' \mid -, -, -, -, -]\mathbf{w}\end{array}}{[B, i \mid B, i, \gamma', \nearrow G, k, \eta \mid C, j, \gamma, \searrow F, k]\mathbf{e}} \quad (\mathbf{e}, C\searrow F, \epsilon) \longmapsto (\mathbf{e}, G, \eta)$$

A partir de esta lista se puede mostrar por inducción en la longitud de las derivaciones que tanto mediante la manipulación de configuraciones como mediante la manipulación de ítems se obtienen los mismos resultados. □

La complejidad espacial de la técnica de tabulación propuesta con respecto a la longitud n de la cadena de entrada es $\mathcal{O}(n^5)$ puesto que en cada ítem se almacenan 5 posiciones de la cadena de entrada. La complejidad temporal en el peor caso es $\mathcal{O}(n^7)$ y viene dada por la regla

$$\frac{\begin{array}{c}[C, j \mid C, j, \gamma, \nearrow F, k, \eta' \mid D, p, \gamma', E, q]\mathbf{e} \\ [A, h \mid B, i, \gamma'', \otimes C, j, \gamma \mid -, -, -, -, -]\mathbf{w} \\ [A, h \mid D, p, \gamma', \searrow E, q, \eta \mid O, u, \gamma, P, v]\mathbf{e}\end{array}}{[A, h \mid B, i, \gamma'', \otimes G, k, \eta \mid O, u, \gamma, P, v]\mathbf{e}} \quad (\mathbf{e}, C\nearrow F, \eta') \longmapsto (\mathbf{e}, G, \epsilon)$$

Esta regla involucra la combinación de 8 posiciones de la cadena de entrada, aunque mediante aplicación parcial sólo es preciso utilizar 7 de dichas posiciones simultáneamente, combinando el primer y tercer ítem antecedente y después combinando el resultado obtenido con el segundo ítem antecedente para obtener finalmente el ítem consecuente. Tal y como muestran De la Clergerie y Alonso Pardo en [53], esta complejidad puede reducirse a $\mathcal{O}(n^6)$ mediante el particionamiento de la regla anterior en las dos reglas siguientes:

$$\frac{\begin{array}{c}[C,j \mid C,j,\gamma,\nearrow F,k,\eta' \mid D,p,\gamma',E,q]\mathbf{e}\\ [A,h \mid D,p,\gamma',\searrow E,q,\eta \mid O,u,\gamma,P,v]\mathbf{e}\end{array}}{[[C,j,\gamma,\nearrow F,k,\eta' \mid O,u,\gamma,P,v]]\mathbf{e}} \quad (\mathbf{e}, C\nearrow F,\eta') \longmapsto (\mathbf{e},G,\epsilon)$$

$$\frac{\begin{array}{l}[[C,j,\gamma,\nearrow F,k,\eta' \mid O,u,\gamma,P,v]]\mathbf{e}\\ [A,h \mid B,i,\gamma'',\otimes C,j,\gamma \mid -,-,-,-,-]\mathbf{w}\\ [A,h \mid D,p,\gamma',\searrow E,q,\eta \mid O,u,\gamma,P,v]\mathbf{e}\end{array}}{[A,h \mid B,i,\gamma'',\otimes G,k,\eta \mid O,u,\gamma,P,v]\mathbf{e}} \quad (\mathbf{e}, C\nearrow F,\eta') \longmapsto (\mathbf{e},G,\epsilon)$$

La primera regla combina el primer y tercer ítem antecedente de la regla original, teniendo como consecuente un pseudo-ítem que permitirá transmitir parte de la información presente en los antecedentes a la segunda regla. En particular, es de destacar que la primera regla obvia los campos (A,h). Esta información es recuperada por la segunda regla mediante la combinación del pseudo-ítem con el segundo y tercer ítem antecedente de la regla original. El ítem consecuente de la segunda regla coincide con el ítem consecuente de la regla original, por lo que la aplicación combinada de las dos reglas propuestas es equivalente a la aplicación de la regla original. Sin embargo, hemos logrado reducir la complejidad temporal de $\mathcal{O}(n^7)$ a $\mathcal{O}(n^6)$, ya que que la primera regla tiene una complejidad $\mathcal{O}(n^6)$ (la posición h no interviene), la misma complejidad que presenta la segunda regla, en cuya aplicación las posiciones p y q no intervienen.

10.4 Autómatas con dos pilas ascendentes

Los autómatas con dos pilas fuertemente dirigidos pueden ser utilizados para implementar estrategias de análisis sintáctico para gramáticas de adjunción de árboles independientemente de que las adjunciones se traten de manera descendente o ascendente. Esta potencia expresiva tiene como contrapunto la complejidad espacial $\mathcal{O}(n^5)$ de la interpretación tabular para esta clase de autómatas, cuando los algoritmos tabulares de análisis sintáctico que únicamente consideran el tratamiento ascendente de la adjunción presentan una complejidad $\mathcal{O}(n^4)$ (véase el capítulo 3). Con respecto a la complejidad temporal, en los algoritmos ascendentes se obtiene $\mathcal{O}(n^6)$ de forma directa, mientras que en los autómatas con dos pilas fuertemente dirigidos se obtiene el mismo resultado tras aplicar una sofisticada optimización. Las mismas consideraciones son aplicables a las gramáticas lineales de índices con respecto al tratamiento de las pilas de índices: aquellos algoritmos que evalúan las pilas de índices de forma ascendente presentan menos complejidad espacial y temporal (véase el capítulo 4). En este contexto, surge la cuestión de si es posible diseñar una versión de los SD–2SA especializada en el tratamiento ascendente de la pila auxiliar que rivalice en cuanto a complejidad especial y temporal con los algoritmos tabulares para el análisis sintáctico de LIG y TAG.

Definimos un *autómata con dos pilas ascendente* (*Bottom-up 2–Stack Automata*, BU–2SA) como un autómata con dos pilas fuertemente dirigido sobre el cual se establece la restricción adicional de que las sesiones de la pila auxiliar deben permanecer vacías durante el modo de escritura. Más formalmente, un autómata con dos pilas ascendente se define mediante una tupla $(V_T, V_S, \$_0, \$_f, V_I, \mathcal{D}', \Theta)$ donde

- V_T es un conjunto finito de símbolos terminales.

- V_S es un conjunto finito de símbolos de la pila maestra.
- $\$_0 \in V_S$ es el símbolo inicial de la pila maestra.
- $\$_f \in V_S$ es el símbolo final de la pila maestra.
- V_I es un conjunto finito de símbolos de la pila auxiliar.
- $\mathcal{D}' = \{\models^{\mathbf{w}}, \models^{\mathbf{e}}, \rhd\}$ es el conjunto de marcas de adjunción, resultado de proyectar el conjunto $\mathcal{D} = \{\models^{\mathbf{w}}, \models^{\mathbf{e}}, \nearrow, \rightarrow, \searrow\}$ utilizado por los autómatas con dos pilas fuertemente dirigidos.
- Θ es un conjunto finito de transiciones.

Una *configuración* de un autómata con dos pilas ascendente es una tupla (m, Ξ, ξ, w), donde $m \in \{\mathbf{w}, \mathbf{e}\}$, $\Xi \in (\mathcal{D}'V_S)^*$, $\xi \in (\mathcal{D}'V_I^*)^*$ y $w \in V_T^*$. Los conceptos de *derivación* y *lenguaje aceptado* se definen de modo idéntico a como se hace para los autómatas con dos pilas fuertemente dirigidos.

El conjunto de transiciones de los autómatas con dos pilas ascendentes debe garantizar que las sesiones de la pila auxiliar permanecen vacías durante el modo de escritura, obteniéndose como resultado el siguiente juego de transiciones:

SWAP1 : Transiciones de la forma $(m, C, \epsilon) \overset{a}{\longmapsto} (m, F, \epsilon)$, donde $m \in \{\mathbf{w}, \mathbf{e}\}$, $C, F \in V_S$ y $a \in V_T \cup \epsilon$. El resultado de aplicar una transición de este tipo a una configuración $(m, \Xi C, \xi, aw)$ es una configuración $(m, \Xi F, \xi, w)$.

SWAP2 : Transiciones de la forma $(\mathbf{w}, C, \models^m) \overset{a}{\longmapsto} (\mathbf{e}, F, \models^m)$. El resultado de aplicar una transición este tipo a una configuración $(m, \Xi C, \xi\models^m, aw)$ es una configuración $(m, \Xi F, \xi\models^m, w)$. Estas transiciones son las únicas que permiten pasar del modo de escritura al modo de borrado.

$\models$**WRITE** : Transiciones de la forma $(m, C, \epsilon) \longmapsto (\mathbf{w}, C\models^m F, \models^m)$ que al ser aplicadas a una configuración $(m, \Xi C, \xi, w)$ producen una configuración $(\mathbf{w}, \Xi C\models^m F, \xi\models^m)$.

$\rhd$**WRITE** : Transiciones de la forma $(\mathbf{w}, C, \epsilon) \longmapsto (\mathbf{w}, C \rhd F, \epsilon)$ que al ser aplicadas a una transición $(\mathbf{w}, \Xi C, \xi\models^m, w)$ producen una configuración $(\mathbf{w}, \Xi C \rhd F, \xi\models^m, w)$.

$\models$**ERASE** : Transiciones de la forma $(\mathbf{e}, C\models^m F, \models^m) \longmapsto (m, G, \epsilon)$ que al ser aplicadas a una configuración $(m, \Xi C\models^m F, \xi\models^m, w)$ producen una configuración $(m, \Xi G, \xi, w)$.

$\rightarrow$**ERASE** : Transiciones de la forma $(\mathbf{e}, C \rhd F, \epsilon) \longmapsto (\mathbf{e}, G, \epsilon)$ que al ser aplicadas a una configuración $(\mathbf{e}, \Xi C \rhd F, \xi, w)$ producen una configuración $(\mathbf{e}, \Xi G, \xi, w)$.

$\nearrow$**ERASE** : Transiciones de la forma $(\mathbf{e}, C \rhd F, \eta') \longmapsto (\mathbf{e}, G, \epsilon)$ que al ser aplicadas a una configuración $(\mathbf{e}, \Xi C \rhd F, \xi\eta', w)$ producen una configuración $(\mathbf{e}, \Xi G, \xi, w)$.

$\searrow$**ERASE** : Transiciones de la forma $(\mathbf{e}, C \rhd F, \epsilon) \longmapsto (\mathbf{e}, G, \eta)$ que al ser aplicadas a una configuración $(\mathbf{e}, \Xi C \rhd F, \xi, w)$ producen una configuración $(\mathbf{e}, \Xi G, \xi\eta, w)$.

Como se puede observar, este juego de transiciones es el resultado de proyectar el conjunto $\mathcal{D}$ en el conjunto $\mathcal{D}'$. Como resultado, las marcas $\nearrow$, $\rightarrow$ y $\searrow$ son sustituidas por una única marca $\rhd$, con lo cual no se predice ninguna información acerca de la formación de la pila auxiliar durante el modo de escritura.

10.4.1 Esquemas de compilación de gramáticas lineales de índices

Al igual que en el caso de los autómatas con dos pilas fuertemente dirigidos, utilizaremos la pila maestra para almacenar no-terminales de la gramática y la pila auxiliar para almacenar los índices. El esquema de compilación genérico se define en función de los siguientes parámetros:

- $\overrightarrow{A}$, la predicción realizada sobre el no-terminal A durante la fase descendente de la estrategia de análisis.
- $\overleftarrow{A}$, la propagación de información respecto al no-terminal A durante la fase ascendente de la estrategia de análisis.

Esquema de compilación 10.3 El esquema de compilación genérico de una gramática lineal de índices en un autómata con dos pilas ascendente queda definido por el conjunto de reglas mostrado en la tabla 10.8 y por los elementos inicial $\$_0$ y final $\overleftarrow{S}$. Este esquema se puede convertir en esquemas de compilación que incorporan estrategias específicas. En la tabla 10.9 se muestran los valores que toman los diferentes parámetros para las estrategias de análisis de LIG más comunes. En dicha tabla, $\square$ indica un símbolo de pila especial que no aparece inicialmente en V_S. §

[INIT]	$(\mathbf{w}, \$_0, \epsilon) \longmapsto (\mathbf{w}, \$_0 \models^{\mathbf{w}} \nabla_{0,0}, \models^{\mathbf{w}})$	
[CALL]	$(m, \nabla_{r,s}, \epsilon) \longmapsto (\mathbf{w}, \nabla_{r,s} \models^m \overrightarrow{A_{r,s+1}}, \models^m)$	$A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\,]\Upsilon_2$
[SCALL]	$(\mathbf{w}, \nabla_{r,s}, \epsilon) \longmapsto (\mathbf{w}, \nabla_{r,s} \rhd \overrightarrow{A_{r,s+1}}, \epsilon)$	$A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\circ\circ\gamma']\Upsilon_2$
[SEL]	$(\mathbf{w}, \overrightarrow{A_{r,0}}, \epsilon) \longmapsto (\mathbf{w}, \nabla_{r,0}, \epsilon)$	$r \neq 0$
[PUB]	$(\mathbf{e}, \nabla_{r,n_r}, \epsilon) \longmapsto (\mathbf{e}, \overleftarrow{A_{r,0}}, \epsilon)$	
[RET]	$(\mathbf{e}, \nabla_{r,s} \models^m \overleftarrow{A_{r,s+1}}, \models^m) \longmapsto (m, \nabla_{r,s+1}, \epsilon)$	$A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\,]\Upsilon_2$
[SRET-1]	$(\mathbf{e}, \nabla_{r,s} \rhd \overleftarrow{A_{r,s+1}}, \epsilon) \longmapsto (\mathbf{e}, \nabla_{r,s+1}, \epsilon)$	$A_{r,0}[\circ\circ] \to \Upsilon_1 A_{r,s+1}[\circ\circ]\Upsilon_2$
[SRET-2]	$(\mathbf{e}, \nabla_{r,s} \rhd \overleftarrow{A_{r,s+1}}, \gamma') \longmapsto (\mathbf{e}, \nabla_{r,s+1}, \epsilon)$	$A_{r,0}[\circ\circ] \to \Upsilon_1 A_{r,s+1}[\circ\circ\gamma']\Upsilon_2$
[SRET-3]	$(\mathbf{e}, \nabla_{r,s} \rhd \overleftarrow{A_{r,s+1}}, \epsilon) \longmapsto (\mathbf{e}, \nabla_{r,s+1}, \gamma)$	$A_{r,0}[\circ\circ\gamma] \to \Upsilon_1 A_{r,s+1}[\circ\circ]\Upsilon_2$
[SCAN]	$(\mathbf{w}, \overrightarrow{A_{r,0}}, \models^m) \overset{a}{\longmapsto} (\mathbf{e}, \overleftarrow{A_{r,0}}, \models^m)$	$A_{r,0}[\,] \to a$

Tabla 10.8: Reglas del esquema de compilación genérico de LIG en BU–2SA

10.4.2 Esquemas de compilación para gramáticas de adjunción de árboles

Al igual que en el caso de los autómatas con dos pilas fuertemente dirigidos, utilizaremos la pila maestra para almacenar los nodos de los árboles elementales y la pila auxiliar para almacenar

estrategia-CF	$\overrightarrow{A_{r,s+1}}$	$\overleftarrow{A_{r,s+1}}$
Ascendente	$\Box$	$A_{r,s+1}$
Earley	$\overline{A_{r,s+1}}$	$\overline{\overline{A_{r,s+1}}}$
Descendente	$A_{r,s+1}$	$\Box$

Tabla 10.9: Parámetros del esquema de compilación genérico de LIG en SD–2SA

la pila de adjunciones pendiente en cada nodo. El esquema de compilación genérico mediante la parametrización del flujo de información de las fases de llamada y retorno. Los parámetros a considerar son:

- $\overrightarrow{N^{\gamma}_{r,s}}$, la información predicha acerca del nodo $N^{\gamma}_{r,s}$.
- $\overleftarrow{N^{\gamma}_{r,s}}$, la información propagada acerca del nodo $N^{\gamma}_{r,s}$.

Esquema de compilación 10.4 El esquema de compilación genérico de una gramática de adjunción de árboles en un autómata con dos pilas ascendente queda definido por el conjunto de reglas mostrado en la tabla 10.10 y los elementos inicial $\$_0$ y final $\overleftarrow{\top^{\alpha}}$, con $\alpha \in \boldsymbol{I}$. Este esquema se puede convertir en esquemas de compilación para diferentes estrategias de análisis según los valores que tomen los parámetros, tal y como se indica en la tabla 10.11. §

10.4.3 BU–2SA y los lenguajes de adjunción de árboles

Los lenguajes aceptados por los autómatas con dos pilas ascendentes coinciden con los lenguajes de adjunción de árboles. Para demostrar esta aseveración definiremos y demostraremos los dos teoremas siguientes.

Teorema 10.6 *Los lenguajes adjunción de árboles son un subconjunto de los lenguajes aceptados por la clase de los autómatas con dos pilas ascendentes.*

<u>Demostración</u>:

Por el esquema de compilación de TAG en BU–2SA presentado anteriormente, a partir de cualquier gramática de adjunción de árboles es posible construir un SD–2SA que acepta el lenguaje reconocido por dicha gramática. $\Box$

Teorema 10.7 *La clase de los lenguajes aceptados por los autómatas con dos pilas ascendentes es un subconjunto de los lenguajes de adjunción de árboles.*

<u>Demostración</u>:

Mostraremos que para todo SD–2SA existe una gramática lineal de índices tal que el lenguaje reconocido por la gramática coincide con el lenguaje aceptado por el autómata.

Sea $\mathcal{A} = (V_T, V_S, \$_0, \$_f, V_I, \mathcal{D}', \Theta)$ un autómata lineal de índices ascendente. Construiremos una gramática lineal de índices $\mathcal{L} = (V_T, V_N, V_I, S, P)$. El conjunto V_N de no-terminales estará formado por pares $\langle E, B\rangle$ tal que $A, B \in V_S$. Para que $\mathcal{L}$ reconozca el lenguaje aceptado por $\mathcal{A}$ el conjunto de producciones en P ha de construirse a partir de las transiciones en Θ de la siguiente manera:

[INIT]	$(\mathbf{w}, \$_0, \epsilon) \longmapsto (\mathbf{w}, \$_0 {\models}^{\mathbf{w}} \nabla^{\alpha}_{0,0}, \models^{\mathbf{w}})$	$\alpha \in \boldsymbol{I}$
[CALL]	$(m, \nabla^{\gamma}_{r,s}, \epsilon) \longmapsto (\mathbf{w}, \nabla^{\gamma}_{r,s} {\models}^m \overrightarrow{N^{\gamma}_{r,s+1}}, \models^m)$	$N^{\gamma}_{r,s+1} \not\in \text{espina}(\gamma)$, $\mathbf{nil} \in \text{adj}(N_{r,s+1})$
[SCALL]	$(\mathbf{w}, \nabla^{\gamma}_{r,s}, \epsilon) \longmapsto (\mathbf{w}, \nabla^{\gamma}_{r,s} \rhd \overrightarrow{N^{\gamma}_{r,s+1}}, \epsilon)$	$N^{\gamma}_{r,s+1} \in \text{espina}(\gamma)$, $\mathbf{nil} \in \text{adj}(N_{r,s+1})$
[SEL]	$(\mathbf{w}, \overrightarrow{N^{\gamma}_{r,0}}, \epsilon) \longmapsto (\mathbf{w}, \nabla^{\gamma}_{r,0}, \epsilon)$	$r \neq 0$
[PUB1]	$(\mathbf{e}, \nabla^{\gamma}_{r,n_r}, \epsilon) \longmapsto (\mathbf{e}, \overleftarrow{N^{\gamma}_{r,0}}, \epsilon)$	
[PUB2]	$(\mathbf{w}, \nabla^{\gamma}_{r,n_r}, \models^m) \longmapsto (\mathbf{e}, \overleftarrow{N^{\gamma}_{r,0}}, \models^m)$	
[RET]	$(\mathbf{e}, \nabla^{\gamma}_{r,s} {\models}^m \overleftarrow{N^{\gamma}_{r,s+1}}, \models^m) \longmapsto (m, \nabla^{\gamma}_{r,s+1}, \epsilon)$	$N^{\gamma}_{r,s+1} \not\in \text{espina}(\gamma)$, $\mathbf{nil} \in \text{adj}(N_{r,s+1})$
[SRET]	$(\mathbf{e}, \nabla^{\gamma}_{r,s} \rhd \overleftarrow{N^{\gamma}_{r,s+1}}, \epsilon) \longmapsto (\mathbf{e}, \nabla^{\gamma}_{r,s+1}, \epsilon)$	$N^{\gamma}_{r,s+1} \in \text{espina}(\gamma)$, $\mathbf{nil} \in \text{adj}(N_{r,s+1})$
[SCAN]	$(\mathbf{w}, \overrightarrow{N^{\gamma}_{r,0}}, \models^m) \overset{a}{\longmapsto} (\mathbf{e}, \overleftarrow{N^{\gamma}_{r,0}}, \models^m)$	$N^{\gamma}_{r,0}[\,] \rightarrow a$
[ACALL]	$(\mathbf{w}, \nabla^{\gamma}_{r,s}, \epsilon) \longmapsto (\mathbf{w}, \nabla^{\gamma}_{r,s} \rhd \overrightarrow{\top^{\beta}}, \epsilon)$	$\beta \in \text{adj}(N^{\gamma}_{r,s+1})$
[ARET]	$(\mathbf{e}, \nabla^{\gamma}_{r,s} \rhd \overleftarrow{\top^{\beta}}, N^{\gamma}_{r,s+1}) \longmapsto (\mathbf{e}, \nabla^{\gamma}_{r,s+1}, \epsilon)$	$\beta \in \text{adj}(N^{\gamma}_{r,s+1})$
[FCALL]	$(\mathbf{w}, \nabla^{\beta}_{f,0}, \epsilon) \longmapsto (\mathbf{w}, \nabla^{\beta}_{f,0} \rhd \overrightarrow{N^{\gamma}_{r,s+1}}, \epsilon)$	$N^{\beta}_{f,0} = \mathbf{F}^{\beta}$, $\beta \in \text{adj}(N^{\gamma}_{r,s+1})$
[FRET]	$(\mathbf{w}, \nabla^{\beta}_{f,0} \rhd \overleftarrow{N^{\gamma}_{r,s+1}}, \epsilon) \longmapsto (\mathbf{e}, \nabla^{\beta}_{f,1}, N^{\gamma}_{r,s+1})$	$N^{\beta}_{f,0} = \mathbf{F}^{\beta}$, $\beta \in \text{adj}(N^{\gamma}_{t,s+1})$

Tabla 10.10: Reglas del esquema de compilación genérico de TAG en BU–2SA

- Para toda transición $(m, C, \epsilon) \overset{a}{\longmapsto} (m, F, \epsilon)$ y para todo $E \in P$ creamos una producción

$$\langle E, F\rangle[\circ\circ] \rightarrow \langle E, C\rangle[\circ\circ]\ a$$

- Para toda transición $(\mathbf{w}, C, \models^m) \overset{a}{\longmapsto} (\mathbf{e}, F, \models^m)$ y para todo $E \in P$ creamos una producción

$$\langle E, F\rangle[\circ\circ] \rightarrow \langle E, C\rangle[\circ\circ]\ a$$

- Para toda transición $(\mathbf{w}, C, \epsilon) \overset{a}{\longmapsto} (\mathbf{w}, C \rhd F, \epsilon)$ creamos una producción

$$\langle C, F\rangle[\,] \rightarrow \epsilon$$

- Para toda transición $(m, C, \epsilon) \longmapsto (\mathbf{w}, C {\models}^m F, \models^m)$ creamos una producción

$$\langle C, F\rangle[\,] \rightarrow \epsilon$$

- Para toda transición $(\mathbf{e}, C {\models}^m F, \models^m) \longmapsto (m, G, \epsilon)$ y para todo $E \in P$ creamos una producción

$$\langle E, G\rangle[\circ\circ] \rightarrow \langle E, C\rangle[\circ\circ]\ \langle C, F\rangle[\,]$$

- Para toda transición $(\mathbf{e}, C \rhd F, \epsilon) \longmapsto (\mathbf{e}, G, \epsilon)$ y para todo $E \in P$ creamos una producción

$$\langle E, G\rangle[\circ\circ] \rightarrow \langle E, C\rangle[\,]\ \langle C, F\rangle[\circ\circ]$$

Estrategia-CF	$\overrightarrow{N^{\gamma}_{r,s+1}}$	$\overleftarrow{N^{\gamma}_{r,s+1}}$
Ascendente	$\square$	$N^{\gamma}_{r,s+1}$
Earley	$\overline{N^{\gamma}_{r,s+1}}$	$\overline{\overline{N^{\gamma}_{r,s+1}}}$
Descendente	$N^{\gamma}_{r,s+1}$	$\square$

Tabla 10.11: Parámetros del esquema de compilación genérico de TAG en BU–2SA

- Para toda transición $(\mathbf{e}, C \rhd F, \epsilon) \longmapsto (\mathbf{e}, G, \gamma')$ y para todo $E \in P$ creamos una producción
$$\langle E, G\rangle[\circ\circ\gamma'] \to \langle E, C\rangle[\,]\ \langle C, F\rangle[\circ\circ]$$
- Para toda transición $(\mathbf{e}, C \rhd F, \gamma) \longmapsto (\mathbf{e}, G, \epsilon)$ y para todo $E \in P$ creamos una producción
$$\langle E, G\rangle[\circ\circ] \to \langle E, C\rangle[\,]\ \langle C, F\rangle[\circ\circ\gamma]$$

Con respecto al axioma de la gramática, tenemos que $S = \langle \$_0, \$_f\rangle$.

Las derivaciones de la gramática lineal de índices obtenida y del autómata con dos pilas ascendente cumplen las siguientes propiedades:

- $\langle E, B\rangle[\alpha] \stackrel{*}{\Rightarrow} w$, con $\alpha \neq \epsilon$, si y sólo si $(\mathbf{w}, E, \epsilon, w) \stackrel{*}{\vdash} (\mathbf{e}, E \rhd B, \alpha, \epsilon)$.
- $\langle E, B\rangle[\,] \stackrel{*}{\Rightarrow} w$ si y sólo si $(m, E, \epsilon, w) \stackrel{*}{\vdash} (\mathbf{e}, E {\models^m} B, \models^m, \epsilon)$.

La veracidad de estas dos propiedades se demuestra mediante inducción mutua en la longitud de las respectivas derivaciones:

- Si una derivación $(\mathbf{w}, E, \epsilon, w) \stackrel{*}{\vdash} (\mathbf{e}, E \rhd B, \alpha, \epsilon)$, con $\alpha \neq \epsilon$, es el resultado de aplicar la secuencia $t_1, \ldots, t_m$ de transiciones en Θ, entonces existe una secuencia $p_1, \ldots, p_m$ de producciones en P tal que p_i es una producción creada a partir de t_i y la derivación derecha $\langle E, B\rangle[\alpha] \stackrel{*}{\Rightarrow} w$ resultado de aplicar $p_m, \ldots, p_1$ reconoce w.
- Si una derivación $(m, E, \epsilon, w) \stackrel{*}{\vdash} (\mathbf{e}, E {\models^m} B, \models^m, \epsilon)$ es el resultado de aplicar la secuencia $t_1, \ldots, t_m$ de transiciones en Θ, entonces existe una secuencia $p_1, \ldots, p_m$ de producciones en P tal que p_i es una producción creada a partir de t_i y la derivación derecha $\langle E, B\rangle[\,] \stackrel{*}{\Rightarrow} w$ resultado de aplicar $p_m, \ldots, p_1$ reconoce w.
- Si una derivación derecha $\langle E, B\rangle[\alpha] \stackrel{*}{\Rightarrow} w$, con $\alpha \neq \epsilon$, reconoce la cadena w como resultado de aplicar la secuencia $p_1, \ldots, p_m$ de producciones en P, entonces existe una secuencia de transiciones $t_1, \ldots, t_m$ tal que la p_i es una producción creada a partir de t_i y la derivación $(\mathbf{w}, E, \epsilon, w) \stackrel{*}{\vdash} (\mathbf{e}, E \rhd B, \alpha, , \epsilon)$ es el resultado de aplicar la secuencia de transiciones $t_m, \ldots, t_1$.
- Si una derivación derecha $\langle E, B\rangle[\ ] \stackrel{*}{\Rightarrow} w$ reconoce la cadena w como resultado de aplicar la secuencia $p_1, \ldots, p_m$ de producciones en P, entonces existe una secuencia de transiciones $t_1, \ldots, t_m$ tal que la p_i es una producción creada a partir de t_i y la derivación $(m, E, \epsilon, w) \stackrel{*}{\vdash} (\mathbf{e}, E {\models^m} B, \models^m, \epsilon)$ es el resultado de aplicar la secuencia de transiciones $t_m, \ldots, t_1$.

$\square$

Ejemplo 10.2 El autómata con dos pilas ascendente de la tabla 10.12 acepta el lenguaje $\{a^n b^n c^n d^n \mid n > 0\}$. En la tabla 10.13 se muestra la derivación para la cadena de entrada *aaabbbcccddd* en dicho autómata. La primera columna indica la transición aplicada, la segunda señala el modo del autómata en ese momento, la tercera muestra el contenido de la pila maestra, la cuarta muestra el contenido de la pila auxiliar y la quinta muestra la parte de la cadena de entrada que resta por leer.

Resulta interesante comparar este autómata con el autómata con dos pilas fuertemente dirigido definido en la tabla 10.1. Denotaremos por $|X|$ el número de elementos X apilados en la pila maestra. En el caso del SD–2SA de la tabla 10.1 se comprueba que $|A| = |B|$ durante el modo de escritura mediante la manipulación de los elementos γ almacenados en la pila auxiliar. En el modo de borrado, las transiciones que extraían elementos de la pila maestra se encargaban de comprobar que $|B| = |C|$ y que $|A| = |D|$ al tiempo que mediante la manipulación de los elementos η en la pila se comprobaba que $|C| = |D|$. En el caso del BU–2SA de la tabla 10.12 no se realiza la comprobación de que $|A| = |B|$ durante el modo de escritura. Sin embargo, durante el modo de borrado se comprueba que $|B| = |C|$, $|A| = |D|$ y $|C| = |D|$ y en tal caso, por transitividad, se cumple que $|A| = |B|$.

En el caso del SD–2SA de la tabla 10.1, la comprobación de que $|A| = |B|$ durante el modo de llamada le permite mantener la propiedad del prefijo válido. En cambio, el BU–2SA de la tabla 10.12 no garantiza dicha propiedad puesto que la no gramaticalidad de la cadena de entrada *aabbbcccddd* sólo puede determinarse después de leer *aabbbcccdd*, que no es prefijo de ninguna cadena válida del lenguaje aceptado por el autómata, y detectar que $|A| \neq |D|$.

En la tabla 10.14 se muestran las producciones de la gramática lineal de índices obtenida a partir de las transiciones del autómata con dos pilas ascendente. Utilizamos Γ para representar cualquier símbolo de pila. La derivación de la cadena *aaabbbcccddd* con esta gramática se muestra en la tabla 10.15, en la cual la primera columna indica la producción aplicada. ¶

10.4.4 Tabulación

Las derivaciones de los autómatas con dos pilas ascendentes se pueden clasificar en los tres tipos siguientes:

Derivaciones de llamada. Son aquellas que dependen únicamente de la cima de la pila maestra ya que la sesión en la cima de la pila auxiliar está vacía. Presentan la forma:

$$(\mathbf{w},\ \Xi B, \xi {\models}^{m'}, a_i \ldots a_n) \ \overset{*}{\vdash} (m,\ \Xi B \rhd C, \xi {\models}^{m'}, a_j \ldots a_n)$$

donde $\mathbf{w} \leq m$ y tanto B como C pertenecen a la misma sesión. En la derivación, se puede consultar B pero no se permite la consulta ni la alteración de Ξ y ξ. En la figura 10.4 se muestra una representación gráfica de este tipo de derivaciones[2].

Para cualquier $\Xi' \in (\mathcal{D}' V_S)^*$ y $\xi' \in ({\models}^x V_I^*)^*$ tal que $x \in \{\mathbf{w}, \mathbf{e}\}$ y el número de sesiones en Ξ' y ξ' coincide, se cumple

$$(\mathbf{w},\ \Xi' B, \xi' {\models}^{m'}, a_i \ldots a_n) \ \overset{*}{\vdash} (m,\ \Xi' B \rhd C, \xi' {\models}^{m'}, a_j \ldots a_n)$$

[2]Puesto que m no es necesariamente $\mathbf{w}$, este tipo de derivaciones pueden extenderse más allá del modo de escritura y por tanto la denominación de *derivaciones de llamada* puede parecer contraproducente. Sin embargo, con el fin de mantener la homogeneidad con el resto de los modelos de autómata, hemos optado por mantener tal denominación.

$$
\begin{array}{ll}
(a) & (\mathbf{w}, \$_0, \epsilon) \longmapsto (\mathbf{w}, \$_0 \models^{\mathbf{w}} A, \models^{\mathbf{w}}) \\
(b) & (\mathbf{w}, A, \epsilon) \overset{a}{\longmapsto} (\mathbf{w}, A', \epsilon) \\
(c) & (\mathbf{w}, A', \epsilon) \longmapsto (\mathbf{w}, A' \rhd A, \epsilon) \\
(d) & (\mathbf{w}, A', \epsilon) \overset{b}{\longmapsto} (\mathbf{w}, B', \epsilon) \\
(e) & (\mathbf{w}, B', \epsilon) \longmapsto (\mathbf{w}, B' \rhd B, \epsilon) \\
(f) & (\mathbf{w}, B, \epsilon) \overset{b}{\longmapsto} (\mathbf{w}, B', \epsilon) \\
(g) & (\mathbf{w}, B', \models^{m}) \overset{c}{\longmapsto} (\mathbf{e}, C', \models^{m}) \\
(h) & (\mathbf{e}, B' \rhd C', \epsilon) \longmapsto (\mathbf{e}, C, \eta) \\
(i) & (\mathbf{e}, C, \epsilon) \overset{c}{\longmapsto} (\mathbf{e}, C', \epsilon) \\
(j) & (\mathbf{e}, C', \epsilon) \overset{d}{\longmapsto} (\mathbf{e}, D', \epsilon) \\
(k) & (\mathbf{e}, A' \rhd D') \longmapsto (\mathbf{e}, D, \epsilon) \\
(l) & (\mathbf{e}, D, \epsilon) \overset{d}{\longmapsto} (\mathbf{e}, D', \epsilon) \\
(m) & (\mathbf{e}, D', \epsilon) \longmapsto (\mathbf{e}, \$_f, \epsilon)
\end{array}
$$

Tabla 10.12: Transiciones del BU–2SA que acepta $\{a^n b^n c^n d^n \mid n > 0\}$

	$\mathbf{w}$	$\models^{\mathbf{w}}\$_0$	$\models^{\mathbf{w}}$	$aaabbbcccddd$
(a)	$\mathbf{w}$	$\models^{\mathbf{w}}\$_0 \ \models^{\mathbf{w}}A$	$\models^{\mathbf{w}}\models^{\mathbf{w}}$	$aaabbbcccddd$
(b)	$\mathbf{w}$	$\models^{\mathbf{w}}\$_0 \ \models^{\mathbf{w}}A'$	$\models^{\mathbf{w}}\models^{\mathbf{w}}$	$aabbbcccddd$
(c)	$\mathbf{w}$	$\models^{\mathbf{w}}\$_0 \ \models^{\mathbf{w}}A' \rhd A$	$\models^{\mathbf{w}}\models^{\mathbf{w}}$	$aabbbcccddd$
(b)	$\mathbf{w}$	$\models^{\mathbf{w}}\$_0 \ \models^{\mathbf{w}}A' \rhd A'$	$\models^{\mathbf{w}}\models^{\mathbf{w}}$	$abbbcccddd$
(c)	$\mathbf{w}$	$\models^{\mathbf{w}}\$_0 \ \models^{\mathbf{w}}A' \rhd A' \rhd A$	$\models^{\mathbf{w}}\models^{\mathbf{w}}$	$abbbcccddd$
(b)	$\mathbf{w}$	$\models^{\mathbf{w}}\$_0 \ \models^{\mathbf{w}}A' \rhd A' \rhd A'$	$\models^{\mathbf{w}}\models^{\mathbf{w}}$	$bbbcccddd$
(d)	$\mathbf{w}$	$\models^{\mathbf{w}}\$_0 \ \models^{\mathbf{w}}A' \rhd A' \rhd B'$	$\models^{\mathbf{w}}\models^{\mathbf{w}}$	$bbcccddd$
(e)	$\mathbf{w}$	$\models^{\mathbf{w}}\$_0 \ \models^{\mathbf{w}}A' \rhd A' \rhd B' \rhd B$	$\models^{\mathbf{w}}\models^{\mathbf{w}}$	$bbcccddd$
(f)	$\mathbf{w}$	$\models^{\mathbf{w}}\$_0 \ \models^{\mathbf{w}}A' \rhd A' \rhd B' \rhd B'$	$\models^{\mathbf{w}}\models^{\mathbf{w}}$	$bcccddd$
(e)	$\mathbf{w}$	$\models^{\mathbf{w}}\$_0 \ \models^{\mathbf{w}}A' \rhd A' \rhd B' \rhd B' \rhd B$	$\models^{\mathbf{w}}\models^{\mathbf{w}}$	$bcccddd$
(f)	$\mathbf{w}$	$\models^{\mathbf{w}}\$_0 \ \models^{\mathbf{w}}A' \rhd A' \rhd B' \rhd B' \rhd B'$	$\models^{\mathbf{w}}\models^{\mathbf{w}}$	$cccddd$
(g)	$\mathbf{e}$	$\models^{\mathbf{w}}\$_0 \ \models^{\mathbf{w}}A' \rhd A' \rhd B' \rhd B' \rhd C'$	$\models^{\mathbf{w}}\models^{\mathbf{w}}$	$ccddd$
(h)	$\mathbf{e}$	$\models^{\mathbf{w}}\$_0 \ \models^{\mathbf{w}}A' \rhd A' \rhd B' \rhd C$	$\models^{\mathbf{w}}\models^{\mathbf{w}} \eta$	$ccddd$
(i)	$\mathbf{e}$	$\models^{\mathbf{w}}\$_0 \ \models^{\mathbf{w}}A' \rhd A' \rhd B' \rhd C'$	$\models^{\mathbf{w}}\models^{\mathbf{w}} \eta$	$cddd$
(h)	$\mathbf{e}$	$\models^{\mathbf{w}}\$_0 \ \models^{\mathbf{w}}A' \rhd A' \rhd C$	$\models^{\mathbf{w}}\models^{\mathbf{w}} \eta\,\eta$	$cddd$
(i)	$\mathbf{e}$	$\models^{\mathbf{w}}\$_0 \ \models^{\mathbf{w}}A' \rhd A' \rhd C'$	$\models^{\mathbf{w}}\models^{\mathbf{w}} \eta\,\eta$	ddd
(j)	$\mathbf{e}$	$\models^{\mathbf{w}}\$_0 \ \models^{\mathbf{w}}A' \rhd A' \rhd D'$	$\models^{\mathbf{w}}\models^{\mathbf{w}} \eta\,\eta$	dd
(k)	$\mathbf{e}$	$\models^{\mathbf{w}}\$_0 \ \models^{\mathbf{w}}A' \rhd D$	$\models^{\mathbf{w}}\models^{\mathbf{w}} \eta$	dd
(l)	$\mathbf{e}$	$\models^{\mathbf{w}}\$_0 \ \models^{\mathbf{w}}A' \rhd D'$	$\models^{\mathbf{w}}\models^{\mathbf{w}} \eta$	d
(k)	$\mathbf{e}$	$\models^{\mathbf{w}}\$_0 \ \models^{\mathbf{w}}D$	$\models^{\mathbf{w}}\models^{\mathbf{w}}$	d
(l)	$\mathbf{e}$	$\models^{\mathbf{w}}\$_0 \ \models^{\mathbf{w}}D'$	$\models^{\mathbf{w}}\models^{\mathbf{w}}$	
(m)	$\mathbf{e}$	$\models^{\mathbf{w}}\$_0 \ \models^{\mathbf{w}}\$_f$	$\models^{\mathbf{w}}\models^{\mathbf{w}}$	

Tabla 10.13: Configuraciones del BU–2SA para la cadena de entrada $aaabbbcccddd$

(a)	$\langle \$_0, A\rangle[\,] \rightarrow \epsilon$
(b)	$\langle \Gamma, A'\rangle[\circ\circ] \rightarrow \langle \Gamma, A\rangle[\circ\circ]\ \ a$
(c)	$\langle A', A\rangle[\,] \rightarrow \epsilon$
(d)	$\langle \Gamma, B'\rangle[\circ\circ] \rightarrow \langle \Gamma, A'\rangle[\circ\circ]\ \ b$
(e)	$\langle B', B\rangle[\,] \rightarrow \epsilon$
(f)	$\langle \Gamma, B'\rangle[\circ\circ] \rightarrow \langle \Gamma, B\rangle[\circ\circ]\ \ b$
(g)	$\langle \Gamma, C'\rangle[\circ\circ] \rightarrow \langle \Gamma, B'\rangle[\circ\circ]\ \ c$
(h)	$\langle \Gamma, C\rangle[\circ\circ\eta] \rightarrow \langle \Gamma, B'\rangle[\,]\ \ \langle B', C'\rangle[\circ\circ]$
(i)	$\langle \Gamma, C'\rangle[\circ\circ] \rightarrow \langle \Gamma, C\rangle[\circ\circ]\ \ c$
(j)	$\langle \Gamma, D'\rangle[\circ\circ] \rightarrow \langle \Gamma, C'\rangle[\circ\circ]\ \ d$
(k)	$\langle \Gamma, D\rangle[\circ\circ] \rightarrow \langle \Gamma, A'\rangle[\,]\ \ \langle A', D'\rangle[\circ\circ]$
(l)	$\langle \Gamma, D'\rangle[\circ\circ] \rightarrow \langle \Gamma, D\rangle[\circ\circ]\ \ d$
(m)	$\langle \Gamma, \$_f\rangle[\circ\circ] \rightarrow \langle \Gamma, D'\rangle[\circ\circ]$

Tabla 10.14: Producciones de la LIG obtenida a partir del BU–2SA

		$\langle \$_0, \$_f\rangle[\,]$
(m)	$\Rightarrow$	$\langle \$_0, D'\rangle[\,]$
(l)	$\Rightarrow$	$\langle \$_0, D\rangle[\,]\ \ d$
(k)	$\Rightarrow$	$\langle \$_0, A'\rangle[\,]\ \ \langle A', D'\rangle[\,]\ \ d$
(l)	$\Rightarrow$	$\langle \$_0, A'\rangle[\,]\ \ \langle A', D\rangle[\,]\ \ dd$
(k)	$\Rightarrow$	$\langle \$_0, A'\rangle[\,]\ \ \langle A', A'\rangle[\,]\ \ \langle A', D'\rangle[\,]\ \ dd$
(j)	$\Rightarrow$	$\langle \$_0, A'\rangle[\,]\ \ \langle A', A'\rangle[\,]\ \ \langle A', C'\rangle[\,]\ \ ddd$
(i)	$\Rightarrow$	$\langle \$_0, A'\rangle[\,]\ \ \langle A', A'\rangle[\,]\ \ \langle A', C\rangle[\,]\ \ cddd$
(h)	$\Rightarrow$	$\langle \$_0, A'\rangle[\,]\ \ \langle A', A'\rangle[\,]\ \ \langle A', B'\rangle[\,]\ \ \langle B', C'\rangle[\,]\ \ cddd$
(i)	$\Rightarrow$	$\langle \$_0, A'\rangle[\,]\ \ \langle A', A'\rangle[\,]\ \ \langle A', B'\rangle[\,]\ \ \langle B', C\rangle[\,]\ \ ccddd$
(h)	$\Rightarrow$	$\langle \$_0, A'\rangle[\,]\ \ \langle A', A'\rangle[\,]\ \ \langle A', B'\rangle[\,]\ \ \langle B', B'\rangle[\,]\ \ \langle B', C'\rangle[\,]\ \ ccddd$
(g)	$\Rightarrow$	$\langle \$_0, A'\rangle[\,]\ \ \langle A', A'\rangle[\,]\ \ \langle A', B'\rangle[\,]\ \ \langle B', B'\rangle[\,]\ \ \langle B', B'\rangle[\,]\ \ cccddd$
(f)	$\Rightarrow$	$\langle \$_0, A'\rangle[\,]\ \ \langle A', A'\rangle[\,]\ \ \langle A', B'\rangle[\,]\ \ \langle B', B'\rangle[\,]\ \ \langle B', B\rangle[\,]\ \ bcccddd$
(e)	$\Rightarrow$	$\langle \$_0, A'\rangle[\,]\ \ \langle A', A'\rangle[\,]\ \ \langle A', B'\rangle[\,]\ \ \langle B', B'\rangle[\,]\ \ bcccddd$
(f)	$\Rightarrow$	$\langle \$_0, A'\rangle[\,]\ \ \langle A', A'\rangle[\,]\ \ \langle A', B'\rangle[\,]\ \ \langle B', B\rangle[\,]\ \ bbcccddd$
(e)	$\Rightarrow$	$\langle \$_0, A'\rangle[\,]\ \ \langle A', A'\rangle[\,]\ \ \langle A', B'\rangle[\,]\ \ bbcccddd$
(d)	$\Rightarrow$	$\langle \$_0, A'\rangle[\,]\ \ \langle A', A'\rangle[\,]\ \ \langle A', A'\rangle[\,]\ \ bbbcccddd$
(b)	$\Rightarrow$	$\langle \$_0, A'\rangle[\,]\ \ \langle A', A'\rangle[\,]\ \ \langle A', A\rangle[\,]\ \ abbbcccddd$
(c)	$\Rightarrow$	$\langle \$_0, A'\rangle[\,]\ \ \langle A', A'\rangle[\,]\ \ abbbcccddd$
(b)	$\Rightarrow$	$\langle \$_0, A'\rangle[\,]\ \ \langle A', A\rangle[\,]\ \ aabbbcccddd$
(c)	$\Rightarrow$	$\langle \$_0, A'\rangle[\,]\ \ aabbbcccddd$
(b)	$\Rightarrow$	$\langle \$_0, A\rangle[\,]\ \ aaabbbcccddd$
(a)	$\Rightarrow$	$aaabbbcccddd$

Tabla 10.15: Derivación en LIG de la cadena *aaabbbcccddd*

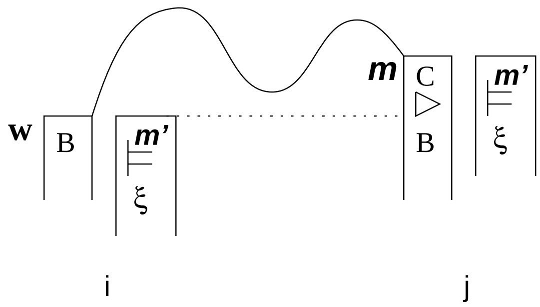

Figura 10.4: Derivaciones de llamada en BU-2SA

por lo que este tipo de derivaciones puede ser representado de manera compacta por los correspondientes ítems de la forma

$$[B, i, \rhd C, j, \models^{m'} \mid -, -, -, -, -]m$$

Derivaciones de retorno. Son aquellas que se inician en una sesión en modo de escritura y terminan en la misma sesión en modo de borrado con una sesión no vacía en la cima de la pila auxiliar. Presentan la forma

$$\begin{aligned}(\mathbf{w},\, \Xi B, \xi{\models^m}, a_i \ldots a_n) \;&\overset{*}{\vdash}_{d_1} (\mathbf{w},\, \Xi B\, \Xi_1 D, \xi{\models^m}, a_p \ldots a_n)\\ &\overset{*}{\vdash}_{d_2} (\mathbf{e},\, \Xi B\, \Xi_1 D \rhd E, \xi{\models^m}\phi, a_q \ldots a_n)\\ &\overset{*}{\vdash}_{d_3} (\mathbf{e},\, \Xi B \otimes C, \xi{\models^m}\phi\eta, a_j \ldots a_n)\end{aligned}$$

donde $\otimes \in \mathcal{D}'$, $\phi \in V_I^*$, $\eta \in V_I$ y tanto B como D, E y C pertenecen a la misma sesión. La subderivación d_1 puede consultar B pero no puede consultar ni alterar Ξ y ξ. La subderivación d_2 puede consultar D pero no puede consultar ni alterar $\Xi B\, \Xi_1$ ni ξ. Finalmente, la pila $\xi{\models^m}$ es la misma en toda la derivación. Es importante señalar que si $\otimes = {\models^m}$, entonces la derivación no puede conducir a la aceptación de la cadena de entrada puesto que no es posible eliminar ϕ de la pila auxiliar. Sin embargo, tal tipo de derivaciones puede obtenerse y por tanto es necesario considerarlas. En la figura 10.5 se muestra una representación gráfica de este tipo de derivaciones.

Para cualquier $\Xi' \in (\mathcal{D}'V_S)^*$ y $\xi' \in ({\models^x} V_I^*)^*$ tal que $x \in \{\mathbf{w}, \mathbf{e}\}$, el número de sesiones en Ξ' y ξ' coincide y existe una derivación

$$(\mathbf{w}, D, \xi{\models^m}, a_p \ldots a_n) \overset{*}{\vdash} (\mathbf{e}, D \rhd E, \xi{\models^m}\phi, a_q \ldots a_n)$$

se cumple que

$$\begin{aligned}(\mathbf{w},\, \Xi' B, \xi'{\models^m}, a_i \ldots a_n) \;&\overset{*}{\vdash}_{d_1} (\mathbf{w},\, \Xi' B\, \Xi_1 D, \xi'{\models^m}, a_p \ldots a_n)\\ &\overset{*}{\vdash}_{d_2} (\mathbf{e},\, \Xi' B\, \Xi_1 D \rhd E, \xi'{\models^m}\phi, a_q \ldots a_n)\\ &\overset{*}{\vdash}_{d_3} (\mathbf{e},\, \Xi' B \otimes C, \xi'{\models^m}\phi\eta, a_j \ldots a_n)\end{aligned}$$

Ello posibilita que las derivaciones de retorno puedan ser representadas por ítems de la forma

$$[B, i, \otimes C, j, \eta \mid D, p, \models^m, E, q]\mathbf{e}$$

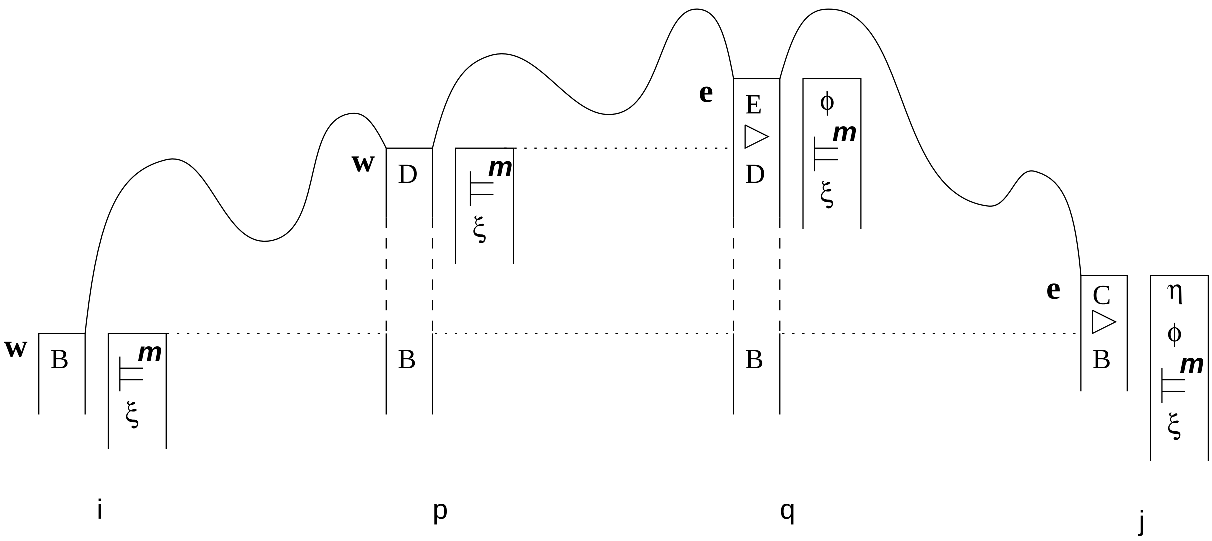

Figura 10.5: Derivaciones de retorno en BU-2SA

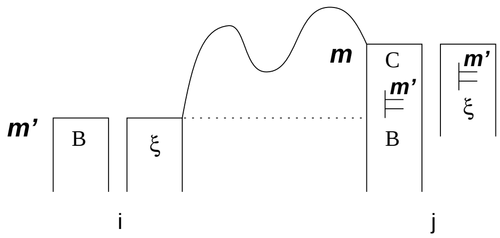

Figura 10.6: Derivaciones de puntos especiales en BU-2SA

Derivaciones de puntos especiales. Son aquellas derivaciones que finalizan en una sesión mínima, con un único elemento el la pila maestra y ninguna en la auxiliar. Estas configuraciones se tienen típicamente cuando se acaba de crear una nueva sesión y cuando se está a punto de finalizar una sesión. Presentan la forma

$$(m', \Xi B, \xi, a_i \dots a_n) \ \stackrel{*}{\vdash} (m, \Xi B {\models^{m'}} C, \xi {\models^{m'}}, a_j \dots a_n)$$

Este tipo de derivaciones se muestra gráficamente en la figura 10.6.

Para cualquier $\Xi' \in (\mathcal{D}' V_S)^*$ y $\xi' \in (\models^x V_I^*)^*$ tal que $x \in \{\mathbf{w}, \mathbf{e}\}$ y el número de sesiones en Ξ' y ξ' coincide, se cumple

$$(m', \Xi' B, \xi', a_i \dots a_n) \ \stackrel{*}{\vdash} (m, \Xi' B {\models^{m'}} C, \xi' {\models^{m'}}, a_j \dots a_n)$$

por lo que podemos utilizar ítems de la siguientes formas para representar este tipo de derivaciones:

$$[B, i, \models^{m'} C, j, \models^{m'} \mid -, -, -, -, -]m$$

Los ítems se combinan mediante las reglas descritas en la tabla 10.16, a partir del ítem inicial

$$[-, -, \models^{\mathbf{w}} \$_0, 0, \models^{\mathbf{w}} \mid -, -, -, -, -]\mathbf{w}$$

La aceptación de la cadena se entrada $a_1 \dots a_n$ se indica mediante la presencia de ítems de la forma

$$[\$_0, 0, \models^{\mathbf{w}} \$_f, n, \models^{\mathbf{w}} \mid -, -, -, -, -]\mathbf{e}$$

$$\frac{[B,i,\otimes C,j,\eta \mid D,p,\models^{m'},E,q]m}{[B,i,\otimes F,k,\eta \mid D,p,\models^{m'},E,q]m} \quad \begin{array}{l}(m,C,\epsilon) \overset{a}{\longmapsto} (m,F,\epsilon),\\ k=j \text{ si } a=\epsilon,\ k=j+1 \text{ si } a\in V_T\end{array}$$

$$\frac{[B,i,\otimes C,j,\eta \mid D,p,\models^{m'},E,q]m}{[C,j,\models^m F,j,\models^m \mid -,-,-,-,-]\mathbf{w}} \quad (m,C,\epsilon) \longmapsto (\mathbf{w},C\models^m F,\models^m)$$

$$\frac{[B,i,\otimes C,j,\models^m \mid -,-,-,-,-]\mathbf{w}}{[C,j,\rhd F,j,\models^m \mid -,-,-,-,-]\mathbf{w}} \quad (\mathbf{w},C,\epsilon) \longmapsto (\mathbf{w},C\rhd F,\epsilon)$$

$$\frac{[B,i,\otimes C,j,\models^m \mid -,-,-,-,-]\mathbf{w}}{[B,i,\otimes F,k,\models^m \mid -,-,-,-,-]\mathbf{e}} \quad \begin{array}{l}(\mathbf{w},C,\models^m) \overset{a}{\longmapsto} (\mathbf{e},F,\models^m),\\ k=j \text{ si } a=\epsilon,\ k=j+1 \text{ si } a\in V_T\end{array}$$

$$\frac{\begin{array}{c}[C,j,\models^m F,k,\models^m \mid -,-,-,-,-]\mathbf{e}\\ [B,i,\otimes C,j,\eta \mid D,p,\models^{m'},E,q]m\end{array}}{[B,i,\otimes G,k,\eta \mid D,p,\models^{m'},E,q]m} \quad (\mathbf{e},C\models^m F,\models^m) \longmapsto (m,G,\epsilon)$$

$$\frac{\begin{array}{c}[C,j,\rhd F,k,\eta \mid D,p,\models^m,E,q]\mathbf{e}\\ [B,i,\otimes C,j,\models^m \mid -,-,-,-,-]\mathbf{w}\end{array}}{[B,i,\otimes G,k,\eta \mid D,p,\models^m,E,q]\mathbf{e}} \quad (\mathbf{e},C\rhd F,\epsilon) \longmapsto (\mathbf{e},G,\epsilon)$$

$$\frac{\begin{array}{c}[C,j,\rhd F,k,\eta' \mid D,p,\models^m,E,q]\mathbf{e}\\ [B,i,\otimes_1 C,j,\models^m \mid -,-,-,-,-]\mathbf{w}\\ [D,p,\otimes_2 E,q,\eta \mid O,u,\models^m,P,v]\mathbf{e}\end{array}}{[B,i,\otimes_1 G,k,\eta \mid O,u,\models^m,P,v]\mathbf{e}} \quad (\mathbf{e},C\rhd F,\eta') \longmapsto (\mathbf{e},G,\epsilon)$$

$$\frac{\begin{array}{c}[C,j,\rhd F,k,\eta' \mid D,p,\models^m,E,q]\mathbf{e}\\ [B,i,\otimes C,j,\models^m \mid -,-,-,-,-]\mathbf{w}\end{array}}{[B,i,\otimes G,k,\eta \mid C,j,\models^m,F,k]\mathbf{e}} \quad (\mathbf{e},C\rhd F,\epsilon) \longmapsto (\mathbf{e},G,\eta)$$

Tabla 10.16: Reglas de combinación de ítems en BU-2SA

Teorema 10.8 *La manipulación de configuraciones mediante la aplicación de transiciones en los autómatas con dos pilas ascendentes es equivalente a la manipulación de ítems mediante las reglas de combinación de la tabla 10.16.*

Demostración:

Mostraremos que para toda derivación existe una regla de combinación que produce un ítem que representa de forma compacta dicha derivación y que toda regla de combinación se corresponde con una derivación válida del autómata. Para ello detallaremos todas las posibles derivaciones, junto con las reglas de combinación ítems correspondientes, en la siguiente lista.

- Derivaciones que son el resultado de aplicar una transición $(m,C,\epsilon) \overset{a}{\longmapsto} (m,F,\epsilon)$

– a una derivación de llamada:

$$\begin{array}{ll}(\mathbf{w},\, \Xi B, \xi\models^{m'}, a_i \dots a_n) & \overset{*}{\vdash} (m,\, \Xi B \rhd C, \xi\models^{m'}, a_j \dots a_n) \\ & \vdash (m,\, \Xi B \rhd F, \xi\models^{m'}, a_k \dots a_n)\end{array}$$

$$\frac{[B, i, \rhd C, j, \models^{m'} \mid -,-,-,-,-]m}{[B, i, \rhd F, k, \models^{m'} \mid -,-,-,-,-]m} \quad (m, C, \epsilon) \overset{a}{\longmapsto} (m, F, \epsilon)$$

– a una derivación de retorno:

$$\begin{array}{ll}(\mathbf{w},\, \Xi B, \xi\models^{m}, a_i \dots a_n) & \overset{*}{\vdash} (\mathbf{w},\, \Xi B\, \Xi_1 D, \xi\models^{m}, a_p \dots a_n) \\ & \overset{*}{\vdash} (\mathbf{e},\, \Xi B\, \Xi_1 D \rhd E, \xi\models^{m}\phi, a_q \dots a_n) \\ & \overset{*}{\vdash} (\mathbf{e},\, \Xi B \otimes C, \xi\models^{m}\phi\eta, a_j \dots a_n) \\ & \vdash (\mathbf{e},\, \Xi B \otimes F, \xi\models^{m}\phi\eta, a_k \dots a_n)\end{array}$$

$$\frac{[B, i, \otimes C, j, \eta \mid D, p, \models^{m}, E, q]\mathbf{e}}{[B, i, \otimes F, k, \eta \mid D, p, \models^{m}, E, q]\mathbf{e}} \quad (\mathbf{e}, C, \epsilon) \overset{a}{\longmapsto} (\mathbf{e}, F, \epsilon)$$

– a una derivación de puntos especiales:

$$\begin{array}{ll}(m',\, \Xi B, \xi, a_i \dots a_n) & \overset{*}{\vdash} (m,\, \Xi B\models^{m'} C, \xi\models^{m'}, a_j \dots a_n) \\ & \vdash (m,\, \Xi B\models^{m'} F, \xi\models^{m'}, a_k \dots a_n)\end{array}$$

$$\frac{[B, i, \models^{m'} C, j, \models^{m'} \mid -,-,-,-,-]m}{[B, i, \models^{m'} F, k, \models^{m'} \mid -,-,-,-,-]m} \quad (m, C, \epsilon) \overset{a}{\longmapsto} (m, F, \epsilon)$$

donde $k = j$ si $a = \epsilon$ y $k = j + 1$ si $a \in V_T$:

- Derivaciones que son el resultado de aplicar una transición de tipo $(m, C, \epsilon) \longmapsto (\mathbf{w}, C\models^{m} F, \models^{m})$

– a una derivación de llamada:

$$\begin{array}{ll}(\mathbf{w},\, \Xi B, \xi\models^{m'}, a_i \dots a_n) & \overset{*}{\vdash} (m,\, \Xi B \rhd C, \xi\models^{m'}, a_j \dots a_n) \\ & \vdash (\mathbf{w},\, \Xi B \rhd C\models^{m} F, \xi\models^{m'}\models^{m}, a_j \dots a_n)\end{array}$$

$$\frac{[B, i, \rhd C, j, \models^{m'} \mid -,-,-,-,-]m}{[C, j, \models^{m} F, j, \models^{m} \mid -,-,-,-,-]\mathbf{w}} \quad (m, C, \epsilon) \longmapsto (\mathbf{w}, C\models^{m} F, \models^{m})$$

– a una derivación de retorno:

$$\begin{array}{ll}(\mathbf{w},\, \Xi B, \xi\models^{m'}, a_i \dots a_n) & \overset{*}{\vdash} (\mathbf{w},\, \Xi B\, \Xi_1 D, \xi\models^{m'}, a_p \dots a_n) \\ & \overset{*}{\vdash} (\mathbf{e},\, \Xi B\, \Xi_1 D \rhd E, \xi\models^{m'}\phi, a_q \dots a_n) \\ & \overset{*}{\vdash} (\mathbf{e},\, \Xi B \otimes C, \xi\models^{m'}\phi\eta, a_j \dots a_n) \\ & \vdash (\mathbf{w},\, \Xi B \otimes C\models^{\mathbf{e}} F, \xi\models^{m'}\phi\eta\models^{\mathbf{e}}, a_j \dots a_n)\end{array}$$

$$\frac{[B, i, \otimes C, j, \eta \mid D, p, \models^{m'}, E, q]\mathbf{e}}{[C, j, \models^{\mathbf{e}} F, j, \models^{\mathbf{e}} \mid -,-,-,-,-]\mathbf{w}} \quad (\mathbf{e}, C, \epsilon) \longmapsto (\mathbf{w}, C\models^{\mathbf{e}} F, \models^{\mathbf{e}})$$

– a una derivación de puntos especiales:

$$\begin{array}{ll}(m',\, \Xi B, \xi, a_i \dots a_n) & \overset{*}{\vdash} (m,\, \Xi B\models^{m'} C, \xi\models^{m'}, a_j \dots a_n) \\ & \vdash (\mathbf{w},\, \Xi B\models^{m'} C\models^{m} F, \xi\models^{m'}\models^{m}, a_j \dots a_n)\end{array}$$

$$\frac{[B, i, \models^{m'} C, j, \models^{m'} \mid -,-,-,-,-]m}{[C, j, \models^{m} F, j, \models^{m} \mid -,-,-,-,-]\mathbf{w}} \quad (m, C, \epsilon) \longmapsto (\mathbf{w}, C\models^{m} F, \models^{m})$$

- Derivaciones que son el resultado de aplicar una transición de tipo $(\mathbf{w}, C, \epsilon) \longmapsto (\mathbf{w}, C \rhd F, \epsilon)$
  - a una derivación de llamada:

$$\begin{array}{rl}(\mathbf{w},\, \Xi B, \xi{\models^m}, a_i \ldots a_n) & \overset{*}{\vdash} (\mathbf{w},\, \Xi B \rhd C, \xi{\models^m}, a_j \ldots a_n) \\ & \vdash (\mathbf{w},\, \Xi B \rhd C \rhd F, \xi{\models^m}, a_j \ldots a_n)\end{array}$$

$$\frac{[B, i, \rhd C, j, \models^m \mid -,-,-,-,-]\mathbf{w}}{[C, j, \rhd F, j, \models^m \mid -,-,-,-,-]\mathbf{w}} \quad (\mathbf{w}, C, \epsilon) \longmapsto (\mathbf{w}, C \rhd F, \epsilon)$$

  - a una derivación de puntos especiales:

$$\begin{array}{rl}(m,\, \Xi B, \xi, a_i \ldots a_n) & \overset{*}{\vdash} (\mathbf{w},\, \Xi B{\models^m} C, \xi{\models^m}, a_j \ldots a_n) \\ & \vdash (\mathbf{w},\, \Xi B{\models^m} C \rhd F, \xi{\models^m}, a_j \ldots a_n)\end{array}$$

$$\frac{[B, i, \models^m C, j, \models^m \mid -,-,-,-,-]\mathbf{w}}{[C, j, \rhd F, j, \models^m \mid -,-,-,-,-]\mathbf{w}} \quad (\mathbf{w}, C, \epsilon) \longmapsto (\mathbf{w}, C \rhd F, \epsilon)$$

- Derivaciones que son el resultado de aplicar una transición de tipo $(\mathbf{w}, C, \models^m) \overset{a}{\longmapsto} (\mathbf{e}, F, \models^m)$
  - a una derivación de llamada:

$$\begin{array}{rl}(\mathbf{w},\, \Xi B, \xi{\models^m}, a_i \ldots a_n) & \overset{*}{\vdash} (\mathbf{w},\, \Xi B \rhd C, \xi{\models^m}, a_j \ldots a_n) \\ & \vdash (\mathbf{e},\, \Xi B \rhd F, \xi{\models^m}, a_k \ldots a_n)\end{array}$$

$$\frac{[B, i, \rhd C, j, \models^m \mid -,-,-,-,-]\mathbf{w}}{[B, i, \rhd F, k, \models^m \mid -,-,-,-,-]\mathbf{e}} \quad (\mathbf{w}, C, \models^m) \overset{a}{\longmapsto} (\mathbf{e}, F, \models^m)$$

  - a una derivación de puntos especiales:

$$\begin{array}{rl}(m,\, \Xi B, \xi, a_i \ldots a_n) & \overset{*}{\vdash} (\mathbf{w},\, \Xi B{\models^m} C, \xi{\models^m}, a_j \ldots a_n) \\ & \vdash (\mathbf{e},\, \Xi B{\models^m} F, \xi{\models^m}, a_k \ldots a_n)\end{array}$$

$$\frac{[B, i, \models^m C, j, \models^m \mid -,-,-,-,-]\mathbf{w}}{[B, i, \models^m F, k, \models^m \mid -,-,-,-,-]\mathbf{e}} \quad (\mathbf{w}, C, \models^m) \overset{a}{\longmapsto} (\mathbf{e}, F, \models^m)$$

  donde $k = j$ si $a = \epsilon$ y $k = j + 1$ si $a \in V_T$:

- Derivaciones que son el resultado de aplicar una transición de tipo $(\mathbf{e}, C{\models^m} F, \models^m) \longmapsto (m, G, \epsilon)$ a una derivación obtenida tras aplicar una transición $(m, C, \epsilon) \longmapsto (\mathbf{w}, C{\models^m} F', \models^m)$
  - a una derivación de llamada:

$$\begin{array}{rl}(\mathbf{w},\, \Xi B, \xi{\models^{m'}}, a_i \ldots a_n) & \overset{*}{\vdash} (m,\, \Xi B \rhd C, \xi{\models^{m'}}, a_j \ldots a_n) \\ & \vdash (\mathbf{w},\, \Xi B \rhd C{\models^m} F', \xi{\models^{m'}}{\models^m}, a_j \ldots a_n) \\ & \overset{*}{\vdash} (\mathbf{e},\, \Xi B \rhd C{\models^m} F, \xi{\models^{m'}}{\models^m}, a_k \ldots a_n) \\ & \vdash (m,\, \Xi B \rhd G, \xi{\models^{m'}}, a_k \ldots a_n)\end{array}$$

$$\frac{\begin{array}{c}[C, j, \models^m F, k, \models^m \mid -,-,-,-,-]\mathbf{e} \\ [B, i, \rhd C, j, \models^{m'} \mid -,-,-,-,-]m\end{array}}{[B, i, \rhd G, k, \models^{m'} \mid -,-,-,-,-]m} \quad (\mathbf{e}, C{\models^{\mathbf{w}}} F, \models^{\mathbf{w}}) \longmapsto (\mathbf{w}, G, \epsilon)$$

  - a una derivación de retorno:

$$\begin{array}{rl}(\mathbf{w},\, \Xi B, \xi{\models^{m'}}, a_i \ldots a_n) & \overset{*}{\vdash} (\mathbf{w},\, \Xi B\, \Xi_1 D, \xi{\models^{m'}}, a_p \ldots a_n) \\ & \overset{*}{\vdash} (\mathbf{e},\, \Xi B\, \Xi_1 D \rhd E, \xi{\models^{m'}} \phi, a_q \ldots a_n) \\ & \overset{*}{\vdash} (\mathbf{e},\, \Xi B \otimes C, \xi{\models^{m'}} \phi\eta, a_j \ldots a_n) \\ & \vdash (\mathbf{w},\, \Xi B \otimes C{\models^{\mathbf{e}}} F', \xi{\models^{m'}} \phi\eta{\models^{\mathbf{e}}}, a_j \ldots a_n) \\ & \overset{*}{\vdash} (\mathbf{e},\, \Xi B \otimes C{\models^{\mathbf{e}}} F, \xi{\models^{m'}} \phi\eta{\models^{\mathbf{e}}}, a_k \ldots a_n) \\ & \vdash (\mathbf{e},\, \Xi B \otimes G, \xi{\models^{m'}} \phi\eta, a_k \ldots a_n)\end{array}$$

$$\frac{\begin{array}{c}[C,j,\models^{\mathbf{e}}F,k,\models^{\mathbf{e}} \mid -,-,-,-,-]\mathbf{e}\\ [B,i,\otimes C,j,\eta \mid D,p,\models^{m'},E,q]\mathbf{e}\end{array}}{[B,i,\otimes G,k,\eta \mid D,p,\models^{m'},E,q]\mathbf{e}}\quad (\mathbf{e},C\models^{\mathbf{e}}F,\models^{\mathbf{e}}) \longmapsto (\mathbf{e},G,\epsilon)$$

– a una derivación de puntos especiales:

$$\begin{array}{rl}(m',\,\Xi B,\xi,a_i\dots a_n) & \overset{*}{\vdash} (m,\,\Xi B\models^{m'}C,\xi\models^{m'},a_j\dots a_n)\\ & \vdash (\mathbf{w},\,\Xi B\models^{m'}C\models^{m}F',\xi\models^{m'}\models^{m},a_j\dots a_n)\\ & \overset{*}{\vdash} (\mathbf{e},\,\Xi B\models^{m'}C\models^{m}F,\xi\models^{m'}\models^{m},a_k\dots a_n)\\ & \vdash (m,\,\Xi B\models^{m'}G,\xi\models^{m'},a_k\dots a_n)\end{array}$$

$$\frac{\begin{array}{c}[C,j,\models^{m}F,k,\models^{m} \mid -,-,-,-,-]\mathbf{e}\\ [B,i,\models^{m'}C,j,\models^{m'} \mid -,-,-,-,-]m\end{array}}{[B,i,\models^{m'}G,k,\models^{m'} \mid -,-,-,-,-]m}\quad (\mathbf{e},C\models^{m}F,\models^{m}) \longmapsto (m,G,\epsilon)$$

- Derivaciones que son el resultado de aplicar una transición de tipo $(\mathbf{e},C \rhd F,\epsilon) \longmapsto (\mathbf{e},G,\epsilon)$ a una derivación obtenida tras aplicar una transición $(\mathbf{w},C,\epsilon) \longmapsto (\mathbf{w},C \rhd F',\epsilon)$
  – a una derivación de llamada:

$$\begin{array}{rl}(\mathbf{w},\,\Xi B,\xi\models^{m},a_i\dots a_n) & \overset{*}{\vdash} (\mathbf{w},\,\Xi B \rhd C,\xi\models^{m},a_j\dots a_n)\\ & \vdash (\mathbf{w},\,\Xi B \rhd C \rhd F',\xi\models^{m},a_j\dots a_n)\\ & \overset{*}{\vdash} (\mathbf{w},\,\Xi B \rhd C \rhd F'\,\Xi_1 D,\xi\models^{m},a_p\dots a_n)\\ & \overset{*}{\vdash} (\mathbf{e},\,\Xi B \rhd C \rhd F'\,\Xi_1 D \rhd E,\xi\models^{m}\phi,a_q\dots a_n)\\ & \overset{*}{\vdash} (\mathbf{e},\,\Xi B \rhd C \rhd F,\xi\models^{m}\phi\eta,a_k\dots a_n)\\ & \vdash (\mathbf{e},\,\Xi B \rhd G,\xi\models^{m}\phi\eta,a_k\dots a_n)\end{array}$$

$$\frac{\begin{array}{c}[C,j,\rhd F,k,\eta \mid D,p,\models^{m},E,q]\mathbf{e}\\ [B,i,\rhd C,j,\models^{m} \mid -,-,-,-,-]\mathbf{w}\end{array}}{[B,i,\rhd G,k,\eta \mid D,p,\models^{m},E,q]\mathbf{e}}\quad (\mathbf{e},C \rhd F,\epsilon) \longmapsto (\mathbf{e},G,\epsilon)$$

  – a una derivación de puntos especiales:

$$\begin{array}{rl}(m,\,\Xi B,\xi,a_i\dots a_n) & \overset{*}{\vdash} (\mathbf{w},\,\Xi B\models^{m}C,\xi\models^{m},a_j\dots a_n)\\ & \vdash (\mathbf{w},\,\Xi B\models^{m}C \rhd F',\xi\models^{m},a_j\dots a_n)\\ & \overset{*}{\vdash} (\mathbf{e},\,\Xi B\models^{m}C \rhd F,\xi\models^{m},a_k\dots a_n)\\ & \vdash (\mathbf{e},\,\Xi B\models^{m}G,\xi\models^{m},a_k\dots a_n)\end{array}$$

$$\frac{\begin{array}{c}[C,j,\rhd F,k,\models^{m} \mid -,-,-,-,-]\mathbf{e}\\ [B,i,\models^{m}C,j,\models^{m} \mid -,-,-,-,-]\mathbf{w}\end{array}}{[B,i,\models^{m}G,k,\models^{m} \mid -,-,-,-,-]\mathbf{e}}\quad (\mathbf{e},C \rhd F,\epsilon) \longmapsto (\mathbf{e},G,\epsilon)$$

- Derivaciones que son el resultado de aplicar una transición de tipo $(\mathbf{e},C \rhd F,\eta') \longmapsto (\mathbf{e},G,\epsilon)$ a una derivación obtenida tras aplicar una transición $(\mathbf{w},C,\epsilon) \longmapsto (\mathbf{w},C \rhd F',\epsilon)$
  – a una derivación de llamada, con los dos casos siguientes:

$$\begin{array}{rl}(\mathbf{w},\,\Xi B,\xi\models^{m},a_i\dots a_n) & \overset{*}{\vdash} (\mathbf{w},\,\Xi B \rhd C,\xi\models^{m},a_j\dots a_n)\\ & \vdash (\mathbf{w},\,\Xi B \rhd C \rhd F',\xi\models^{m},a_j\dots a_n)\\ & \overset{*}{\vdash} (\mathbf{w},\,\Xi B \rhd C \rhd F'\,\Xi_1 D,\xi\models^{m},a_p\dots a_n)\\ & \overset{*}{\vdash} (\mathbf{w},\,\Xi B \rhd C \rhd F'\,\Xi_1 D\,\Xi_2 O,\xi\models^{m},a_u\dots a_n)\\ & \overset{*}{\vdash} (\mathbf{e},\,\Xi B \rhd C \rhd F'\,\Xi_1 D\,\Xi_2 O \rhd P,\xi\models^{m}\phi\eta,a_u\dots a_n)\\ & \overset{*}{\vdash} (\mathbf{e},\,\Xi B \rhd C \rhd F'\,\Xi_1 D \rhd E,\xi\models^{m}\phi\eta,a_q\dots a_n)\\ & \overset{*}{\vdash} (\mathbf{e},\,\Xi B \rhd C \rhd F,\xi\models^{m}\phi\eta\eta',a_k\dots a_n)\\ & \vdash (\mathbf{e},\,\Xi B \rhd G,\xi\models^{m}\phi\eta,a_k\dots a_n)\end{array}$$

$$\frac{\begin{array}{l}[C,j,\triangleright F,k,\eta' \mid D,p,\models^m,E,q]\mathbf{e}\\ [B,i,\triangleright C,j,\models^m \mid -,-,-,-,-]\mathbf{w}\\ [D,p,\triangleright E,q,\eta \mid O,u,\models^m,P,v]\mathbf{e}\end{array}}{[B,i,\triangleright G,k,\eta \mid O,u,\models^m,P,v]\mathbf{e}}\ (\mathbf{e},C\triangleright F,\eta')\longmapsto(\mathbf{e},G,\epsilon)$$

$$\begin{array}{ll}(\mathbf{w},\,\Xi B,\xi\models^m,a_i\dots a_n) & \overset{*}{\vdash}(\mathbf{w},\,\Xi B\triangleright C,\xi\models^m,a_j\dots a_n)\\ & \vdash(\mathbf{w},\,\Xi B\triangleright C\triangleright F',\xi\models^m,a_j\dots a_n)\\ & \overset{*}{\vdash}(\mathbf{w},\,\Xi B\triangleright C\triangleright F'\,\Xi D,\xi\models^m,a_p\dots a_n)\\ & \overset{*}{\vdash}(\mathbf{e},\,\Xi B\triangleright C\triangleright F'\,\Xi D\triangleright E,\xi\models^m,a_q\dots a_n)\\ & \overset{*}{\vdash}(\mathbf{e},\,\Xi B\triangleright C\triangleright F,\xi\models^m\eta',a_k\dots a_n)\\ & \vdash(\mathbf{e},\,\Xi B\triangleright G,\xi\models^m,a_k\dots a_n)\end{array}$$

$$\frac{\begin{array}{l}[C,j,\triangleright F,k,\eta' \mid D,p,\models^m,E,q]\mathbf{e}\\ [B,i,\triangleright C,j,\models^m \mid -,-,-,-,-]\mathbf{w}\\ [D,p,\triangleright E,q,\models^m \mid -,-,-,-,-]\mathbf{e}\end{array}}{[B,i,\triangleright G,k,\models^m \mid -,-,-,-,-]\mathbf{e}}\ (\mathbf{e},C\triangleright F,\eta')\longmapsto(\mathbf{e},G,\epsilon)$$

– a una derivación de puntos especiales, con los dos casos siguientes:

$$\begin{array}{ll}(m,\,\Xi B,\xi,a_i\dots a_n) & \overset{*}{\vdash}(\mathbf{w},\,\Xi B\models^m C,\xi\models^m,a_j\dots a_n)\\ & \vdash(\mathbf{w},\,\Xi B\models^m C\triangleright F',\xi\models^m,a_j\dots a_n)\\ & \overset{*}{\vdash}(\mathbf{w},\,\Xi B\models^m C\triangleright F'\,\Xi_1 D,\xi\models^m,a_p\dots a_n)\\ & \overset{*}{\vdash}(\mathbf{e},\,\Xi B\models^m C\triangleright F'\,\Xi_1 D\triangleright E,\xi\models^m,a_q\dots a_n)\\ & \overset{*}{\vdash}(\mathbf{e},\,\Xi B\models^m C\triangleright F,\xi\models^m\eta',a_k\dots a_n)\\ & \vdash(\mathbf{e},\,\Xi B\models^m G,\xi\models^m,a_k\dots a_n)\end{array}$$

$$\frac{\begin{array}{l}[C,j,\triangleright F,k,\eta' \mid D,p,\models^m,E,q]\mathbf{e}\\ [B,i,\models^m C,j,\models^m \mid -,-,-,-,-]\mathbf{w}\\ [D,p,\triangleright E,q,\eta \mid -,-,-,-,-]\mathbf{e}\end{array}}{[B,i,\models^m G,k,\eta \mid -,-,-,-,-]\mathbf{e}}\ (\mathbf{e},C\triangleright F,\eta')\longmapsto(\mathbf{e},G,\epsilon)$$

$$\begin{array}{ll}(m,\,\Xi B,\xi,a_i\dots a_n) & \overset{*}{\vdash}(\mathbf{w},\,\Xi B\models^m C,\xi\models^m,a_j\dots a_n)\\ & \vdash(\mathbf{w},\,\Xi B\models^m C\triangleright F',\xi\models^m,a_j\dots a_n)\\ & \overset{*}{\vdash}(\mathbf{w},\,\Xi B\models^m C\triangleright F'\,\Xi_1 D,\xi\models^m,a_p\dots a_n)\\ & \overset{*}{\vdash}(\mathbf{w},\,\Xi B\models^m C\triangleright F'\,\Xi_1 D\,\Xi O,\xi\models^m,a_u\dots a_n)\\ & \overset{*}{\vdash}(\mathbf{e},\,\Xi B\models^m C\triangleright F'\,\Xi_1 D\,\Xi O\triangleright P,\xi\models^m\phi,a_v\dots a_n)\\ & \overset{*}{\vdash}(\mathbf{e},\,\Xi B\models^m C\triangleright F'\,\Xi_1 D\triangleright E,\xi\models^m\phi\eta,a_q\dots a_n)\\ & \overset{*}{\vdash}(\mathbf{e},\,\Xi B\models^m C\triangleright F,\xi\models^m\phi\eta\eta',a_k\dots a_n)\\ & \vdash(\mathbf{e},\,\Xi B\models^m G,\xi\models^m\phi\eta,a_k\dots a_n)\end{array}$$

$$\frac{\begin{array}{l}[C,j,\triangleright F,k,\eta' \mid D,p,\models^m,E,q]\mathbf{e}\\ [B,i,\models^m C,j,\models^m \mid -,-,-,-,-]\mathbf{w}\\ [D,p,\triangleright E,q,\eta \mid O,u,\models^m,P,v]\mathbf{e}\end{array}}{[B,i,\models^m G,k,\eta \mid O,u,\models^m,P,v]\mathbf{e}}\ (\mathbf{e},C\triangleright F,\eta')\longmapsto(\mathbf{e},G,\epsilon)$$

En este caso la configuración resultante no puede conducir a una configuración final porque la pila auxiliar contiene en su cima una sesión no vacía que es imposible eliminar.

- Derivaciones que son el resultado de aplicar una transición de tipo $(\mathbf{e},C\triangleright F,\epsilon)\longmapsto(\mathbf{e},G,\eta)$ a una derivación obtenida tras aplicar una transición $(\mathbf{w},C,\epsilon)\longmapsto(\mathbf{w},C\triangleright F',\epsilon)$

– a una derivación de llamada, con los dos casos siguientes:

$$\begin{array}{ll}(\mathbf{w}, \Xi B, \xi \models^m, a_i \dots a_n) & \overset{*}{\vdash} (\mathbf{w}, \Xi B \rhd C, \xi \models^m, a_j \dots a_n) \\ & \vdash (\mathbf{w}, \Xi B \rhd C \rhd F', \xi \models^m, a_j \dots a_n) \\ & \overset{*}{\vdash} (\mathbf{w}, \Xi B \rhd C \rhd F' \, \Xi_1 D, \xi \models^m, a_p \dots a_n) \\ & \overset{*}{\vdash} (\mathbf{e}, \Xi B \rhd C \rhd F' \, \Xi_1 D \rhd E, \xi \models^m \phi, a_q \dots a_n) \\ & \overset{*}{\vdash} (\mathbf{e}, \Xi B \rhd C \rhd F, \xi \models^m \phi \eta', a_k \dots a_n) \\ & \vdash (\mathbf{e}, \Xi B \rhd G, \xi \models^m \phi \eta' \eta, a_k \dots a_n)\end{array}$$

$$\frac{\begin{array}{c}[C, j, \rhd F, k, \eta' \mid D, p, \models^m, E, q]\mathbf{e} \\ [B, i, \rhd C, j, \models^m \mid -, -, -, -, -]\mathbf{w}\end{array}}{[B, i, \rhd G, k, \eta \mid C, j, \models^m, F, k]\mathbf{e}} \quad (\mathbf{e}, C \rhd F, \epsilon) \longmapsto (\mathbf{e}, G, \eta)$$

$$\begin{array}{ll}(\mathbf{w}, \Xi B, \xi \models^m, a_i \dots a_n) & \overset{*}{\vdash} (\mathbf{w}, \Xi B \rhd C, \xi \models^m, a_j \dots a_n) \\ & \vdash (\mathbf{w}, \Xi B \rhd C \rhd F', \xi \models^m, a_j \dots a_n) \\ & \overset{*}{\vdash} (\mathbf{e}, \Xi B \rhd C \rhd F, \xi \models^m, a_k \dots a_n) \\ & \vdash (\mathbf{e}, \Xi B \rhd G, \xi \models^m \eta, a_k \dots a_n)\end{array}$$

$$\frac{\begin{array}{c}[C, j, \rhd F, k, \models^m \mid -, -, -, -, -]\mathbf{e} \\ [B, i, \rhd C, j, \models^m \mid -, -, -, -, -]\mathbf{w}\end{array}}{[B, i, \rhd G, k, \eta \mid C, j, \models^m, F, k]\mathbf{e}} \quad (\mathbf{e}, C \rhd F, \epsilon) \longmapsto (\mathbf{e}, G, \eta)$$

– a una derivación de puntos especiales, con los dos casos siguientes:

$$\begin{array}{ll}(m, \Xi B, \xi, a_i \dots a_n) & \overset{*}{\vdash} (\mathbf{w}, \Xi B \models^m C, \xi \models^m, a_j \dots a_n) \\ & \vdash (\mathbf{w}, \Xi B \models^m C \rhd F', \xi \models^m, a_j \dots a_n) \\ & \overset{*}{\vdash} (\mathbf{w}, \Xi B \models^m C \rhd F' \, \Xi_1 D, \xi \models^m, a_p \dots a_n) \\ & \overset{*}{\vdash} (\mathbf{e}, \Xi B \models^m C \rhd F' \, \Xi_1 D \rhd E, \xi \models^m \phi, a_q \dots a_n) \\ & \overset{*}{\vdash} (\mathbf{e}, \Xi B \models^m C \rhd F, \xi \models^m \phi \eta', a_k \dots a_n) \\ & \vdash (\mathbf{e}, \Xi B \models^m G, \xi \models^m \phi \eta' \eta, a_k \dots a_n)\end{array}$$

$$\frac{\begin{array}{c}[C, j, \rhd F, k, \eta' \mid D, p, \models^m, E, q]\mathbf{e} \\ [B, i, \models^m C, j, \models^m \mid -, -, -, -, -]\mathbf{w}\end{array}}{[B, i, \models^m G, k, \eta \mid C, j, \models^m, F, k]\mathbf{e}} \quad (\mathbf{e}, C \rhd F, \epsilon) \longmapsto (\mathbf{e}, G, \eta)$$

$$\begin{array}{ll}(m, \Xi B, \xi, a_i \dots a_n) & \overset{*}{\vdash} (\mathbf{w}, \Xi B \models^m C, \xi \models^m, a_j \dots a_n) \\ & \vdash (\mathbf{w}, \Xi B \models^m C \rhd F', \xi \models^m, a_j \dots a_n) \\ & \overset{*}{\vdash} (\mathbf{e}, \Xi B \models^m C \rhd F, \xi \models^m, a_k \dots a_n) \\ & \vdash (\mathbf{e}, \Xi B \models^m G, \xi \models^m \eta, a_k \dots a_n)\end{array}$$

$$\frac{\begin{array}{c}[C, j, \rhd F, k, \models^m \mid -, -, -, -, -]\mathbf{e} \\ [B, i, \models^m C, j, \models^m \mid -, -, -, -, -]\mathbf{w}\end{array}}{[B, i, \models^m G, k, \eta \mid C, j, \models^m, F, k]\mathbf{e}} \quad (\mathbf{e}, C \rhd F, \epsilon) \longmapsto (\mathbf{e}, G, \eta)$$

En ambos casos, las configuraciones resultantes no pueden llevar al autómata a una configuración final puesto que al ser la forma de la pila principal $\Xi B \models^m G$ y no estar vacía la sesión situada en la cima de la pila auxiliar, es imposible vaciar esta última.

A partir de esta lista se puede mostrar por inducción en la longitud de las derivaciones que tanto mediante la manipulación de configuraciones como mediante la manipulación de ítems se obtienen los mismos resultados. □

La complejidad temporal en el peor caso con respecto a la longitud n de la cadena de entrada es $\mathcal{O}(n^6)$ y viene determinada por la regla de combinación de ítems

$$\frac{\begin{array}{l}[C, j, \rhd F, k, \eta' \mid D, p, \models^m, E, q]\mathbf{e} \\ [B, i, \otimes_1 C, j, \models^m \mid -, -, -, -, -]\mathbf{w} \\ [D, p, \otimes_2 E, q, \eta \mid O, u, \models^m, P, v]\mathbf{e}\end{array}}{[B, i, \otimes_1 G, k, \eta \mid O, u, \models^m, P, v]\mathbf{e}} \quad (\mathbf{e}, C \rhd F, \eta') \longmapsto (\mathbf{e}, G, \epsilon)$$

que manipula 7 posiciones de la cadena de entrada, aunque sólo 6 de manera simultánea. La complejidad temporal con respecto a la cadena de entrada es $\mathcal{O}(n^4)$ puesto que cada ítem almacena 4 posiciones de la cadena de entrada.

Capítulo 11

Recapitulación

En los capítulos precedentes se han definido varios modelos de autómata para la clase de los lenguajes de adjunción de árboles. Estos modelos surgen de motivaciones diferentes y utilizan estructuras de almacenamiento diferentes: pilas embebidas, pilas lógicas, pilas de índices o dos pilas. Sin embargo mantienen estrechas relaciones entre sí. En este capítulo analizamos las relaciones de los distintos modelos de autómata y de las técnicas de interpretación tabular asociadas.

11.1 Autómatas generales

Consideraremos en esta sección aquellos autómatas que no limitan la estrategia de análisis sintáctico en lo que respecta al tratamiento de las pilas de índices. Nos referimos pues a los autómatas lineales de índices fuertemente dirigidos (SD–LIA) y a los autómatas con dos pilas fuertemente dirigidos (SD–2SA).

11.1.1 Autómatas fuertemente dirigidos

Los autómatas lineales de índices fuertemente dirigidos y los autómatas con dos pilas fuertemente dirigidos presentan características comunes, entre las que destacamos:

- Ambos permiten la propagación de pilas de índices tanto en las derivaciones de llamada como en las derivaciones de retorno.

- Ambos deben establecer un mecanismo para garantizar que su potencia expresiva es equivalente a los lenguajes de adjunción de árboles. Este mecanismo se basa en la utilización de dos modos, uno de escritura y otro de borrado, que limitan el conjunto de transiciones aplicables en un momento dado y en la utilización de marcas de acción para garantizar que las computaciones realizadas en la fase de llamada se corresponden con las computaciones realizadas en la fase de retorno.

La diferencia entre ambos tipos de autómatas radica en la estructura de datos manipulada por cada uno de ellos, pues mientras los autómatas lineales de índices manipulan una pila en la que cada de uno de sus elementos se compone de un símbolo de pila junto con una pila de índices, los autómatas con dos pilas manipulan dos pilas, una que contiene los símbolos de pila y otra que contiene los índices. Esta diferencia no es más que aparente, puesto que si observamos detenidamente las transiciones permitidas en ambos tipos de autómatas, vemos que existe una correspondencia uno a uno, tal y como se muestra en la tabla 11.1. Concretamente:

Transición	**SD–LIA**	**SD–2SA**
SWAP1	$C[\circ\circ]\ \ m \overset{a}{\longmapsto} m\ \ F[\circ\circ]$	$(m, C, \epsilon) \overset{a}{\longmapsto} (m, F, \epsilon)$
SWAP2	$C[\,]\ \ \mathbf{w} \overset{a}{\longmapsto} \mathbf{e}\ \ F[\,]$	$(\mathbf{w}, C, \models^m) \overset{a}{\longmapsto} (\mathbf{e}, F, \models^m)$
$\models$**WRITE**	$C[\circ\circ]\ m \longmapsto \mathbf{w}\ C[\circ\circ] \models^m F[\,]$	$(m, C, \epsilon) \longmapsto (\mathbf{w}, C \models^m F, \models^m)$
$\rightarrow$**WRITE**	$C[\circ\circ]\ \mathbf{w} \longmapsto \mathbf{w}\ C[\,] \rightarrow F[\circ\circ]$	$(\mathbf{w}, C, \epsilon) \longmapsto (\mathbf{w}, C \rightarrow F, \epsilon)$
$\nearrow$**WRITE**	$C[\circ\circ]\ \mathbf{w} \longmapsto \mathbf{w}\ C[\,] \nearrow F[\circ\circ\gamma']$	$(\mathbf{w}, C, \epsilon) \longmapsto (\mathbf{w}, C \nearrow F, \gamma')$
$\searrow$**WRITE**	$C[\circ\circ\gamma]\ \mathbf{w} \longmapsto \mathbf{w}\ C[\,] \searrow F[\circ\circ]$	$(\mathbf{w}, C, \gamma) \longmapsto (\mathbf{w}, C \searrow F, \epsilon)$
$\models$**ERASE**	$C[\circ\circ] \models^m F[\,]\ \mathbf{e} \longmapsto m\ G[\circ\circ]$	$(\mathbf{e}, C \models^m F, \models^m) \longmapsto (m, G, \epsilon)$
$\rightarrow$**ERASE**	$C[\,] \rightarrow F[\circ\circ]\ \mathbf{e} \longmapsto \mathbf{e}\ G[\circ\circ]$	$(\mathbf{e}, C \rightarrow F, \epsilon) \longmapsto (\mathbf{e}, G, \epsilon)$
$\nearrow$**ERASE**	$C[\,] \nearrow F[\circ\circ\eta']\ \mathbf{e} \longmapsto \mathbf{e}\ G[\circ\circ]$	$(\mathbf{e}, C \nearrow F, \eta') \longmapsto (\mathbf{e}, G, \epsilon)$
$\searrow$**ERASE**	$C[\,] \searrow F[\circ\circ]\ \mathbf{e} \longmapsto \mathbf{e}\ G[\circ\circ\eta]$	$(\mathbf{e}, C \searrow F, \epsilon) \longmapsto (\mathbf{e}, G, \eta)$

Tabla 11.1: Equivalencia de las transiciones de SD–LIA y de SD–2SA

- Las transiciones de SD–LIA apilan y eliminan de la pila los mismos elementos que las transiciones de SD–2SA en la pila maestra, tal y como se indica en la tabla 11.2 .
- Con respecto a los índices, las operaciones realizadas sobre la pila de índices del elemento en la cima de la pila de los SD–LIA se corresponden con los movimientos realizados sobre la pila auxiliar de los SD–2SA, tal y como se muestra en la tabla 11.3. En dicha tabla observamos que siempre que se crea una pila de índices en una transición SD–LIA, se crea una nueva sesión en las pilas del SD–2SA y siempre que se elimina una pila de índices vacía en SD–LIA, se elimina una sesión de las pilas del SD–2SA.
- Se realizan los mismos cambios de modo, tal como se muestra en la tabla 11.4.

Una vez establecida la equivalencia entre transiciones, podemos establecer la equivalencia entre configuraciones. Una configuración

$$(m, \Upsilon \models^{m'} A_1[\,] \ldots \otimes_{l-1} A_{l-1}[\,]\ \otimes_l A_l[\alpha], w)$$

de un autómata lineal de índices fuertemente dirigido es equivalente a la configuración

$$(m, \Xi \models^{m'} A_1 \ldots \otimes_{l-1} A_{l-1}\ \otimes A_l, \xi \models^{m'} \alpha, w)$$

de un autómata con dos pilas fuertemente dirigido, donde Ξ y ξ se obtienen a partir de Υ aplicando recursivamente la misma equivalencia entre configuraciones, teniendo en cuenta que Ξ se refiere a los símbolos de pila y marcas de acción de Υ y ξ a las pilas de índices.

En lo que respecta a las técnicas de tabulación propuestas, ambas son equivalentes. La técnica propuesta para los autómatas lineales de índices fuertemente dirigidos está basada en transiciones SWAP, tratándose por tanto de una extensión de la técnica propuesta por Nederhof para los autómatas a pila [126], mientras que la propuesta para los autómatas con dos pilas fuertemente dirigidos se basa en transiciones PUSH, tratándose en este último caso de una extensión de la técnica propuesta por Lang para los autómatas a pila [104, 107].

Transición	**SD–LIA**	**SD–2SA**
SWAP1	$C \overset{a}{\longmapsto} F$	$C \overset{a}{\longmapsto} F$
SWAP2	$C \overset{a}{\longmapsto} F$	$C \overset{a}{\longmapsto} F$
$\models$**WRITE**	$C \longmapsto C \models^m F$	$C \longmapsto C \models^m F$
$\rightarrow$**WRITE**	$C \longmapsto C \rightarrow F$	$C \longmapsto C \rightarrow F$
$\nearrow$**WRITE**	$C \longmapsto C \nearrow F$	$C \longmapsto C \nearrow F$
$\searrow$**WRITE**	$C \longmapsto C \searrow F$	$C \longmapsto C \searrow F$
$\models$**ERASE**	$C \models^m F \longmapsto G$	$C \models^m F \longmapsto G$
$\rightarrow$**ERASE**	$C \rightarrow F \longmapsto G$	$C \rightarrow F \longmapsto G$
$\nearrow$**ERASE**	$C \nearrow F \longmapsto G$	$C \nearrow F \longmapsto G$
$\searrow$**ERASE**	$C \searrow F \longmapsto G$	$C \searrow F \longmapsto G$

Tabla 11.2: Esqueleto independiente del contexto de las transiciones de SD–LIA y SD–2SA

Transición	**SD–LIA**	**SD–2SA**
SWAP1	$[\circ\circ] \overset{a}{\longmapsto} [\circ\circ]$	$\epsilon \overset{a}{\longmapsto} \epsilon$
SWAP2	$[\,] \overset{a}{\longmapsto} [\,]$	$\models^m \overset{a}{\longmapsto} \models^m$
$\models$**WRITE**	$[\circ\circ] \longmapsto [\circ\circ]\,[\,]$	$\epsilon \longmapsto \models^m$
$\rightarrow$**WRITE**	$[\circ\circ] \longmapsto [\,]\,[\circ\circ]$	$\epsilon \longmapsto \epsilon$
$\nearrow$**WRITE**	$[\circ\circ] \longmapsto [\,]\,[\circ\circ\gamma']$	$\epsilon \longmapsto \gamma'$
$\searrow$**WRITE**	$[\circ\circ\gamma] \longmapsto [\,]\,[\circ\circ]$	$\gamma \longmapsto \epsilon$
$\models$**ERASE**	$[\circ\circ]\,[\,] \longmapsto [\circ\circ]$	$\models^m \longmapsto \epsilon$
$\rightarrow$**ERASE**	$[\,]\,[\circ\circ] \longmapsto [\circ\circ]$	$\epsilon \longmapsto \epsilon$
$\nearrow$**ERASE**	$[\,]\,[\circ\circ\eta'] \longmapsto [\circ\circ]$	$\eta' \longmapsto \epsilon$
$\searrow$**ERASE**	$[\,]\,[\circ\circ] \longmapsto [\circ\circ\eta]$	$\epsilon \longmapsto \eta$

Tabla 11.3: Tratamiento de los índices en las transiciones de SD–LIA y SD–2SA

Transición	**Origen**	**Destino**
SWAP1	m	m
SWAP2	**w**	**e**
$\models$**WRITE**	m	**w**
$\rightarrow$**WRITE**	**w**	**w**
$\nearrow$**WRITE**	**w**	**w**
$\searrow$**WRITE**	**w**	**w**
$\models$**ERASE**	**e**	m
$\rightarrow$**ERASE**	**e**	**e**
$\nearrow$**ERASE**	**e**	**e**
$\searrow$**ERASE**	**e**	**e**

Tabla 11.4: Cambios de modo en SD–LIA y SD–2SA

Ejemplo 11.1 Consideremos el autómata lineal de índices fuertemente dirigido cuyas transiciones se muestran en la tabla 9.22 (página 317), que acepta el lenguaje $\{a^n b^n c^n d^n \mid n > 0\}$. Dicho autómata es equivalente al autómata con dos pilas fuertemente dirigido de la tabla 10.1 (página 336). En la tabla 9.24 (página 318) se muestran las producciones de la gramática lineal de índices obtenida a partir de las transiciones de dichos autómatas. ¶

RLPDA $*$-ascendentes	**SD–LIA**
$C[\circ\circ] \longmapsto F[\circ\circ]$	$C[\circ\circ] \longmapsto F[\circ\circ]$
$C[\,] \overset{a}{\longmapsto} F[\,]$	$C[\,] \overset{a}{\longmapsto} F[\,]$
$C[\circ\circ] \longmapsto C[\circ\circ]\ F[\,]$	$C[\circ\circ] \longmapsto C[\circ\circ]\ \models F[\,]$
$C[\circ\circ] \longmapsto C[\circ\circ]\ F[\circ\circ]$	$C[\circ\circ] \longmapsto C[\,] \rightarrow F[\circ\circ]$
$C[\circ\circ\gamma] \longmapsto C[\circ\circ\gamma]\ F[\circ\circ]$	$C[\circ\circ\gamma] \longmapsto C[\,] \searrow F[\circ\circ]$
$C[\circ\circ] \longmapsto C[\circ\circ]\ F[\circ\circ\gamma']$	$C[\circ\circ] \longmapsto C[\,] \nearrow F[\circ\circ\gamma']$
$C[\circ\circ]\ F[\,] \longmapsto G[\circ\circ]$	$C[\circ\circ]\ \models F[\,] \longmapsto G[\circ\circ]$
$B[\circ\circ_1]\ C[\circ\circ_2] \overset{a}{\longmapsto} F[\circ\circ_2]$	$B[\,] \rightarrow C[\circ\circ] \longmapsto F[\circ\circ]$
$B[\circ\circ_1]\ C[\circ\circ_2\gamma'] \longmapsto F[\circ\circ_2]$	$B[\,] \nearrow C[\circ\circ\gamma'] \longmapsto F[\circ\circ]$
$B[\circ\circ_1\gamma]\ C[\circ\circ_2] \longmapsto F[\circ\circ_2\gamma]$	$B[\,] \searrow C[\circ\circ] \longmapsto F[\circ\circ\gamma]$

Tabla 11.5: Correspondencia entre las transiciones de los RLPDA $*$-Earley y los SD–LIA

Los autómatas a pila restringidos para estrategias $*$-Earley pueden considerarse como una versión restringida de los autómatas lineales de índices fuertemente dirigidos, válida únicamente para expresar estrategias de tipo Earley con respecto a las pilas de índices. En este tipo de estrategias, las pilas de índices son predichas en la fase de llamada y propagadas en la fase ascendente. La equivalencia de las transiciones de ambos modelos de autómata se muestra en la tabla 11.5, en la que se ha prescindido de los modos. La técnica de tabulación propuesta es una especialización de la técnica de tabulación de los autómatas fuertemente dirigidos, en la que se han suprimido los modos de escritura y borrado y las marcas de acción.

11.2 Autómatas descendentes

Consideraremos en esta sección aquellos autómatas en los cuales se permite la predicción de los índices pero no se permite su propagación en las derivaciones de retorno. Nos referiremos entonces a los autómatas a pila embebidos (EPDA), los autómatas lineales de índices orientados a la izquierda (L–LIA) y los autómatas lógicos a pila restringidos a estrategias $*$-descendentes (RLPDA $*$-descendentes).

11.2.1 Autómatas a pila embebidos y autómatas lineales de índices orientados a la izquierda

Los autómatas a pila embebidos son equivalentes a los autómatas lineales de índices orientados a la izquierda. Para mostrar esta equivalencia, realizaremos un cambio de notación. Puesto que todas las pilas contenidas en la pila principal de un EPDA contienen al menos un elemento, pasaremos a representar una pila $[\alpha C$ mediante la notación $C[\alpha]$. Se trata simplemente de un cambio de notación y, por lo tanto, aunque coincide con la semántica que se ha dado a las pilas durante la definición de los esquemas de compilación para gramáticas lineales de índices, no quiere decir que estemos suponiendo que todo EPDA corresponde a la compilación de una de dichas gramáticas. Por citar otro caso tratado en el capítulo 6, en un esquema de compilación para gramáticas de adjunción de árboles, el componente C representará un punto en el recorrido de un árbol elemental mientras que α representará la pila de adjunciones pendientes en dicho punto. De hecho, la semántica asignada a las pilas depende de la motivación del EPDA, en particular de si ha sido construido a partir de un determinado formalismo gramatical o no.

Como consecuencia del cambio de notación, la configuración de un autómata a pila embebido se denotará por el par (Υ, w), donde $\Upsilon \in (V_S[V_S^*])^*$ y $w \in V_T^*$. Deberemos adaptar también la notación de las transiciones. Consideremos caso por caso:

SWAP: Las transiciones de este tipo, que antes se denotaban $C \overset{a}{\longmapsto} F$, pasarán a denotarse $C[\circ\circ] \overset{a}{\longmapsto} F[\circ\circ]$, donde $\circ\circ$ se utiliza para denotar que $\forall \alpha \in V_S^*, C[\alpha] \overset{a}{\longmapsto} F[\alpha]$. El resultado de aplicar una transición de este tipo a una pila $\Upsilon C[\alpha]$ será una pila $\Upsilon F[\alpha]$.

PUSH: Las transiciones de este tipo, que antes se denotaban $C \overset{a}{\longmapsto} CF$, pasarán a denotarse $C[\circ\circ] \overset{a}{\longmapsto} F[\circ\circ C]$. El resultado de aplicar una transición de este tipo a una pila $\Upsilon C[\alpha]$ será una pila $\Upsilon F[\alpha C]$.

POP: Las transiciones de este tipo, que antes se denotaban $CF \overset{a}{\longmapsto} G$, pasarán a denotarse $F[\circ\circ C] \overset{a}{\longmapsto} G[\circ\circ]$. El resultado de aplicar una transición de este tipo a una pila $\Upsilon F[\alpha C]$ será una pila $\Upsilon G[\alpha]$.

WRAP-A: Las transiciones de este tipo, que antes se denotaban $C \overset{a}{\longmapsto} C, [F$, pasarán a denotarse $C[\circ\circ] \overset{a}{\longmapsto} C[\circ\circ]F[\,]$. El resultado de aplicar una transición de este tipo a una pila $\Upsilon C[\alpha]$ será una pila $\Upsilon C[\alpha]F[\,]$.

WRAP-B: Las transiciones de este tipo, que antes se denotaban $C \overset{a}{\longmapsto} [C, F$, pasarán a denotarse $C[\circ\circ] \overset{a}{\longmapsto} C[\,]F[\circ\circ]$. El resultado de aplicar una transición de este tipo a una pila $\Upsilon C[\alpha]$ será una pila $\Upsilon C[\,]F[\alpha]$.

Transición	Original	Nueva notación
SWAP	$C \overset{a}{\longmapsto} F$	$C[\circ\circ] \overset{a}{\longmapsto} F[\circ\circ]$
PUSH	$C \overset{a}{\longmapsto} CF$	$C[\circ\circ] \overset{a}{\longmapsto} F[\circ\circ C]$
POP	$CF \overset{a}{\longmapsto} G$	$F[\circ\circ C] \overset{a}{\longmapsto} G[\circ\circ]$
WRAP-A	$C \overset{a}{\longmapsto} C, [\![F$	$C[\circ\circ] \overset{a}{\longmapsto} C[\circ\circ]\ F[\,]$
WRAP-B	$C \overset{a}{\longmapsto} [\![C, F$	$C[\circ\circ] \overset{a}{\longmapsto} C[\,]\ F[\circ\circ]$
UNWRAP	$C, [\![F \overset{a}{\longmapsto} G$	$C[\circ\circ]\ F[\,] \overset{a}{\longmapsto} G[\circ\circ]$

Tabla 11.6: Cambio de notación en las transiciones de los EPDA

Transición	EPDA	L–LIA
SWAP	$C \overset{a}{\longmapsto} F$	$C[\circ\circ] \overset{a}{\longmapsto} F[\circ\circ]$
WRAP-A	$C \longmapsto C, [\![F$	$C[\circ\circ] \longmapsto C[\circ\circ]\ F[\,]$
WRAP-B	$C \longmapsto [\![C, F$	$C[\circ\circ] \longmapsto C[\,]\ F[\circ\circ]$
UNWRAP	$C, [\![F \longmapsto G$	$C[\circ\circ]\ F[\,] \longmapsto G[\circ\circ]$
WRAP-B + PUSH	$C \longmapsto [\![C,\ X\,F$	$C[\circ\circ] \longmapsto C[\,]\ F[\circ\circ X]$
WRAP-B + POP	$X\,C \longmapsto [\![C,\ F$	$C[\circ\circ X] \longmapsto C[\,]\ F[\circ\circ]$

Tabla 11.7: Equivalencia entre L–LIA y un subconjunto de los EPDA

UNWRAP: Las transiciones de este tipo, que antes se denotaban $C, [\![F \overset{a}{\longmapsto} G$, pasarán a denotarse $C[\circ\circ]F[\,] \overset{a}{\longmapsto} G[\circ\circ]$. El resultado de aplicar una transición de este tipo a una pila $\Upsilon C[\alpha]F[\,]$ será una pila $\Upsilon G[\alpha]$.

En la tabla 11.6 se muestra la relación entre las transiciones originales de los autómatas a pila embebidos y las nuevas transiciones. En la tabla 11.7 se muestra la equivalencia entre el juego de transiciones de los autómatas lineales de índices orientados a la izquierda considerados en el capítulo 9 y el juego de transiciones utilizado para la definición de la técnica de tabulación de los autómata a pila embebidos en el capítulo 6.

11.2.2 Autómatas a pila restringidos $*$-descendentes y autómatas lineales de índices orientados a la izquierda

Los autómatas a pila restringidos para estrategias $*$-descendentes y los autómatas lineales de índices orientados a la izquierda presentan grandes semejanzas. De hecho, podemos establecer una correspondencia casi directa entre las transiciones de los dos modelos de autómata, tal y como se muestra en la tabla 11.8. Observamos que la diferencia estriba en que los RLPDA no pueden modificar el elemento que quedará en segunda posición en la pila después de la aplicación de una transición PUSH, lo que conlleva la necesidad de distinguir dos tipos de transiciones POP y de establecer la restricción de que las transiciones $C[\circ\circ]\ F[\,] \overset{a}{\longmapsto} G[\circ\circ]$ sólo pueden emparejarse

RLPDA $*$-descendentes	**l-LIA**
$C[\circ\circ] \overset{a}{\longmapsto} F[\circ\circ]$	$C[\circ\circ] \overset{a}{\longmapsto} F[\circ\circ]$
$C[\circ\circ] \longmapsto C[\circ\circ]\ F[\,]$	$C[\circ\circ] \longmapsto C[\circ\circ]\ F[\,]$
$C[\circ\circ] \longmapsto C[\circ\circ]\ F[\circ\circ]$	$C[\circ\circ] \longmapsto C[\,]\ F[\circ\circ]$
$C[\circ\circ] \longmapsto C[\circ\circ]\ F[\circ\circ\gamma']$	$C[\circ\circ] \longmapsto C[\,]\ F[\circ\circ\gamma']$
$C[\circ\circ\gamma] \longmapsto C[\circ\circ\gamma]\ F[\circ\circ]$	$C[\circ\circ\gamma] \longmapsto C[\,]\ F[\circ\circ]$
$C[\circ\circ]\ F[\,] \longmapsto G[\circ\circ]$	$C[\circ\circ]\ F[\,] \longmapsto G[\circ\circ]$
$C[\circ\circ]\ F[\,] \longmapsto G[\,]$	

Tabla 11.8: Correspondencia entre las transiciones de los RLPDA $*$-descendentes y de los L–LIA

con transiciones $C[\circ\circ\gamma] \overset{a}{\longmapsto} C[\circ\circ\gamma]\ F'[\,]$. En cambio, los L–LIA no presentan dicha limitación y en consecuencia sólo es necesario un tipo de transición POP, sin restricciones de aplicación. Teniendo en cuenta esta correspondencia entre transiciones, las técnicas de tabulación definidas para los autómatas a pila restringidos para estrategias $*$-descendentes y para los autómatas lineales de índices orientados a la izquierda son equivalentes, con la salvedad de que la primera está basada en transiciones PUSH, tratándose por tanto de una extensión de la técnica propuesta por Lang para los autómatas a pila [104, 107], mientras que la segunda se basa en las transiciones SWAP, tratándose de una extensión de la técnica propuesta por Nederhof para los autómatas a pila [126].

11.3 Autómatas ascendentes

Consideramos en esta sección aquellos autómatas que no permiten realizar predicciones sobre los índices. Nos estamos refiriendo por tanto a los autómatas a pila embebidos ascendentes (BEPDA), los autómatas lógicos a pila restringidos con estrategias $*$-ascendentes (RLPDA $*$-ascendentes), los autómatas lineales de índices orientados a la derecha (R–LIA) y los autómatas con dos pilas ascendentes (BU–2SA). Con respecto a RLPDA $*$-ascendentes y R–LIA, en el capítulo 9 ya se mostró que ambos modelos de autómata son equivalentes, por lo que no incidiremos más en ese aspecto. Mostramos a continuación las relaciones existentes entre los otros modelos de autómatas ascendentes.

11.3.1 Autómatas a pila embebidos ascendentes y autómatas lineales de índices orientados a la derecha

Utilizaremos el mismo cambio de notación realizado en el caso de los autómatas a pila embebidos. Ello conlleva una redefinición de las transiciones. Las transiciones de tipo SWAP, PUSH y POP se redefinen igual que en el caso de los EPDA y las restantes se redefinen como sigue:

UNWRAP-A: Las transiciones de este tipo, que antes se denotaban $C, [F \overset{a}{\longmapsto} G$, pasarán a denotarse $C[\circ\circ]\ F[\,] \overset{a}{\longmapsto} G[\circ\circ]$. El resultado de aplicar una transición de este tipo a una pila $\Upsilon C[\alpha]F[\,]$ será una pila $\Upsilon G[\alpha]$.

Transición	**BEPDA**	**R–LIA**
SWAP	$C \overset{a}{\longmapsto} F$	$C[\circ\circ] \overset{a}{\longmapsto} F[\circ\circ]$
PUSH	$C \overset{a}{\longmapsto} CF$	$C[\circ\circ] \overset{a}{\longmapsto} F[\circ\circ C]$
POP	$CF \overset{a}{\longmapsto} G$	$F[\circ\circ C] \overset{a}{\longmapsto} G[\circ\circ]$
UNWRAP-A	$C, [F \overset{a}{\longmapsto} G$	$C[\circ\circ]\ F[\,] \overset{a}{\longmapsto} G[\circ\circ]$
UNWRAP-B	$[C, F \overset{a}{\longmapsto} G$	$C[\,]\ F[\circ\circ] \overset{a}{\longmapsto} G[\circ\circ]$
WRAP	$C \overset{a}{\longmapsto} C, [F$	$C[\circ\circ] \overset{a}{\longmapsto} C[\circ\circ]F[\,]$

Tabla 11.9: Equivalencia entre las Transiciones de BEPDA y de R–LIA

UNWRAP-B: Las transiciones de este tipo, que antes se denotaban $[C, F \overset{a}{\longmapsto} G$, pasarán a denotarse $C[\,]\ F[\circ\circ] \overset{a}{\longmapsto} G[\circ\circ]$. El resultado de aplicar una transición de este tipo a una pila $\Upsilon C[\,]F[\alpha]$ será una pila $\Upsilon G[\alpha]$.

WRAP: Las transiciones de este tipo, que antes se denotaban $C \overset{a}{\longmapsto} C, [F$, pasarán a denotarse $C[\circ\circ] \overset{a}{\longmapsto} C[\circ\circ]F[\,]$. El resultado de aplicar una transición de este tipo a una pila $\Upsilon C[\alpha]$ será una pila $\Upsilon C[\alpha]F[\,]$.

En la tabla 11.9 se comparan las transiciones de los autómatas a pila embebidos ascendentes con las transiciones utilizadas por los autómatas lineales de índices orientados a la derecha. Aparentemente, el juego de transiciones de los BEPDA no completa el juego de transiciones de los R–LIA. Sin embargo, podemos observar que mediante las transiciones de los autómatas lineales de índices orientados a la derecha que tienen equivalente en los autómatas a pila embebidos ascendentes podemos obtener el resto de transiciones permitidas en los primeros, tal y como se muestra a continuación:

- Una transición $C[\circ\circ X]\ F[\,] \overset{a}{\longmapsto} G[\circ\circ]$ puede emularse mediante la aplicación consecutiva de las transiciones $C[\circ\circ]\ F[\,] \overset{a}{\longmapsto} G'[\circ\circ]$ y $G'[\circ\circ X] \longmapsto G[\circ\circ]$.
- Una transición $C[\circ\circ]\ F[\,] \overset{a}{\longmapsto} G[\circ\circ X]$ puede emularse mediante la aplicación consecutiva de las transiciones $C[\circ\circ]\ F[\,] \overset{a}{\longmapsto} G'[\circ\circ]$ y $G'[\circ\circ] \longmapsto G[\circ\circ X]$.
- Una transición $C[\,]\ F[\circ\circ X] \overset{a}{\longmapsto} G[\circ\circ]$ puede emularse mediante la aplicación consecutiva de las transiciones $C[\,]\ F[\circ\circ] \longmapsto G'[\circ\circ]$ y $G'[\circ\circ X] \overset{a}{\longmapsto} G[\circ\circ]$.
- Una transición $C[\,]\ F[\circ\circ] \overset{a}{\longmapsto} G[\circ\circ X]$ puede emularse mediante la aplicación consecutiva de las transiciones $C[\,]\ F[\circ\circ] \longmapsto G'[\circ\circ]$ y $G'[\circ\circ] \longmapsto G[\circ\circ X]$.

donde G' es un nuevo símbolo de pila. En consecuencia, los autómatas a pila embebidos ascendentes son completamente equivalentes a los autómatas lineales de índices orientados a la derecha y a los autómatas lógicos a pila restringidos con estrategias $*$-ascendentes.

11.3.2 Autómatas con dos pilas ascendentes y autómatas lineales de índices orientados a la derecha

Los autómatas con dos pilas ascendentes presentan grandes similitudes con los autómatas lineales de índices orientados a la derecha. En particular, ambos modelos impiden predecir información respecto a los índices en la fase de llamada. El mecanismo con el cual se maneja esta restricción es lo que los diferencia, pues mientras los primeros optan por la utilización de modos de escritura y de borrado y la restricción de las transiciones que es posible utilizar en cada modo, los autómatas lineales de índices orientados a la derecha optan por establecer una limitación mucho más simple, consistente en que todo símbolo de pila que es apilado debe tener asociada una pila de índices vacía. Teniendo en cuenta estas consideraciones, podemos establecer la correspondencia entre transiciones de ambos modelos autómata que se muestra en la tabla 11.10.

Con respecto a dicha tabla, observamos que las transiciones $C[\,] \overset{a}{\longmapsto} F[\,]$ no forma parte del juego de transiciones básicas de los autómatas lineales de índices orientados a la derecha, aunque pueden ser definidas mediante la aplicación consecutiva de dos transiciones $C[\circ\circ] \longmapsto C[\circ\circ]\ F'[\,]$ y $C[\,]\ F'[\circ\circ] \overset{a}{\longmapsto} F[\circ\circ]$, donde F' es un nuevo símbolo de pila. Por otra parte, las transiciones de tipo $\triangleright$**WRITE** se traducen por transiciones $C[\,] \overset{a}{\longmapsto} C[\,]\ F[\,]$, que si bien no son transiciones elementales pueden ser emuladas mediante la aplicación consecutiva de las transiciones $C[\,] \overset{a}{\longmapsto} C'[\,]$ y $C'[\circ\circ] \longmapsto C'[\circ\circ]\ F[\,]$, la adición de una transición $C'[\circ\circ]\ G[\,] \longmapsto K[\circ\circ]$ por cada transición $C[\circ\circ]\ G[\,] \longmapsto K[\circ\circ]$ presente en el autómata y la adición de una transición $C'[\,]\ G[\circ\circ] \longmapsto K[\circ\circ]$ por cada transición $C[\,]\ G[\circ\circ] \longmapsto K[\circ\circ]$ presente en el autómata, donde C' es un nuevo símbolo de pila.

En lo referente a las marcas de acción, las marcas $\triangleright$ son redundantes puesto que se almacenan en la pila siempre que se realiza una operación de apilamiento en la pila maestra, independientemente de la operación realizada sobre la pila auxiliar. Las marcas $\models^m$ de sesión son necesarias puesto que es preciso señalar los límites entre las distintas pilas de índices almacenadas en la pila auxiliar.

Considerando únicamente las operaciones realizadas sobre las pila maestra en el caso de los BU–2SA y las operaciones realizadas sobre los símbolos de pila en el caso de los R–LIA, observamos que ambos modelos de autómata presentan un comportamiento equivalente en este aspecto, tal y como sugiere la tabla 11.11.

En lo que respecta al tratamiento de los índices, las operaciones realizadas sobre la pila auxiliar de los BU–2SA se corresponden con las operaciones realizadas sobre la pila de índices asociada al símbolo de pila situado en la cima, tal y como se muestra en la tabla 11.11. En dicha tabla observamos que siempre que se crea una nueva sesión en BU–2SA se crea una pila de índices vacía en R–LIA y siempre que se elimina una sesión de las pilas del BU–2SA se elimina una pila de índices vacía en R–LIA.

Transición	**BU–2SA**	**R–LIA**
SWAP1	$(m, C, \epsilon) \overset{a}{\longmapsto} (m, F, \epsilon)$	$C[\circ\circ] \overset{a}{\longmapsto} F[\circ\circ]$
SWAP2	$(\mathbf{w}, C, \models^m) \overset{a}{\longmapsto} (\mathbf{e}, F, \models^m)$	$C[\,] \overset{a}{\longmapsto} F[\,]$
$\models$**WRITE**	$(m, C, \epsilon) \longmapsto (\mathbf{w}, C\models^m F, \models^m)$	$C[\circ\circ] \longmapsto C[\circ\circ]\ F[\,]$
$\triangleright$**WRITE**	$(\mathbf{w}, C, \epsilon) \longmapsto (\mathbf{w}, C \triangleright F, \epsilon)$	$C[\,] \longmapsto C[\,]\ F[\,]$
$\models$**ERASE**	$(\mathbf{e}, C\models^m F, \models^m) \longmapsto (m, G, \epsilon)$	$C[\circ\circ]\ F[\,] \longmapsto G[\circ\circ]$
$\rightarrow$**ERASE**	$(\mathbf{e}, C \triangleright F, \epsilon) \longmapsto (\mathbf{e}, G, \epsilon)$	$C[\,]\ F[\circ\circ] \longmapsto G[\circ\circ]$
$\nearrow$**ERASE**	$(\mathbf{e}, C \triangleright F, \eta') \longmapsto (\mathbf{e}, G, \epsilon)$	$C[\,]\ F[\circ\circ\eta'] \longmapsto G[\circ\circ]$
$\searrow$**ERASE**	$(\mathbf{e}, C \triangleright F, \epsilon) \longmapsto (\mathbf{e}, G, \eta)$	$C[\,]\ F[\circ\circ] \longmapsto G[\circ\circ\eta]$

Tabla 11.10: Correspondencia entre las transiciones de BU–2SA y R–LIA

Transición	**Esqueleto**		**Índices**	
	BU–2SA	**R–LIA**	**BU–2SA**	**R–LIA**
SWAP1	$C \overset{a}{\longmapsto} F$	$C \overset{a}{\longmapsto} F$	$\epsilon \overset{a}{\longmapsto} \epsilon$	$[\circ\circ] \overset{a}{\longmapsto} [\circ\circ]$
SWAP2	$C \overset{a}{\longmapsto} F$	$C \overset{a}{\longmapsto} F$	$\models^m \overset{a}{\longmapsto} \models^m$	$[\,] \overset{a}{\longmapsto} [\,]$
$\models$**WRITE**	$C \longmapsto C\ F$	$C \longmapsto C\ F$	$\epsilon \longmapsto \models^m$	$[\circ\circ] \longmapsto [\circ\circ]\ [\,]$
$\triangleright$**WRITE**	$C \longmapsto C\ F$	$C \longmapsto C\ F$	$\epsilon \longmapsto \epsilon$	$[\,] \longmapsto [\,]\ [\,]$
$\models$**ERASE**	$C\ F \longmapsto G$	$C\ F \longmapsto G$	$\models^m \longmapsto \epsilon$	$[\circ\circ]\ [\,] \longmapsto [\circ\circ]$
$\rightarrow$**ERASE**	$C\ F \longmapsto G$	$C\ F \longmapsto G$	$\epsilon \longmapsto \epsilon$	$[\,]\ [\circ\circ] \longmapsto [\circ\circ]$
$\nearrow$**ERASE**	$C\ F \longmapsto G$	$C\ F \longmapsto G$	$\eta' \longmapsto \epsilon$	$[\,]\ [\circ\circ\eta'] \longmapsto [\circ\circ]$
$\searrow$**ERASE**	$C\ F \longmapsto G$	$C\ F \longmapsto G$	$\epsilon \longmapsto \eta$	$[\,]\ [\circ\circ] \longmapsto [\circ\circ\eta]$

Tabla 11.11: Esqueleto independiente del contexto y tratamiento de los índices en las transiciones de BU–2SA y R–LIA

Queda por tratar el tema de los modos en BU–2SA. En las derivaciones de llamada el elemento relevante lo constitiuye el hecho de que la sesión actual en la pila auxiliar está vacía. Por contra, el modo puede ser de escritura o de borrado. Lo mismo ocurre en el caso de las derivaciones de puntos especiales. En lo que respecta a las derivaciones de retorno, la información relevante se refiere al contenido de la sesión actual en la pila auxiliar, que no debe estar vacía. Por tanto, los modos de escritura y de borrado no juegan un papel relevante en este modelo de autómatas y pueden ser suprimidos.

Podemos establecer entonces una equivalencia entre configuraciones de tal modo que una configuración

$$(m, \Xi \models^{m'} A_1 \ldots \triangleright A_{l-1}\ \triangleright A_l, \xi \models^{m'} \alpha, w)$$

de un autómata con dos pilas ascendente es equivalente a la configuración

$$(\Upsilon \models^{m'} A_1[\,] \ldots A_{l-1}[\,]\ A_l[\alpha], w)$$

de un autómata lineal de índices orientado a la derecha, donde Ξ y ξ se obtienen a partir de Υ aplicando recursivamente la misma equivalencia entre configuraciones, teniendo en cuenta que Ξ se refiere a los símbolos de pila y marcas de acción de Υ y ξ a las pilas de índices.

Sin embargo, los BU–2SA no son equivalentes a los R–LIA, puesto que las transiciones definidas para los primeros no permiten emular las transiciones $C[\circ\circ\eta]\ \ F[\ ]\ \overset{a}{\longmapsto}\ G[\circ\circ]$ y $C[\circ\circ]\ \ F[\ ]\ \overset{a}{\longmapsto}\ G[\circ\circ\eta']$. En consecuencia, los autómatas con dos pilas ascendentes son un subconjunto propio de los autómatas lineales de índices orientados a la derecha y la técnica de tabulación desarrollada para los segundos puede ser aplicada a los primeros con las modificaciones pertinentes debidas a las diferentes estructuras de almacenamiento utilizadas en ambos casos.

Capítulo 12

Conclusiones

El trabajo descrito en esta tesis representa una aportación significativa y original al análisis sintáctico de los lenguajes de adjunción de árboles y por extensión al procesamiento del lenguaje natural, a la inteligencia artificial, y a la teoría de autómatas y lenguajes formales.

El panorama al comenzar el trabajo que daría lugar a esta memoria consistía en un grupo disperso de algoritmos para el análisis sintáctico de gramáticas de adjunción de árboles y apenas un par de algoritmos para el análisis sintáctico de las gramáticas lineales de índices. En esta memoria se ha mostrado que es posible establecer un camino evolutivo continuo en el que se sitúan los algoritmos de análisis sintáctico que incorporan las estrategias de análisis más importantes, tanto para el caso de las gramáticas de adjunción de árboles como para el caso de las gramáticas lineales de índices. Los diferentes algoritmos se han definido con esquemas de análisis sintáctico, de tal modo que los algoritmos más complejos se derivan a partir de los menos complejos aplicando una secuencia de transformaciones simples.

En el caso de las gramáticas lineales de índices el resultado es doblemente interesante, pues si bien se ha esgrimido a su favor su adecuación como formalismo intermedio para el análisis de gramáticas de adjunción de árboles, lo cierto es que numerosas estrategias de análisis para estas últimas no se hallaban incorporadas a ningún algoritmo de análisis sintáctico para gramáticas lineales de índices. En consecuencia, era necesario sacrificar la estrategia de análisis si se optaba por este enfoque, lo que limitaba enormemente su aplicación práctica. Con el trabajo desarrollado en esta memoria hemos salvado ese obstáculo definiendo algoritmos de análisis sintáctico para gramáticas lineales de índices que incorporan la versión equivalente de las estrategias de análisis más populares para gramáticas de adjunción de árboles.

En el caso de las gramáticas independientes del contexto es posible optar por un diseño modular en el cual se separa la definición y la ejecución de una determinada estrategia de análisis. En particular, es posible definir un algoritmo de análisis sintáctico como un conjunto de transiciones de un autómata a pila, probablemente no determinista, el cual puede ser interpretado eficientemente mediante las técnicas de tabulación disponibles. Este enfoque presenta ventajas evidentes, entre la cuales cabe citar la simplificación de las pruebas de corrección de los algoritmos, los cuales son más fáciles de comprender y, al ser ejecutados en un entorno homogéneo, son fácilmente comparables. En esta memoria hemos adaptado este enfoque a los lenguajes de adjunción de árboles, proporcionando modelos de autómata con los que describir los algoritmos de análisis y técnicas de tabulación con las que pueden ser ejecutados eficientemente.

El primer modelo de autómata considerado ha sido el de los autómatas a pila embebidos, una extensión de los autómatas a pila que utiliza como estructura de almacenamiento una pila de pilas. Hemos definido una versión de este tipo de autómatas en la cual se simplifica la forma de las transiciones y se elimina el control de estado finito, manteniendo la potencia expresiva. La nueva formulación permite diseñar una técnica de tabulación que, si bien no es general, permite ejecutar eficientemente los autómatas a pila embebidos obtenidos a partir de los esquemas de compilación que incorporan las estrategias de análisis sintáctico presentes en la literatura.

En el caso de las gramáticas de adjunción de árboles, los autómatas a pila embebidos permiten reconocer las adjunciones de modo descendente. Si deseamos reconocer las adjunciones de manera ascendente, debemos recurrir a los autómatas a pila embebidos ascendentes, los cuales han sido completamente reformulados con el fin de lograr una definición formal consistente, hecho que no se había conseguido hasta el momento. Se han definido versiones con estados y sin estados, esta última más simple y apta para la aplicación de técnicas de tabulación.

Hemos considerado también autómatas con una estructura de almacenamiento diferente, una pila en la que cada uno de los elementos que la componen tiene asociado una pila de índices. En este caso hemos seguido dos aproximaciones diferentes. La primera aproximación parte de la restricción de los autómatas lógicos a pila para adaptarlos al reconocimiento de las gramáticas lineales de índices, obteniéndose diferentes tipos de autómatas especializados en diversas estrategias de análisis según el tipo de transiciones permitidos. Este enfoque presenta el inconveniente de que es preciso añadir ciertas restricciones a la combinación de transiciones para garantizar que la clase de los lenguajes aceptados es equivalente a la clase de los lenguajes de adjunción de árboles.

La segunda aproximación parte de los autómatas a pila y de la extensión de las técnicas de tabulación propuestas para estos últimos, dando lugar al concepto de autómata lineal de índices, con tres variantes diferentes: Los autómatas lineales de índices orientados a la derecha, adecuados para estrategias en las cuales las adjunciones se reconocen de manera ascendente, los autómatas lineales de índices orientados a la izquierda, aptos para estrategias de análisis en las que las adjunciones se tratan de forma descendente, y los autómatas lineales de índices fuertemente dirigidos, capaces de incorporar estrategias de análisis en las cuales las adjunciones se tratan de manera ascendente, descendente o de ambas maneras. Esta última variante es completamente original, como también lo son su técnica de interpretación tabular y la correspondiente a la variante orientada a la izquierda.

El último modelo de autómata definido hace uso de una estructura de almacenamiento diferente. En este caso, se conserva la pila de los autómatas a pila, que pasa a denominarse pila maestra pues es la encargada de dirigir el proceso de análisis, y se añade una nueva pila, que recibe el nombre de pila auxiliar pues es utilizada para almacenar elementos auxiliares o índices que restringen las transiciones aplicables en un momento dado. Hemos descrito dos versiones diferentes de este tipo de autómatas, los autómatas con dos pilas fuertemente dirigidos, aptos para describir estrategias de análisis arbitrarias, y los autómatas con dos pilas ascendentes, adecuados para describir estrategias de análisis en las cuales las adjunciones de procesan ascendentemente.

Finalmente, hemos analizado conjuntamente todos estos modelos de autómata, los cuales se pueden dividir en tres grandes grupos: la familia de los autómatas generales, de la que forman parte los autómatas lineales de índices fuertemente dirigidos y los autómatas con dos pilas fuertemente dirigidos; la familia de los autómatas descendentes, en la que se encuadran los autómatas a pila embebidos y los autómatas lineales de índices orientados a la izquierda; y la familia de los autómatas ascendentes, en la que se enmarcan los autómatas a pila embebidos ascendentes, los autómatas lineales de índices orientados a la derecha y los autómatas con dos pilas ascendentes.

12.1 Trabajo futuro

El trabajo de investigación mostrado en esta memoria no termina en sí mismo sino que abre las puertas a numerosas líneas de trabajo futuro.

Un aspecto que presenta gran interés es el estudio del comportamiento práctico de los diferentes modelos de autómata y el desarrollo de esquemas de compilación y de técnicas de tabulación optimizadas que tengan en cuenta las construcciones específicas que aparecen en las gramáticas de las lenguas naturales. Premisa indispensable para emprender este estudio es la disponibilidad de una gramática de adjunción de árboles de amplia cobertura, desgraciadamente inexistente en el caso de las lenguas peninsulares.

En los últimos años se están dedicando esfuerzos importantes a la definición de formalismos gramaticales que preservan, al menos en parte, las características que hacen de las gramáticas de adjunción de árboles un formalismo adecuado para la descripción sintáctica del lenguaje natural pero que presenten una menor complejidad computacional, aunque ello conlleve un pérdida parcial de la sensibilidad al contexto. En este ámbito, resulta de interés la definición de modelos de autómata especializados en el análisis de dichos formalismos, entre los que cabe destacar las gramáticas de adjunción de árboles que restringen las adjunciones realizables en la espina [167] y las gramáticas de inserción de árboles [179].

Una vez que se dispone de mecanismos operacionales que permiten analizar eficientemente la estructura sintagmática de las frases, resulta inevitable avanzar hacia el siguiente nivel de análisis del lenguaje, tratando de capturar los aspectos semánticos del mismo. La extensión natural de las gramáticas de adjunción de árboles que permite capturar, al menos en parte, la semántica del lenguaje, son las gramáticas de adjunción de árboles con estructuras de rasgos [210]. En esta dirección, ya se han dado los primeros pasos para incorporar el tratamiento de las estructuras de rasgos a los modelos de autómata propuestos en esta memoria.

Ya finalmente, apartándonos del procesamiento del lenguaje natural, una línea de investigación futura consiste en la aplicación al campo de la programación lógica de los principios subyacentes a las técnicas de tabulación presentadas para los modelos de autómata definidos en esta memoria. Las gramáticas lineales de índices pueden verse como un tipo especial de programas lógicos en el que los argumentos de los predicados tienen un único argumento en la forma de una pila. La forma particular en que se manipulan los predicados y sus argumentos, nos ha permitido colapsar las derivaciones en una forma muy compacta. Tenemos la intuición de que un tipo similar de compactación es posible para ciertos predicados de programas lógicos, en particular para aquellos cuyos argumentos se construyen a partir de los valores de otros argumentos tal que dichos valores son poco dependientes del contexto en el que son evaluados.

El sino más hermoso de una teoría física es el de señalar el camino para establecer otra más amplia, en cuyo seno pervive como caso límite.

Albert Einstein [70, página 72]

Parte III

Apéndices

Apéndice A

Esquemas de análisis sintáctico

Los algoritmos de análisis sintáctico de esta memoria se describen utilizando *esquemas de análisis sintáctico*, una estructura para realizar descripciones de alto nivel de algoritmos de análisis desarrollada por Sikkel en [189]. En [191] se pueden encontrar una descripción más breve. En [190] se tratan los problemas inherentes a la verificación de la corrección de esquemas de análisis.

A.1 Esquemas de análisis sintáctico para CFG

Los esquemas de análisis sintáctico permiten describir cómodamente algoritmos de análisis sintáctico y estudiar fácilmente las relaciones entre diferentes algoritmos mediante el análisis de las relaciones formales entre los esquemas de análisis subyacentes.

Definición A.1 *Un* sistema de análisis sintáctico *para una gramática independiente del contexto $\mathcal{G}$ y una cadena de entrada $a_1 \ldots a_n$ es un triple $\mathbb{P} = \langle \mathcal{I}, \mathcal{H}, \mathcal{D} \rangle$, en el que*

- *$\mathcal{I}$ es un conjunto de* ítems *que representan resultados intermedios del proceso de análisis;*
- *$\mathcal{H}$ es un conjunto inicial de ítems denominados* hipótesis *que representan la cadena que va a ser analizada*[1]*;*
- *$\mathcal{D} \subseteq \wp_{fin}(\mathcal{H} \cup \mathcal{I}) \times \mathcal{I}$ es un conjunto de reglas de inferencia, denominadas* pasos deductivos, *mediante las cuales se derivan nuevos ítems a partir de los ítems existentes. Estos pasos deductivos tienen la forma $\eta_1, \ldots, \eta_k \vdash \xi$ y su significado es el siguiente: si todos los antecedentes $\eta_i \in \mathcal{H} \cup \mathcal{I}$ de un paso deductivo ya existen, entonces el consecuente ξ deberá ser generado por el analizador sintáctico. Los pasos deductivos se suponen cuantificados universalmente para todas las posibles variables que aparezcan en los ítems a menos que se indique explícitamente lo contrario.*

Un conjunto $\mathcal{F} \subseteq \mathcal{I}$ de ítems finales *indica mediante su presencia el reconocimiento de la cadena de entrada.*

Definición A.2 *Un* sistema de análisis sintáctico no instanciado *para una gramática independiente del contexto $\mathcal{G}$ es un triple $\langle \mathcal{I}, \mathcal{H}, \mathcal{D} \rangle$ con $\mathcal{H}$ una función que asigna un conjunto de hipótesis a cada cadena de entrada $a_1 \ldots a_n \in V_T^*$, tal que $\langle \mathcal{I}, \mathcal{H}(a_1 \ldots a_n), \mathcal{D} \rangle$ es un sistema de análisis sintáctico.*

[1] No es necesario que $\mathcal{H} \subset \mathcal{I}$. Por el contrario, es habitual que ambos conjuntos sean disjuntos.

Definición A.3 *Un* esquema de análisis *para alguna subclase de gramáticas independientes del contexto es una función que asigna un sistema de análisis sintáctico no instanciado a cada gramática en dicha subclase.*

A.2 Sistemas de deducción gramatical

Es interesante reseñar la similitud de los esquemas de análisis sintáctico con los *sistemas de deducción gramatical* propuestos por Shieber, Schabes y Pereira en [188].

Definición A.4 *Un* sistema de deducción gramatical *para una gramática $\mathcal{G}$ y una cadena de entrada $a_1 \ldots a_n$ es una cuádrupla $\langle I, A, G, R\rangle$ en la que*

- *I es una clase de* esquemas de fórmula lógica *comúnmente llamados ítems que representan resultados intermedios del proceso de análisis.*
- *A es un conjunto de* axiomas, *esto es, ítems que representan resultados intermedios que son ciertos* per se *y que por tanto no precisan ser deducidos a partir de otros ítems.*
- *G es un conjunto de* fórmulas meta, *esto es, un conjunto de ítems que establecen si la cadena de entrada pertenece al lenguaje definido por la gramática.*
- *R es un conjunto de* reglas de inferencia *que permiten derivar nuevos ítems a partir de otros ítems. La forma general de tales reglas es*

$$\frac{A_1 \ldots A_n}{B} \ \langle \textit{condiciones de aplicación} \rangle$$

  donde los antecedentes $A_1 \ldots A_n$ antecedentes *y el* consecuente B *son esquemas de fórmula, esto es, pueden contener metavariables sintácticas que serán instanciadas por los términos apropiados cuando la regla sea utilizada.*

La relación entre los esquemas de análisis sintáctico y los sistemas de deducción gramatical queda resumida en la tabla A.1.

Esquemas de análisis sintáctico	**Sistemas de deducción gramatical**
ítems	esquemas de fórmula lógica
pasos deductivos	reglas de inferencia
hipótesis+ítems iniciales	axiomas
ítems finales	fórmulas meta

Tabla A.1: Relación entre esquemas de análisis y sistemas de deducción gramatical

El motivo por el cual, aun siendo prácticamente equivalentes, nos hemos decantado por la utilización de los esquemas de análisis es porque consideramos que estos están más desarrollados, por cuanto se dispone de métodos de comprobación de la corrección y la completud [190] y se han descrito las reglas que permiten derivar esquemas de análisis correctos a partir de otros que ya son conocidos como correctos [189].

A.3 Transformación de esquemas de análisis sintáctico

Se pueden establecer varias clases de relaciones entre algoritmos de análisis sintáctico mediante la definición de relaciones entre los esquemas de análisis subyacentes. Concretamente, podemos *generalizar* y *filtrar* esquemas [189]. Antes de presentar estas relaciones, debemos realizar una serie de definiciones.

Definición A.5 *Dado un sistema de análisis* $\mathbb{P} = \langle \mathcal{I}, \mathcal{H}, \mathcal{D} \rangle$ *la relación* $\vdash \subseteq \wp_{fin}(\mathcal{H} \cup \mathcal{I}) \times \mathcal{I}$ *está definida por*

$$Y \vdash \xi \textit{ si } (Y', \xi) \in \mathcal{D} \textit{ para algún } Y' \in Y$$

Debemos precisar que existe una distinción entre el conjunto de pasos deductivos $\mathcal{D}$ y la relación de inferencia $\vdash$: si $\eta_1, \ldots, \eta_k \vdash \xi$ entonces también se mantiene que $\eta_1, \ldots, \eta_k \xi' \vdash \xi$ para cualquier ξ'. Por consiguiente, se define $\vdash$ como el cierre de $\mathcal{D}$ bajo la adición de antecedentes a una inferencia.

Definición A.6 *Dado un sistema de análisis* $\mathbb{P} = \langle \mathcal{I}, \mathcal{H}, \mathcal{D} \rangle$, *una* secuencia deductiva *en* $\mathbb{P}$ *es un par* $(Y; \xi_1, \ldots, \xi_j) \in \wp_{fin}(\mathcal{H} \cup \mathcal{I}) \times \mathcal{I}^+$. *Por cuestiones prácticas, escribiremos* $Y \vdash \xi_1 \vdash \ldots \vdash \xi_j$ *en lugar de* $(Y; \xi_1, \ldots, \xi_j)$.

Definición A.7 *Dado un sistema de análisis* $\mathbb{P} = \langle \mathcal{I}, \mathcal{H}, \mathcal{D} \rangle$, *el* conjunto de secuencias deductivas $\Delta \subseteq \wp_{fin}(\mathcal{H} \cup \mathcal{I}) \times \mathcal{I}^+$ *se define como*

$$\Delta = \{(Y; \xi_1, \ldots, \xi_j) \in \wp_{fin}(\mathcal{H} \cup \mathcal{I}) \times \mathcal{I}^+ \mid Y \vdash \xi_1 \vdash \ldots \vdash \xi_j\}$$

Definición A.8 *Dado un sistema de análisis* $\mathbb{P} = \langle \mathcal{I}, \mathcal{H}, \mathcal{D} \rangle$, *la relación* $\overset{*}{\vdash} \subseteq \wp_{fin}(\mathcal{H} \cup \mathcal{I}) \times \mathcal{I}$ *se define como*

$$Y \overset{*}{\vdash} \xi \textit{ si } \xi \in Y \textit{ ó } Y \vdash \ldots \vdash \xi$$

A.3.1 Generalización

Existen tres tipos de relaciones de generalización:

- *Refinamiento de los ítems*: un esquema de análisis sintáctico **B** es el resultado de aplicar un refinamiento a los ítems del esquema de análisis **A**, denotado por $\mathbf{A} \overset{\mathbf{ir}}{\Longrightarrow} \mathbf{B}$, si un ítem individual de **A** es dividido en varios ítems en **B** y los pasos deductivos son adaptados de acuerdo con los cambios producidos. La relación $\mathbf{A} \overset{\mathbf{ir}}{\Longrightarrow} \mathbf{B}$ se mantiene si $\mathbb{P}_{\mathrm{A}} \xrightarrow{\mathbf{ir}} \mathbb{P}_{\mathrm{B}}$ para toda gramática y toda cadena de entrada. A su vez, la relación de refinamiento $\mathbb{P}_{\mathrm{A}} \xrightarrow{\mathbf{ir}} \mathbb{P}_{\mathrm{B}}$ entre sistemas de análisis sintáctico es válida si existe una función f de $\mathcal{I}_{\mathrm{B}}$ en $\mathcal{I}_{\mathrm{A}}$, extensible a secuencias de pasos deductivos Δ, tal que

$$\mathcal{I}_{\mathrm{A}} = f(\mathcal{I}_{\mathrm{B}})$$

$$\Delta_{\mathrm{A}} = f(\Delta_{\mathrm{B}})$$

- *Refinamiento de los pasos*: un esquema de análisis sintáctico **B** es el resultado de aplicar un refinamiento a los pasos del esquema de análisis **A**, denotado por $\mathbf{A} \overset{\mathbf{sr}}{\Longrightarrow} \mathbf{B}$, si un paso deductivo individual en **A** es dividido en varios pasos deductivos en **B**, teniendo en cuenta que se pueden introducir nuevos ítems siempre que sea necesario almacenar nuevos resultados intermedios. La relación $\mathbf{A} \overset{\mathbf{sr}}{\Longrightarrow} \mathbf{B}$ se cumple si $\mathbb{P}_{\mathrm{A}} \xrightarrow{\mathbf{sr}} \mathbb{P}_{\mathrm{B}}$ para toda gramática y toda cadena de entrada. A su vez, la relación de refinamiento $\mathbb{P}_\mathrm{A} \xrightarrow{\mathbf{sr}} \mathbb{P}_\mathrm{B}$ entre sistemas de análisis sintáctico es válida si

$$\mathcal{I}_\mathrm{A} \subseteq \mathcal{I}_\mathrm{B}$$

$$\overset{*}{\vdash}_\mathrm{A} \subseteq \overset{*}{\vdash}_\mathrm{B}$$

donde la relación $\overset{*}{\vdash}$ en $\wp_{\mathit{fin}}(\mathcal{I} \cup \mathcal{H}) \times \mathcal{I}$ para un sistema de análisis sintáctico $\mathbb{P} = \langle \mathcal{I}, \mathcal{H}, \mathcal{D} \rangle$ se define como $Y \overset{*}{\vdash} \xi$ *si* $\xi \in Y$ *ó* $Y \vdash \ldots \vdash \xi$.

- *Extensión*: un esquema de análisis $\mathbf{B}$ es una extensión del esquema $\mathbf{A}$, denotado por $\mathbf{A} \overset{\mathbf{ext}}{\Longrightarrow} \mathbf{B}$, si está definido para una clase más amplia de gramáticas. Esta relación es válida si $\mathbf{A}(\mathcal{G}) = \mathbf{B}(\mathcal{G})$ para toda gramática $\mathcal{G}$ para la cual $\mathbf{A}$ está definida.

Proposición A.1 *Cada una de las relaciones* $\overset{\mathbf{ir}}{\Longrightarrow}$, $\overset{\mathbf{sr}}{\Longrightarrow}$ *y* $\overset{\mathbf{ext}}{\Longrightarrow}$ *posee las propiedades reflexiva y transitiva.*

Proposición A.2 *Si* $\mathbf{A} \overset{\mathbf{ir}}{\Longrightarrow} \mathbf{B}$ *entonces la corrección y completud de* $\mathbf{B}$ *implica la corrección y completud de* $\mathbf{A}$*, pero no necesariamente a la inversa.*

Proposición A.3 *La relación* $\overset{\mathbf{sr}}{\Longrightarrow}$ *preserva la corrección y completud: si* $\mathbf{A} \overset{\mathbf{sr}}{\Longrightarrow} \mathbf{B}$ *entonces la corrección y completud de* $\mathbf{A}$ *implica la corrección y completud de* $\mathbf{B}$*.*

Proposición A.4 *Si* $\mathbf{A} \overset{\mathbf{sr}}{\Longrightarrow} \mathbf{B} \overset{\mathbf{ir}}{\Longrightarrow} \mathbf{C}$ *entonces existe un esquema* $\mathbf{D}$ *tal que* $\mathbf{A} \overset{\mathbf{ir}}{\Longrightarrow} \mathbf{D} \overset{\mathbf{sr}}{\Longrightarrow} \mathbf{C}$*.*

A.3.2 Filtrado

Mediante la generalización se pueden obtener mejoras cualitativas de un esquema de análisis, puesto que se obtiene un control más fino sobre los ítems y los pasos deductivos. En cambio, si lo que se desea es mejorar cuantitativamente un esquema de análisis, entonces deberemos reducir el número de ítems y pasos deductivos. Esto es posible mediante la realización de uno de los siguientes tipos de relaciones de filtrado:

- *Filtrado estático*: las partes redundantes de un esquema son descartadas sin más. La relación $\mathbf{A} \overset{\mathbf{sf}}{\Longrightarrow} \mathbf{B}$ se mantiene si $\mathbb{P}_\mathrm{A} \xrightarrow{\mathbf{sf}} \mathbb{P}_\mathrm{B}$ para toda gramática y cadena de entrada. A su vez, la relación de refinamiento $\mathbb{P}_\mathrm{A} \xrightarrow{\mathbf{sf}} \mathbb{P}_\mathrm{B}$ entre sistemas de análisis sintáctico es válida si

$$\mathcal{I}_\mathrm{A} \supseteq \mathcal{I}_\mathrm{B}$$

$$\mathcal{D}_\mathrm{A} \supseteq \mathcal{D}_\mathrm{B}$$

- *Filtrado dinámico*: la validez de algunos ítems puede hacerse depender a su vez de la validez de otros ítems . Por consiguiente, en este caso se aplica información acerca del contexto para determinar si se puede aplicar un determinado paso deductivo. La relación $\mathbf{A} \overset{\mathbf{df}}{\Longrightarrow} \mathbf{B}$ se mantiene si $\mathbb{P}_\mathrm{A} \xrightarrow{\mathbf{df}} \mathbb{P}_\mathrm{B}$ para toda gramática y cadena de entrada. A su vez, la relación de refinamiento $\mathbb{P}_\mathrm{A} \xrightarrow{\mathbf{df}} \mathbb{P}_\mathrm{B}$ entre sistemas de análisis sintáctico es válida si

$$\mathcal{I}_\mathrm{A} \supseteq \mathcal{I}_\mathrm{B}$$

$$\vdash_\mathrm{A} \supseteq \vdash_\mathrm{B}$$

- *contracción de los ítems*: Varios ítems son reemplazados por un único ítem del esquema de análisis. La relación $\mathbf{A} \stackrel{\mathbf{ic}}{\Longrightarrow} \mathbf{B}$ se mantiene si $\mathbb{P}_{\mathrm{A}} \stackrel{\mathbf{ic}}{\Longrightarrow} \mathbb{P}_{\mathrm{B}}$ para toda gramática y cadena de entrada, para la cual debe existir una función f de $\mathcal{I}_{\mathrm{A}}$ en $\mathcal{I}_{\mathrm{B}}$, extensible a secuencias de pasos deductivos Δ, tal que

$$\mathcal{I}_{\mathrm{B}} = f(\mathcal{I}_{\mathrm{A}})$$
$$\Delta_{\mathrm{B}} = f(\Delta_{\mathrm{A}})$$

- *Contracción de pasos*: secuencias de pasos deductivos son reemplazadas por un único paso deductivo. La relación $\mathbf{A} \stackrel{\mathbf{sc}}{\Longrightarrow} \mathbf{B}$ se mantiene si $\mathbb{P}_{\mathrm{A}} \stackrel{\mathbf{sc}}{\longrightarrow} \mathbb{P}_{\mathrm{B}}$ para toda gramática y cadena de entrada. A su vez, la relación de refinamiento $\mathbb{P}_{\mathrm{A}} \stackrel{\mathbf{sc}}{\longrightarrow} \mathbb{P}_{\mathrm{B}}$ entre sistemas de análisis sintáctico es válida si

$$\mathcal{I}_{\mathrm{A}} \supseteq \mathcal{I}_{\mathrm{B}}$$
$$\stackrel{*}{\vdash}_{\mathrm{A}} \supseteq \stackrel{*}{\vdash}_{\mathrm{B}}$$

Proposición A.5 $\stackrel{\mathbf{sf}}{\Longrightarrow} \subseteq \stackrel{\mathbf{df}}{\Longrightarrow} \subseteq \stackrel{\mathbf{sc}}{\Longrightarrow}$.

Proposición A.6 *Cada una de las relaciones* $\stackrel{\mathbf{sf}}{\Longrightarrow}$, $\stackrel{\mathbf{df}}{\Longrightarrow}$, $\stackrel{\mathbf{ic}}{\Longrightarrow}$ *y* $\stackrel{\mathbf{sc}}{\Longrightarrow}$ *posee las propiedades reflexiva y transitiva.*

Proposición A.7 *Las relaciones* $\stackrel{\mathbf{sf}}{\Longrightarrow}$, $\stackrel{\mathbf{df}}{\Longrightarrow}$ *y* $\stackrel{\mathbf{sc}}{\Longrightarrow}$ *preservan la corrección, aunque no necesariamente la completud.*

Proposición A.8 *La relación* $\stackrel{\mathbf{ic}}{\Longrightarrow}$ *preserva la corrección y completud.*

Proposición A.9 *La relación* $\mathbf{A} \stackrel{\mathbf{ir}}{\Longrightarrow} \mathbf{B}$ *es válida si y sólo si* $\mathbf{B} \stackrel{\mathbf{ic}}{\Longrightarrow} \mathbf{A}$.

Proposición A.10 *La relación* $\mathbf{A} \stackrel{\mathbf{sr}}{\Longrightarrow} \mathbf{B}$ *es válida si y sólo si* $\mathbf{B} \stackrel{\mathbf{sc}}{\Longrightarrow} \mathbf{A}$.

A.4 Esquemas de análisis sintáctico para autómatas a pila

El método de interpretación dinámica para autómatas a pila propuesto en [104, 107] puede verse como un sistema de inferencia en el cual las transiciones del autómata juegan el papel de las reglas de inferencia [52]. Puesto que los esquemas de análisis sintáctico también son sistemas de inferencia es posible utilizar este formalismo para describir interpretaciones de autómatas a pila en programación dinámica. Abusando del formalismo, utilizaremos la misma notación de los pasos deductivos para describir las transiciones del autómata. En la página 423 puede encontrarse un ejemplo de una descripción de autómata a pila utilizando la estructura de los esquemas de análisis sintáctico.

A.5 Esquemas de análisis sintáctico para TAG y LIG

Los esquemas de análisis sintáctico fueron diseñados para representar algoritmos de análisis sintáctico de gramáticas independientes del contexto. Sin embargo, no resulta complicado extenderlos a formalismos gramaticales que son a su vez extensiones de las gramáticas independientes al contexto, como es el caso de las gramáticas de adjunción de árboles (TAG) y de las gramáticas lineales de índices (LIG). Los mecanismos operaciones de los esquemas de análisis seguirán siendo válidos aunque debemos resaltar que:

- los ítems serán más complejos que los utilizados en el caso de gramáticas independientes del contexto, puesto que los resultados intermedios de los algoritmos de análisis sintáctico deberán tomar nota de ciertas dependencias contextuales;
- las pruebas de validez y corrección tendrán que realizarse de acuerdo a las características propias del formalismo gramatical utilizado en cada momento.

El aumento del tamaño de los ítems, consecuencia del aumento de la información contextual utilizada, puede hacer difícil la lectura de los algoritmos si utilizamos la notación mostrada anteriormente para los pasos deductivos. Aunque ello no presenta inconvenientes desde el punto de vista meramente formal, atenta en cierta medida contra la intención de los esquemas de análisis de proporcionar una visión clara del funcionamiento de los diferentes algoritmos de tal modo que aquellos que no estén interesados en detalles acerca de la corrección y la validez puedan comprender sin demasiada dificultad las bases que rigen el comportamiento de cada algoritmo. Es por ello que en el caso de TAG y LIG hemos decidido utilizar la siguiente notación

$$\frac{I_1, I_2, \ldots, I_n}{I_{n+1}} \quad \langle \text{condición} \rangle$$

para los pasos deductivos en lugar de la clásica

$$I_1, I_2, \ldots, I_n \vdash I_{n+1} \mid \langle \text{condición} \rangle$$

donde $I_1 \ldots I_n$ son los antecedentes, I_{n+1} es el consecuente y $\langle$condición$\rangle$ es el filtro dinámico aplicado al paso deductivo.

A.6 Análisis de complejidad

A.6.1 Complejidad espacial

El análisis de la complejidad espacial es sencillo, pues para calcular el espacio total necesario para almacenar todos los posibles ítems es suficiente con multiplicar las cotas superiores de los distintos componentes. Por ejemplo, dado un un ítem $[h, B^\gamma \to \delta \bullet \nu, k, l \mid p, q]$ tenemos que:

- Los componentes h, k, l, p y q son 5 posiciones de la cadena de entrada y por tanto su cota es la longitud n de la cadena de entrada.
- El componente $B^\gamma \to \delta \bullet \nu$ depende del tamaño de la gramática y por tanto su cota superior es dicho tamaño $|G|$.

Por tanto, la complejidad espacial viene dada por $\mathcal{O}(|G| n^5)$.

A.6.2 Complejidad temporal

Nos referiremos únicamente a la complejidad con respecto a la cadena de entrada, que es la medida usualmente utilizada en la literatura. Dado un algoritmo de análisis sintáctico definido por un esquema de análisis, su complejidad viene dada por el paso deductivo $\mathcal{D}$ más complejo. En una primera aproximación, la complejidad de $\mathcal{D}$ vendrá dada por $\mathcal{O}(n^p)$, donde p es el máximo número de posiciones de la cadena de entrada utilizadas por los antecedentes de $\mathcal{D}$. Sin embargo, un análisis más refinado nos lleva a excluir aquellas posiciones que ocurren una sola vez en $\mathcal{D}$, a las que llamaremos *sin-importancia* [125]. Esto es así puesto que se podría aplicar un paso intermedio implícito $I \vdash I'$ que al ser aplicado redujese un ítem I con q posiciones a otro ítem I' con q' posiciones, con $q' \leq q$, mediante la omisión de las posiciones sin-importancia. El ítem I' tomaría entonces el lugar de I en la parte antecedente de $\mathcal{D}$.

Por ejemplo, el paso deductivo

$$\mathcal{D} = \frac{\begin{array}{l}[h, B^\gamma \to \delta\bullet, k, l \mid p, q],\\ [j, \mathbf{F}^\beta \to \bullet\bot, k, k \mid -, -],\\ [h, A^\gamma \to \delta \bullet B^\gamma\nu, i, j \mid p', q']\end{array}}{[j, \mathbf{F}^\beta \to \bot\bullet, k, l \mid k, l]}$$

que tiene 9 posiciones de la cadena de entrada, presenta un complejidad $\mathcal{O}(n^4)$ puesto que i, p, q, p' y q' son posiciones sin-importancia ya que aparecen una sola vez en $\mathcal{D}$. En consecuencia, para aplicar este paso es suficiente con 4 bucles anidados, cada uno tomando uno de las posiciones h, k, l y j como variable de control.

En ciertos casos la complejidad de un paso deductivo se puede reducir mediante *aplicación parcial*. Un paso deductivo no tiene porqué ser aplicado necesariamente en una sola operación, sino que se puede convertir en una secuencia de pasos intermedios, en cada uno de los cuales se toma un subconjunto de los antecedentes del paso original y se obtiene un resultado en la forma de un ítem intermedio, encargado de trasmitir la información entre pasos intermedios. El resultado del paso deductivo original será igual al resultado del último paso deductivo intermedio. Esta transformación es correcta si todos los ítems antecedentes, así como las condiciones que les afectan, son tomados en cuenta en al menos un paso intermedio. Por ejemplo, el paso deductivo

$$\mathcal{D} = \frac{\begin{array}{l}[\top \to \mathbf{R}^\beta\bullet, j, m \mid k, l],\\ [M^\gamma \to \delta\bullet, k, l \mid -, -],\\ [N^\gamma \to \delta \bullet M^\gamma\nu, i, j \mid p, q]\end{array}}{[N^\gamma \to \delta M^\gamma \bullet \nu, i, m \mid p, q]} \quad \beta \in \mathrm{adj}(M^\gamma)$$

presenta a priori una complejidad $\mathcal{O}(n^7)$ puesto que las 7 posiciones j, m, k, l, i, p y q son utilizadas para determinar la validez de los ítems a combinar. Sin embargo, dicho paso deductivo es equivalente a la aplicación consecutiva de los siguientes pasos $\mathcal{D}^1$ y $\mathcal{D}^2$, el primero de complejidad $\mathcal{O}(n^4)$ y el segundo de complejidad $\mathcal{O}(n^5)$, por lo que la complejidad conjunta de ambos es $\mathcal{O}(n^5)$. El ítem intermedio producido se distingue por la utilización de dobles corchetes:

$$\mathcal{D}^1 = \frac{\begin{array}{l}[\top \to \mathbf{R}^\beta\bullet, j, m \mid k, l],\\ [M^\gamma \to \delta\bullet, k, l \mid -, -],\end{array}}{[[\beta, N^\gamma \to \delta M^\gamma \bullet \nu, j, m]]} \quad \beta \in \mathrm{adj}(M^\gamma)$$

$$\mathcal{D}^2 = \frac{\begin{array}{l}[[\beta, N^\gamma \to \delta M^\gamma \bullet \nu, j, m]]\\ [N^\gamma \to \delta \bullet M^\gamma\nu, i, j \mid p, q]\end{array}}{[N^\gamma \to \delta M^\gamma \bullet \nu, i, m \mid p, q]} \quad \beta \in \mathrm{adj}(M^\gamma)$$

Apéndice B

Algoritmos de análisis sintáctico para CFG

En este apéndice se describen los esquemas de análisis sintáctico de aquellos algoritmos de análisis de gramáticas independientes del contexto que han servido como base para la obtención de algoritmos de análisis sintáctico para LIG y TAG. En consecuencia, se abordan los algoritmos CYK, Earley ascendente y Earley.

B.1 El algoritmo CYK

El algoritmo de Cocke-Younger-Kasami (CYK) para análisis sintáctico de gramáticas independientes del contexto en forma normal de Chomsky [85] fue descubierto por J. Cocke, pero fue publicado independientemente por Younger [234] y Kasami [97], de ahí su nombre.

Es este un algoritmo ascendente puro basado en programación dinámica. A este respecto, la versión original del algoritmo hace uso de una matriz bidimensional indexada por posiciones de la cadena de entrada para almacenar los resultados parciales obtenidos, de tal modo que el elemento A se encuentra en la posición $[i,j]$ de dicha matriz si y sólo si $A \stackrel{*}{\Rightarrow} a_{i+1} \dots a_j$. Sin embargo, seguiremos aquí el enfoque de Sikkel [189] de abstraer las estructuras de datos utilizadas puesto que estas forman parte de los detalles de implementación y no son esenciales al algoritmo. En este contexto, consideraremos que el algoritmo CYK construye un conjunto de ítems

$$\left\{ \ [A,i,j] \mid A \stackrel{*}{\Rightarrow} a_{i+1} \dots a_n \ \right\}$$

Una cadena de entrada $a_1 \dots a_n$ pertenece al lenguaje generado por la gramática si y sólo si el ítem $[S,0,n]$ se encuentra en dicho conjunto.

Esquema de análisis sintáctico B.1 El sistema de análisis sintáctico $\mathbb{P}_{\text{CYK}}$ correspondiente al algoritmo CYK para una CFG $\mathcal{G} = (V_N, V_T, P, S)$ en forma normal de Chomsky, esto es, con producciones de la forma $A \to BC$ o $A \to a$, donde $A, B, C \in V_N$ y $a \in V_T$, y una cadena de entrada $a_1 \dots a_n$ es el que se muestra a continuación:

$$\mathcal{I}_{\text{CYK}} = \left\{ \ [A,i,j] \mid A \in V_N, \ 0 \le i \le j \} \ \right\}$$

$$\mathcal{H}_{\text{CYK}} = \left\{ \ [a,i-1,i] \mid a = a_i \ \right\}$$

$$\mathcal{D}_{\text{CYK}}^{\text{Scan}} = \left\{ \ [a,i,i+1] \vdash [A,i,i+1] \mid A \to a \in P \ \right\}$$

$$\mathcal{D}^{\text{Comp}}_{\text{CYK}} = \{\ [B,i,k],[C,k,j] \vdash [A,i,j] \mid A \rightarrow BC \in P\ \}$$

$$\mathcal{D}_{\text{CYK}} = \mathcal{D}^{\text{Scan}}_{\text{CYK}} \cup \mathcal{D}^{\text{Comp}}_{\text{CYK}}$$

$$\mathcal{F}_{\text{CYK}} = \{\ [S,0,n]\ \}$$

§

Básicamente, podemos decir que el algoritmo CYK comienza por la creación de todos los ítems que se corresponden con reglas terminales (pasos $\mathcal{D}^{\text{Scan}}_{\text{CYK}}$) para posteriormente tratar de combinar en parejas los ítems generados a fin de reconocer las reglas binarias (pasos $\mathcal{D}^{\text{Comp}}_{\text{CYK}}$). El algoritmo termina cuando no se pueden combinar más ítems. En tal caso, si se ha generado algún ítem final, la cadena de entrada ha sido reconocida.

B.2 Una versión ascendente del algoritmo de Earley

El algoritmo CYK presenta una importante limitación, ya que sólo es aplicable a gramáticas en forma normal de Chomsky. Nuestro objetivo ahora es extender el esquema CYK a la clase general de CFG, obteniendo un esquema Earley ascendente [191] en el que se construye un conjunto de ítems

$$\left\{\ [A \rightarrow \alpha \bullet \beta, i, j] \mid \alpha \overset{*}{\Rightarrow} a_{i+1} \ldots j\ \right\}$$

que nos permitirán representar el reconocimiento parcial de producciones mediante la utilización de producciones con punto, al contrario de lo que ocurría en el caso del algoritmo CYK, el cual sólo permitía representar producciones binarias completas. En este sentido, un ítem CYK $[A,i,j]$ puede ser equivalentemente representado por $[A \rightarrow \alpha\bullet, i, j]$, donde $\alpha = BC$ o bien $\alpha = a_i$. El punto en las producciones indica que los elementos gramaticales situado a su izquierda han sido reconocidos. Las producciones son por tanto reconocidas de izquierda a derecha de tal modo que cuando el punto está a la derecha del último elemento del lado derecho de una producción si y sólo si esta ha sido completamente reconocida.

Esquema de análisis sintáctico B.2 El sistema de análisis $\mathbb{P}_{\text{buE}}$ correspondiente al algoritmo Earley ascendente para una gramática independendiente del contexto $\mathcal{G}$ y una cadena de entrada $a_1 \ldots a_n$ es el siguiente:

$$\mathcal{I}_{\text{buE}} = \{\ [A \rightarrow \alpha \bullet \beta, i, j] \mid A \rightarrow \alpha\beta \in P,\ 0 \leq i \leq j\ \}$$

$$\mathcal{H}_{\text{buE}} = \mathcal{H}_{\text{CYK}}$$

$$\mathcal{D}^{\text{Init}}_{\text{buE}} = \{\ \vdash [A \rightarrow \bullet\alpha, i, i]\ \}$$

$$\mathcal{D}^{\text{Scan}}_{\text{buE}} = \{\ [A \rightarrow \alpha \bullet a\beta, i, j], [a, j, j+1] \vdash [A \rightarrow \alpha a \bullet \beta, i, j+1]\ \}$$

$$\mathcal{D}_{\text{buE}}^{\text{Comp}} = \left\{ \ [A \to \alpha \bullet B\beta, i, k], [B \to \gamma\bullet, k, j] \vdash [A \to \alpha B \bullet \beta, i, j] \ \right\}$$

$$\mathcal{D}_{\text{buE}} = \mathcal{D}_{\text{buE}}^{\text{Init}} \cup \mathcal{D}_{\text{buE}}^{\text{Scan}} \cup \mathcal{D}_{\text{buE}}^{\text{Comp}}$$

$$\mathcal{F}_{\text{buE}} = \left\{ \ [S \to \alpha\bullet, 0, n] \ \right\}$$

§

Proposición B.1 **CYK** $\overset{\text{ir}}{\Longrightarrow}\overset{\text{sr}}{\Longrightarrow}\overset{\text{ext}}{\Longrightarrow}$ **buE** .

Efectivamente, el sistema de análisis $\mathbb{P}_{\text{buE}}$ se deriva del sistema de análisis $\mathbb{P}_{\text{CYK}}$ mediante la aplicación de un refinamiento de los ítems y de un refinamiento de pasos deductivos, puesto que se ha descompuesto el paso $\mathcal{D}_{\text{CYK}}^{\text{Comp}}$ en dos pasos $\mathcal{D}_{\text{buE}}^{\text{Init}}$ y $\mathcal{D}_{\text{buE}}^{\text{Comp}}$. Finalmente se ha realizado una extensión del sistema de análisis al considerar, no ya gramáticas en forma normal de Chomsky, sino en forma arbitraria. La prueba formal puede encontrarse en [189].

Con respecto al comportamiento del algoritmo, podemos resumirlo indicando que el análisis comienza con la creación por parte de los pasos $\mathcal{D}_{\text{bu}}^{\text{Init}}$ de los ítems $[A \to \bullet\alpha, i, i]$ para toda regla de la gramática y para toda posición en la cadena de entrada. A continuación se aplican los pasos $\mathcal{D}_{\text{buE}}^{\text{Scan}}$ y $\mathcal{D}_{\text{buE}}^{\text{Comp}}$ con el fin de ir desplazando el punto de las producciones hacia la derecha.

Un paso deductivo $\mathcal{D}_{\text{buE}}^{\text{Scan}}$ es aplicable a ítems de la forma $[A \to \alpha \bullet a\beta, i, j]$ cuando $a_{j+1} = a$, obteniéndose un nuevo ítem $[A \to \alpha a \bullet \beta, i, j+1]$. Es decir, se ha reconocido el símbolo terminal que estaba justo a la derecha del punto.

Un paso deductivo $\mathcal{D}_{\text{buE}}^{\text{Comp}}$ se aplica cuando un ítem tiene una regla con el punto en el extremo derecho. Dado un ítem $[B \to \gamma\bullet, k, j]$ se buscan todos los posibles $[A \to \alpha \bullet B\beta, i, k]$ y se generan nuevos ítems $[A \to \alpha B \bullet \beta, i, j]$ que representan que la subcadena $a_{k+1} \ldots a_j$ puede ser reducida a B y por consiguiente, como la subcadena $a_{i+1} \ldots a_k$ se reduce a α, la subcadena $a_{i+1} \ldots a_j$ se reduce a αB.

El proceso termina cuando no se pueden combinar más ítems. En tal caso, si se ha generado algún ítem final la cadena de entrada pertenece al lenguaje generado por la gramática.

B.3 El algoritmo de Earley

Se puede derivar un esquema de análisis correspondiente al algoritmo de Earley [69] a partir del algoritmo de Earley ascendente mediante la aplicación de un *filtrado dinámico* a este último, de modo que los ítems de la forma $[A \to \bullet\alpha, i, i]$ no sean generados por los pasos Init para todas las posibles producciones y posiciones en la cadena de entrada, sino que sean generados o no dependiendo de la validez de otros items mediante un paso deductivo predictivo, encargándose el paso Init únicamente de la creación de los ítems correspondientes a las producciones del axioma de la gramática. Los ítems válidos son por tanto de la forma

$$\left\{ \ [A \to \alpha \bullet \beta, i, j] \mid \alpha \overset{*}{\Rightarrow} a_{i+1} \ldots j, \ S \overset{*}{\Rightarrow} a_1 \ldots a_i A \delta \ \right\}$$

Esquema de análisis sintáctico B.3 El sistema de análisis $\mathbb{P}_{\text{Earley}}$ correspondiente al algoritmo de Earley para una gramática independiente del contexto $\mathcal{G}$ y una cadena de entrada $A_1 \dots a_n$ es el que se muestra a continuación:

$$\mathcal{I}_{\text{Earley}} = \mathcal{I}_{\text{buE}}$$

$$\mathcal{H}_{\text{Earley}} = \mathcal{H}_{\text{CYK}}$$

$$\mathcal{D}^{\text{Init}}_{\text{Earley}} = \{ \ \vdash [S \to \bullet\alpha, 0, 0] \ \}$$

$$\mathcal{D}^{\text{Scan}}_{\text{Earley}} = \mathcal{D}^{\text{Scan}}_{\text{buE}}$$

$$\mathcal{D}^{\text{Pred}}_{\text{Earley}} = \{ \ [A \to \alpha \bullet B\beta, i, j] \vdash [B \to \bullet\gamma, j, j] \ \}$$

$$\mathcal{D}^{\text{Comp}}_{\text{Earley}} = \mathcal{D}^{\text{Comp}}_{\text{buE}}$$

$$\mathcal{D}_{\text{Earley}} = \mathcal{D}^{\text{Init}}_{\text{Earley}} \cup \mathcal{D}^{\text{Scan}}_{\text{Earley}} \cup \mathcal{D}^{\text{Pred}}_{\text{Earley}} \cup \mathcal{D}^{\text{Comp}}_{\text{Earley}}$$

$$\mathcal{F}_{\text{Earley}} = \mathcal{F}_{\text{buE}}$$

§

Proposición B.2 buE $\stackrel{\text{df}}{\Longrightarrow}$ Earley .

La prueba de dicha transformación puede encontrase en [189]. Con respecto al comportamiento del algoritmo, lo resumiremos indicando que el proceso de análisis comienza con la generación, por parte del conjunto de pasos $\mathcal{D}^{\text{Init}}_{\text{Earley}}$, de los ítems de la forma $[S \to \bullet\alpha, 0, 0]$, donde $S \to \alpha \in P$ es una regla del axioma de la gramática. Estos ítems indican que se está tratando reconocer el axioma de la gramática desde el inicio de la cadena de entrada. Los pasos deductivos $\mathcal{D}^{\text{Scan}}_{\text{Earley}}$ y $\mathcal{D}^{\text{Comp}}_{\text{Earley}}$ se comportan como en el caso del algoritmo de Earley ascendente, mientras que el conjunto de pasos $\mathcal{D}^{\text{Pred}}_{\text{Earley}}$ se encarga de realizar la fase descendente o predictiva del algoritmo, ya que a partir de ítems de la forma $[A \to \alpha \bullet B\beta, i, j]$ genera los ítems $[B \to \bullet\gamma, j, j]$, esto es, se predicen todas las reglas que potencialmente pueden ser útiles en el reconocimiento de la cadena de entrada.

Apéndice C

Análisis sintáctico LR generalizado

A partir del algoritmo de Earley es posible derivar la familia de algoritmos LR para el análisis de gramáticas independientes del contexto sin restricciones. Como resultado, se obtienen algoritmos LR Generalizados con complejidad $\mathcal{O}(n^3)$, donde n es la longitud de la cadena de entrada, en el peor caso pero con un mejor comportamiento en casos prácticos. La obtención de esta complejidad se debe a la utilización de programación dinámica para representar la evolución no determinista de la pila, en lugar de las representaciones de pila estructuradas en grafo utilizadas habitualmente por otros autores. El algoritmo resultante puede extenderse fácilmente al caso de las gramáticas de cláusulas definidas y de las gramáticas lineales de índices. La parte de este capítulo concerniente a las gramáticas independientes del contexto está basada en [12, 11, 10], la parte dedicada a gramáticas de cláusulas definidas está basada en [13] y la parte dedicada a gramáticas lineales de índices está basada en [16].

C.1 Introducción

Los algoritmos de análisis LR, que constituyen una de las más poderosas familias de algoritmos de análisis sintáctico para gramáticas independientes del contexto, constan de dos fases diferenciadas:

- una de compilación, en la que la gramática es compilada en una máquina de estado finito denominada *autómata LR* y dos tablas de acciones e ir-a [6, 5];
- una segunda de ejecución, en la que un autómata a pila utilizando el autómata LR como memoria de estado finito determina la pertenencia o no de las sentencias de entrada al lenguaje generado por la gramática.

La eficiencia del proceso de análisis depende de la segunda fase, en cuya descripción se centra este capítulo.

Las gramáticas que pueden ser analizadas determinísticamente mediante analizadores LR con k símbolos de preanálisis constituyen la clase de las *gramáticas LR*, muy útiles a la hora de describir lenguajes de programación pero que resultan insuficientes cuando se trata de analizar el lenguaje natural. Para ampliar el ámbito de los analizadores LR al caso de lenguajes ambiguos, debemos admitir la posibilidad de que las entradas de la tabla de acciones indiquen más de una acción a realizar en un momento dado. En este caso, nos encontramos con los algoritmos LR no deterministas, también conocidos como *LR generalizado.* Tomita ha propuesto un tipo de análisis LR generalizado [199, 200, 201] que utiliza una pila organizada en forma de grafo para representar todos los posibles caminos de análisis de una cadena de entrada dada. Dicho algoritmo presenta problemas con construcciones gramaticales cíclicas y también con aquellas en las que aparece el fenómeno de la recursión izquierda [135]. Otro inconveniente es que su complejidad con respecto a la sentencia de entrada es dependiente de la forma de la gramática; concretamente, su complejidad es $\mathcal{O}(n^{p+1})$, donde p representa la longitud del mayor lado derecho de una producción en la gramática [100].

Se han propuesto varias mejoras al algoritmo de Tomita, entre las que destacan las de Rekers [157], quien ha modificado el algoritmo original de tal modo que ha eliminado los problemas con la recursividad y la ciclicidad, aunque manteniendo la misma complejidad. Nozohoor-Farshi [135] aporta un algoritmo modificado que evita los problemas con la recursividad izquierda. Kipps [100] transforma el algoritmo original en uno de complejidad cúbica a costa de aumentar los requerimientos de espacio. Otros enfoques diferentes utilizan transformaciones en la gramática con el fin de reducir la complejidad espacial y temporal [184], o bien transformaciones en la construcción del autómata LR, aunque en estos el tratamiento de la ciclicidad es más complejo e incluso, con frecuencia, es evitado [127, 121].

Recientemente se ha despertado un gran interés en el establecimiento de relaciones entre diferentes algoritmos de análisis sintáctico y en cómo un algoritmo puede ser derivado a partir de otro [127, 189, 118]. En la mayoría de los casos, se toma el algoritmo de Earley [69] como punto de partida. En este contexto, proponemos algoritmos LR(1) y LALR(1) generalizados que son derivados mediante una secuencia de transformaciones simples del ya bien conocido algoritmo de Earley, preservando su complejidad cúbica en el peor caso pero obteniendo una mejor complejidad en el caso medio e incluso alcanzando complejidad lineal en el caso de gramáticas LR.

C.2 La relación entre los algoritmos de Earley y LR

Un analizador sintáctico Earley [69] construye, para una gramática $\mathcal{G}$ y una cadena de entrada $a_1 \ldots a_n$, una secuencia de $n+1$ conjuntos de ítems , denominados *itemsets*. Cada ítem individual es de la forma $[A \rightarrow \alpha \bullet \beta, i, j]$, donde $A \rightarrow \alpha \bullet \beta$ indica que la parte α de la producción $A \rightarrow \alpha\beta \in P$ ya ha sido reconocida durante el proceso de análisis, j indica que el ítem está en el itemset j e i es un puntero hacia atrás o *backpointer* que apunta al itemset en el cual se inició el reconocimiento de la producción $A \rightarrow \alpha\beta$. Siempre se cumple que $\alpha \stackrel{*}{\Rightarrow} a_{i+1} \ldots a_j$ y $0 \leq i \leq j$.

El análisis comienza con la creación del ítem $[S \rightarrow \bullet\alpha, 0, 0]$ del itemset 0, donde $S \rightarrow \alpha \in P$ es una producción del axioma de la gramática. A partir de entonces, se van aplicando sucesivamente las tres operaciones *scanner*, *predictor* y *completer* hasta que no se pueden generar más ítems. Estas operaciones se describen a continuación:

scanner es una operación que es aplicable a ítems de la forma $[A \rightarrow \alpha \bullet a\beta, i, j]$ cuando $a_{j+1} = a$, obteniéndose un nuevo ítem $[A \rightarrow \alpha a \bullet \beta, i, j+1]$. Es decir, se ha reconocido el símbolo terminal que estaba justo a la derecha del punto.

predictor es la operación que representa la fase descendente predictiva del algoritmo: a partir de ítems de la forma $[A \rightarrow \alpha \bullet B\beta, i, j]$ se generan ítems $[B \rightarrow \bullet\gamma, j, j]$ para toda producción $B \rightarrow \gamma \in P$, esto es, se predicen todas las producciones que potencialmente pueden ser útiles en el reconocimiento de la cadena de entrada.

completer es una operación que se aplica cuando un ítem tiene una producción con el punto en el extremo derecho. Dado un ítem $[B \rightarrow \gamma\bullet, k, j]$ se buscan todos los posibles $[A \rightarrow \alpha \bullet B\beta, i, k]$ y se generan nuevos ítems $[A \rightarrow \alpha B \bullet \beta, i, j]$ que representan que la subcadena $a_{k+1} \ldots a_j$ puede ser reducida a B y por consiguiente, como la subcadena $a_{i+1} \ldots a_k$ se reduce a α, la subcadena $a_{i+1} \ldots a_j$ se reduce a αB.

El final del proceso de análisis se alcanza cuando se genera un ítem de la forma $[S \rightarrow \alpha\bullet, 0, n]$, indicando que la cadena de entrada pertenece al lenguaje generado por la gramática.

Para describir este algoritmo así como los que restan en este capítulo, haremos uso de los *esquemas de análisis* propuestos por Sikkel [189].

Esquema de análisis sintáctico C.1 El sistema de análisis $\mathbb{P}_{\text{Earley}}$ correspondiente al algoritmo de Earley para una gramática independiente del contexto $\mathcal{G}$ y una cadena de entrada $a_1 \ldots a_n$ es el siguiente [191]:

$$\mathcal{I}_{\text{Earley}} = \left\{ \ [A \rightarrow \alpha \bullet \beta, i, j] \mid A \rightarrow \alpha\beta \in P,\ \ 0 \leq i \leq j \ \right\}$$

$$\mathcal{H}_{\text{Earley}} = \left\{ \ [a, i, i+1] \mid a = a_i \ \right\}$$

$$\mathcal{D}^{\text{Init}}_{\text{Earley}} = \left\{ \ \vdash [S \rightarrow \bullet\alpha, 0, 0] \ \right\}$$

$$\mathcal{D}^{\text{Scan}}_{\text{Earley}} = \left\{ \ [A \rightarrow \alpha \bullet a\beta, i, j], [a, j, j+1] \vdash [A \rightarrow \alpha a \bullet \beta, i, j+1] \ \right\}$$

$$\mathcal{D}^{\text{Pred}}_{\text{Earley}} = \left\{ \ [A \rightarrow \alpha \bullet B\beta, i, j] \vdash [B \rightarrow \bullet\gamma, j, j] \ \right\}$$

$$\mathcal{D}^{\text{Comp}}_{\text{Earley}} = \left\{ \ [A \rightarrow \alpha \bullet B\beta, i, k], [B \rightarrow \gamma\bullet, k, j] \vdash [A \rightarrow \alpha B \bullet \beta, i, j] \ \right\}$$

$$\mathcal{D}_{\text{Earley}} = \mathcal{D}^{\text{Init}}_{\text{Earley}} \cup \mathcal{D}^{\text{Scan}}_{\text{Earley}} \cup \mathcal{D}^{\text{Pred}}_{\text{Earley}} \cup \mathcal{D}^{\text{Comp}}_{\text{Earley}}$$

$$\mathcal{F}_{\text{Earley}} = \left\{ \ [S \rightarrow \alpha\bullet, 0, n] \ \right\}$$

§

La definición de las hipótesis realizada en este sistema de análisis sintáctico se corresponde con la estándar y es la misma que se utilizará en los restantes sistemas de análisis del capítulo. Por consiguiente, no las volveremos a definir explícitamente.

En la figura C.1 se muestra una representación gráfica del funcionamiento del algoritmo que facilita la comprensión del mismo. Cada arco en el dibujo representa la parte de la cadena de entrada abarcada por un ítem . Supongamos que la gramática contiene las dos producciones

$$\begin{array}{l} A \rightarrow \alpha B\gamma \\ B \rightarrow abc \end{array}$$

Supongamos también que ya existe un ítem $[A \rightarrow \alpha \bullet B\gamma, i, k]$ indicando que la parte α de la producción reduce la subcadena $a_i \ldots a_{k-1}$, hecho que se representa el el dibujo mediante el arco de línea continua fina de la izquierda. A partir de este ítem se predecirán ítems con producciones de B, hecho que se refleja en el dibujo por el arco de línea discontinua. En este caso, sólo se generará el ítem $[B \rightarrow \bullet abc, k, k]$. La sucesiva aplicación de tres operaciones scanner para reconocer los terminales a, b y c dará lugar, respectivamente, a los ítems $[B \rightarrow a \bullet bc, k, k+1]$, $[B \rightarrow ab \bullet c, k, k+2]$ y $[B \rightarrow abc\bullet, k, j]$, que se corresponden con los tres arcos de línea continua

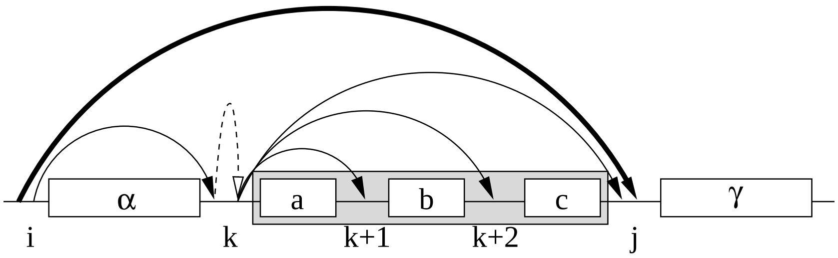

Figura C.1: Representación gráfica del algoritmo de Earley

de la derecha. Combinando este último ítem con $[A \to \alpha \bullet B\gamma, i, k]$ mediante una operación completer, obtenemos el ítem $[A \to \alpha B \bullet \gamma, i, j]$, representado en el dibujo por el arco de línea continua gruesa.

Los pasos $\mathcal{D}^{\text{Pred}}_{\text{Ealrey}}$ del esquema de análisis que acabamos de ver se corresponden con la operación *cerradura* que se utiliza en la construcción del control de estado finito que guía el análisis en los analizadores LR(0). Concretamente, en LR(0) la cerradura de un estado st se realiza incluyendo en st, para cada ítem LR(0) $[A \to \alpha \bullet B\beta]$ de st y para cada producción $B \to \gamma$, todos los ítems $[B \to \bullet\gamma]$. El proceso se repite hasta que no se puede añadir ningún nuevo ítem en st. La misma operación es realizada en tiempo de ejecución en el caso del algoritmo de Earley mediante los pasos Pred si nos percatamos de que las posiciones de la cadena de entrada realmente no intervienen en este tipo de operación. Por tanto, podríamos decir que el esquema de análisis de Earley se corresponde con una "descompilación" de un analizador LR(0).

Siguiendo con la analogía entre Earley y LR(0), los pasos $\mathcal{D}^{\text{Scan}}_{\text{Earley}}$ y $\mathcal{D}^{\text{Comp}}_{\text{Earley}}$ del primero se corresponden con las acciones de desplazamiento y reducción de los analizadores LR(0) clásicos. Una diferencia entre los analizadores Earley y LR estriba en la utilización de gramáticas aumentadas por parte de estos últimos. Este hecho, que se corresponde con la utilización de una producción inicial adicional $\Phi \to S$, es irrelevante en la práctica. En el resto del capítulo, consideraremos que ambos algoritmos utilizan gramáticas aumentadas.

Con el fin de obtener un esquema de análisis más parecido a las operaciones sobre pilas que aparecen en los algoritmos LR clásicos, haremos algunos cambios menores en los pasos Scan y Comp: los componentes i y j de un ítem $[A \to \alpha \bullet \beta, i, j]$ representarán ahora la parte de la cadena de entrada reconocida por el elemento $X \in V$ que aparece inmediatamente antes del punto. Si $\alpha = \varepsilon$, entonces $i = j$. Formalmente, diremos que el algoritmo LR(0) trabaja con ítems

$$\left\{ \; [A \to \alpha' X \bullet \beta, i, j] \mid \exists k, k \leq i, \;\; \alpha' \overset{*}{\Rightarrow} a_{k+1} \dots a_i, \;\; X \overset{*}{\Rightarrow} a_{i+1} \dots a_j \; \right\}$$

Este cambio guarda cierta relación con el propuesto por Nederhof y Satta en la variante del algoritmo de Earley presentada en [131], en donde, con el fin de posibilitar la compartición de los sufijos comunes entre producciones, se suprime durante la fase predictiva la referencia al primer índice presente en los ítems Earley. Con la modificación que nosotros proponemos no es posible compartir sufijos de producciones distintas pero sí es posible compartir ítems que sólo se diferencian por la posición en la que se comenzó a reconocer una producción dada. Consideremos una producción $A \to \alpha B\beta$, y un conjunto D conteniendo d valores de posiciones de la cadena de entrada. En el algoritmo de Earley, si se han generado d ítems $[A \to \alpha \bullet B\beta, i, \mathrm{j}]$, para $i \in D$ y j fijo, y $B \overset{*}{\Rightarrow} a_{\mathrm{j}} \dots a_{\mathrm{k}}$ entonces el paso Comp generará un nuevo conjunto de d ítems de la forma $[A \to \alpha B \bullet \beta, i, \mathrm{k}]$.

Como consecuencia del cambio en las posiciones, el paso Comp necesitará tener ahora m ítems como antecedentes, donde m es la longitud del lado derecho de la producción que se pretende reducir. En el nuevo esquema de análisis $LR(0)$, los pasos Scan y Comp han sido renombrados a Shift y Reduce puesto que muestran un comportamiento idéntico a las operaciones de desplazamiento y de reducción de los analizadores LR.

Esquema de análisis sintáctico C.2 El sistema de análisis $\mathbb{P}_{\mathrm{LR}(0)}$ correspondiente al algoritmo LR(0) para una gramática independiente del contexto $\mathcal{G}$ y una cadena de entrada $a_1 \ldots a_n$ es el que se muestra a continuación:

$$\mathcal{I}_{\mathrm{LR}(0)} = \mathcal{I}_{\mathrm{Earley}}$$

$$\mathcal{D}^{\mathrm{Init}}_{\mathrm{LR}(0)} = \mathcal{D}^{\mathrm{Init}}_{\mathrm{Earley}}$$

$$\mathcal{D}^{\mathrm{Shift}}_{\mathrm{LR}(0)} = \left\{ \ [A \to \alpha \bullet a\beta, i, j], [a, j, j+1] \vdash [A \to \alpha a \bullet \beta, j, j+1] \ \right\}$$

$$\mathcal{D}^{\mathrm{Pred}}_{\mathrm{LR}(0)} = \mathcal{D}^{\mathrm{Pred}}_{\mathrm{Earley}}$$

$$\mathcal{D}^{\mathrm{Reduce}}_{\mathrm{LR}(0)} = \left\{ \begin{array}{l} [B \to X_1 X_2 \cdots X_m \bullet, j_{m-1}, j_m], \ldots, [B \to \bullet X_1 X_2 \cdots X_m, j_0, j_1], [A \to \alpha \bullet B\beta, i, j_0] \\ \vdash [A \to \alpha B \bullet \beta, j_0, j_m] \end{array} \right\}$$

$$\mathcal{D}_{\mathrm{LR}(0)} = \mathcal{D}^{\mathrm{Init}}_{\mathrm{LR}(0)} \cup \mathcal{D}^{\mathrm{Shift}}_{\mathrm{LR}(0)} \cup \mathcal{D}^{\mathrm{Pred}}_{\mathrm{LR}(0)} \cup \mathcal{D}^{\mathrm{Reduce}}_{\mathrm{LR}(0)}$$

$$\mathcal{F}_{\mathrm{LR}(0)} = \mathcal{F}_{\mathrm{Earley}}$$

§

En la figura C.2 se muestra una representación gráfica del modo en que procedería un analizador LR(0) para las dos producciones utilizadas en la figura C.1. En principio, habría un ítem $[A \to \alpha \bullet B\gamma, ?, k]$ con la segunda posición de la cadena de entrada igual a k, mientras que la primera no podemos deducirla de la información existente en el gráfico. En este punto se predice el ítem $[B \to \bullet abc, k, k]$. La aplicación de desplazamientos sobre los terminales a, b y c conlleva la generación de los ítems $[B \to a \bullet bc, k, k+1]$, $[B \to ab \bullet c, k+1, k+2]$ y $[B \to abc\bullet, k+2, j]$ respectivamente. Estos ítems se representan en la figura por arcos de línea continua fina. Precisamente, la principal diferencia entre los algoritmos de Earley y LR(0) queda reflejada en la diferente porción de la cadena de entrada que es abarcada por los ítems correspondientes a los pasos deductivos Scan y Shift. Siguiendo con el proceso de análisis, la reducción de la producción $B \to abc$ conlleva la necesidad de reagrupar los ítems que corresponden al reconocimiento de cada uno de los elementos del lado derecho de la producción, para producir un ítem $[A \to \alpha B \bullet \gamma, k, j]$, representado en la figura por el arco de línea continua gruesa.

C.3 Análisis LR con preanálisis: SLR(1) y LR(1)

A partir del esquema de análisis anterior, podemos obtener el correspondiente al algoritmo SLR(1) mediante la inclusión de un *filtro dinámico* en los pasos Reduce que se encargue de comprobar que el símbolo de preanálisis sea correcto. Para ello debemos utilizar la función *siguiente*, que se define con respecto a la función *primero*. Ambas funciones se muestran a continuación.

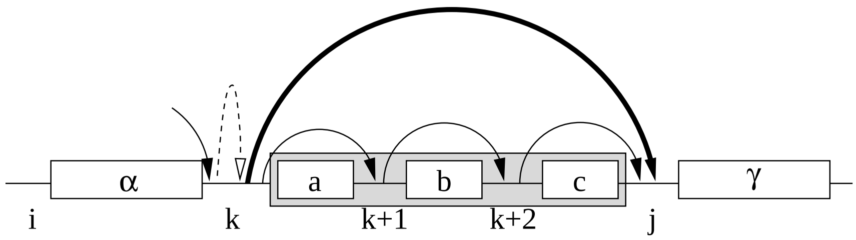

Figura C.2: Representación gráfica del algoritmo LR(0)

Definición C.1 *Un elemento* $a \in V_T \cup \{\epsilon\}$ *pertenece a* $\text{primero}(X)$, *donde* $X \in V$, *si se cumple alguna de las siguientes condiciones:*

- $X = a$
- $X \to \varepsilon \in P$ *y* $a = \varepsilon$
- $X \to Y_1 \cdots Y_i \cdots Y_m \in P$ *y* $a \in \text{primero}(Y_i)$ *y* $\forall_{j=1}^{i-1}\ \varepsilon \in \text{primero}(Y_j)$

La extensión a $\text{primero}(\alpha)$, *donde* $\alpha = X_1 \cdots X_i \cdots X_n \in V$, *es directa:* $a \in \text{primero}(\alpha)$ *si* $a \in \text{primero}(X_1) \cup \cdots \cup \text{primero}(X_i)$ *y* $\varepsilon \notin \text{primero}(X_i)$ *y* $\forall_{j=1}^{i-1}\ \varepsilon \in \text{primero}(X_j)$. *Si* $\alpha \stackrel{*}{\Rightarrow} \varepsilon$ *entonces* $\varepsilon \in \text{primero}(\alpha)$.

Definición C.2 *Un elemento* $a \in V_T \cup \{\$\}$ *pertenece a* $\text{siguiente}(A)$, *donde* $A \in V_N$ *y* $\$$ *es un carácter especial que no pertenece a* V_T *que indica que se ha alcanzado el final de la cadena de entrada, si se cumple alguna de las siguientes condiciones:*

- $a = \$$ *y* A *es el símbolo inicial*
- $A' \to \alpha A \beta \in P$ *y* $a \in (\text{primero}(\beta) - \{\varepsilon\})$
- $A' \to \alpha A \beta \in P$ *y* $\varepsilon \in \text{primero}(\beta)$ *y* $a \in \text{siguiente}(A')$.

La comprobación del símbolo de preanálisis se realiza en los pasos deductivos Reduce, a los cuales se les ha incorporado el filtro dinámico $\exists[a, j, j+1] \in \mathcal{H}_{\text{SLR}}$, $a \in \text{siguiente}(B)$. Dichos pasos quedan ahora como sigue:

$$\mathcal{D}_{\text{SLR}(1)}^{\text{Reduce}} = \left\{ \begin{array}{l} [B \to X_1 X_2 \cdots X_m \bullet, j_{m-1}, j_m], \\ \vdots \\ [B \to \bullet X_1 X_2 \cdots X_m, j_0, j_1], \\ [A \to \alpha \bullet B\beta, i, j_0] \\ \vdash [A \to \alpha B \bullet \beta, j_0, j_m] \mid \\ \exists[a, j_m, j_m + 1] \in \mathcal{H}_{\text{SLR}}, \\ a \in \text{siguiente}(B) \end{array} \right\}$$

Esquema de análisis sintáctico C.3 El sistema de análisis $\mathbb{P}_{\text{SLR}(1)}$ correspondiente al algoritmo SLR(1) para una gramática independiente del contexto $\mathcal{G}$ y una cadena de entrada $a_1 \ldots a_n$ queda definido por:

$$\mathcal{I}_{\text{SLR}(1)} = \mathcal{I}_{\text{lR}(0)}$$

$$\mathcal{D}_{\text{SLR}(1)} = \mathcal{D}_{\text{LR}(0)}^{\text{Init}} \cup \mathcal{D}_{\text{LR}(0)}^{\text{Shift}} \cup \mathcal{D}_{\text{LR}(0)}^{\text{Pred}} \cup \mathcal{D}_{\text{SLR}(1)}^{\text{Reduce}}$$

$$\mathcal{F}_{\mathrm{SLR}(1)} = \mathcal{F}_{\mathrm{Earley}}$$

§

Proposición C.1 LR(0) $\stackrel{\mathbf{df}}{\Longrightarrow}$ SLR(1).

Demostración:

Consideraremos los sistemas de análisis sintáctico $\mathbb{P}_{\mathrm{LR}(0)}$ y $\mathbb{P}_{\mathrm{SLR}(1)}$ para una gramática $\mathcal{G}$ y una cadena de entrada $a_1 \ldots a_n$ cualquiera. Trivialmente $\mathcal{I}_{\mathrm{LR}(0)} \supseteq \mathcal{I}_{\mathrm{SLR}(1)}$ puesto que $\mathcal{I}_{\mathrm{LR}(0)} = \mathcal{I}_{\mathrm{SLR}(1)}$. Para demostrar que $\vdash_{LR(0)} \supseteq \vdash_{SLR(1)}$ bastará con mostrar que $\vdash_{LR(0)} \supseteq \mathcal{D}_{\mathrm{SLR}(1)}$. Los pasos Init, Shift y Pred de $\mathbb{P}_{\mathrm{SLR}(1)}$ son los mismos que en $\mathbb{P}_{\mathrm{LR}(0)}$, mientras que para cada paso $\mathcal{D}^{\mathrm{Reduce}}_{\mathrm{SLR}(1)}$ existe trivialmente un paso $\mathcal{D}^{\mathrm{Reduce}}_{\mathrm{LR}(0)}$ y por tanto una inferencia. □

El análisis LR(1) hace un uso más elaborado del preanálisis que el SLR(1). En vez de limitarse a comprobar si el siguiente elemento de la cadena de entrada es compatible con la reducción en curso, infiere los posibles símbolos de preanálisis para cada ítem en el momento de su creación [6]. Para ello es necesario modificar la estructura de los ítems, ya que se incorpora un nuevo elemento: el símbolo de preanálisis. En consecuencia, los ítems del esquema de análisis LR(1) son el resultado de refinar los ítems del esquema LR(0), ya que cada uno de éstos últimos puede ser considerado como el representante del conjunto de ítems LR(1) que poseen la misma producción con punto y las mismas posiciones pero diferente símbolo de preanálisis:

$$\{\ [A \to \alpha \bullet \beta, b, i, j]\ \}$$

donde b es el símbolo de preanálisis. El esquema de análisis, en el cual el símbolo de preanálisis es inferido en $\mathcal{D}^{\mathrm{Pred}}_{\mathrm{LR}}$ y comprobado en $\mathcal{D}^{\mathrm{Reduce}}_{\mathrm{LR}}$ por medio de un filtro dinámico, es el siguiente:

Esquema de análisis sintáctico C.4 El sistema de análisis $\mathbb{P}_{\mathrm{LR}}$ correspondiente al algoritmo LR(1) para una gramática independiente del contexto $\mathcal{G}$ y una cadena de entrada $a_1 \ldots a_n$ queda definido por:

$$\mathcal{I}_{\mathrm{LR}} = \{\ [A \to \alpha \bullet \beta, b, i, j] \mid A \to \alpha\beta \in P,\ \ b \in V_T,\ \ 0 \le i \le j\ \}$$

$$\mathcal{D}^{\mathrm{Init}}_{\mathrm{LR}} = \{\vdash [S \to \bullet\alpha, \$, 0, 0]\}$$

$$\mathcal{D}^{\mathrm{Shift}}_{\mathrm{LR}} = \{\ [A \to \alpha \bullet a\beta, b, i, j], [a, j, j+1] \vdash [A \to \alpha a \bullet \beta, b, j, j+1]\ \}$$

$$\mathcal{D}^{\mathrm{Pred}}_{\mathrm{LR}} = \{\ [A \to \alpha \bullet B\beta, b, i, j] \vdash [B \to \bullet\gamma, b', j, j] \mid b' \in \mathrm{primero}(\beta b)\ \}$$

$$\mathcal{D}^{\mathrm{Reduce}}_{\mathrm{LR}} = \left\{ \begin{array}{l} [B \to X_1 X_2 \cdots X_m \bullet, b', j_{m-1}, j_m], \ldots, [B \to \bullet X_1 X_2 \cdots X_m, b', j_0, j_1], \\ [A \to \alpha \bullet B\beta, b, i, j_0] \\ \vdash [A \to \alpha B \bullet \beta, b, j_0, j_m] \mid \exists [b', j_m, j_m + 1] \in \mathcal{H}_{\mathrm{LR}},\ \ b' \in \mathrm{primero}(\beta b) \end{array} \right\}$$

$$\mathcal{D}_{\mathrm{LR}} = \mathcal{D}^{\mathrm{Init}}_{\mathrm{LR}} \cup \mathcal{D}^{\mathrm{Shift}}_{\mathrm{LR}} \cup \mathcal{D}^{\mathrm{Pred}}_{\mathrm{LR}} \cup \mathcal{D}^{\mathrm{Reduce}}_{\mathrm{LR}}$$

$$\mathcal{F}_{\mathrm{LR}} = \{\ [S \to \alpha\bullet, \$, 0, n]\ \}$$

§

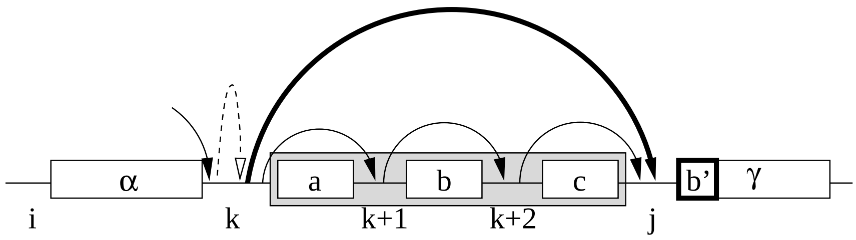

Figura C.3: Representación gráfica del algoritmo LR(1)

En la figura C.3 se representa gráficamente el proceso de análisis LR(1) para el caso de las 2 producciones utilizadas en la figura C.1. La única diferencia relevante con respecto a la figura C.2 es que ahora los pasos Pred calculan b' como posible símbolo de preanálisis, que será comparado con el primer símbolo de γ en el momento de aplicar la reducción de la producción $B \rightarrow abc$.

Proposición C.2 $\mathbf{SLR(1)} \stackrel{\text{ir}}{\Longrightarrow} \mathbf{SLR(1)'} \stackrel{\text{df}}{\Longrightarrow} \mathbf{LR}$.

Demostración:

Como primer paso definiremos el sistema de análisis $\mathbb{P}_{\mathrm{SLR}(1)'}$ para una gramática $\mathcal{G}$ y una cadena de entrada $a_1 \dots a_n$:

$$\mathcal{I}_{\mathrm{SLR}(1)'} = \{\ [A \rightarrow \alpha \bullet \beta, b, i, j] \mid A \rightarrow \alpha\beta \in P,\ b \in V_T,\ 0 \leq i \leq j\ \}$$

$$\mathcal{D}^{\mathrm{Init}}_{\mathrm{SLR}(1)'} = \{\vdash [S \rightarrow \bullet\alpha, \$, 0, 0]\}$$

$$\mathcal{D}^{\mathrm{Shift}}_{\mathrm{SLR}(1)'} = \{\ [A \rightarrow \alpha \bullet a\beta, b, i, j], [a, j, j+1] \vdash [A \rightarrow \alpha a \bullet \beta, b, j, j+1]\ \}$$

$$\mathcal{D}^{\mathrm{Pred}}_{\mathrm{SLR}(1)'} = \{\ [A \rightarrow \alpha \bullet B\beta, b, i, j] \vdash [B \rightarrow \bullet\gamma, b', j, j]\ \}$$

$$\mathcal{D}^{\mathrm{Reduce}}_{\mathrm{SLR}(1)'} = \left\{ \begin{array}{l} [B \rightarrow X_1 X_2 \cdots X_m \bullet, b', j_{m-1}, j_m], \dots, [B \rightarrow \bullet X_1 X_2 \cdots X_m, b', j_0, j_1], \\ [A \rightarrow \alpha \bullet B\beta, b, i, j_0] \\ \vdash [A \rightarrow \alpha B \bullet \beta, b, j_0, j_m] \mid \exists [a, j_m, j_m+1] \in \mathcal{H}_{\mathrm{SLR}},\ a \in \mathrm{siguiente}(B) \end{array} \right\}$$

$$\mathcal{D}_{\mathrm{SLR}(1)'} = \mathcal{D}^{\mathrm{Init}}_{\mathrm{SLR}(1)'} \cup \mathcal{D}^{\mathrm{Shift}}_{\mathrm{SLR}(1)'} \cup \mathcal{D}^{\mathrm{Pred}}_{\mathrm{SLR}(1)'} \cup \mathcal{D}^{\mathrm{Reduce}}_{\mathrm{SLR}(1)'}$$

$$\mathcal{F}_{\mathrm{SLR}(1)'} = \{\ [S \rightarrow \alpha\bullet, \$, 0, n]\ \}$$

Para demostrar que $\mathbf{SLR(1)} \stackrel{\text{ir}}{\Longrightarrow} \mathbf{SLR(1)'}$, definiremos la siguiente función

$$f([A \rightarrow \alpha \bullet \beta, b, i, j]) = \{\ [A \rightarrow \alpha \bullet \beta, i, j]\ \}$$

de la cual se obtiene directamente que $\mathcal{I}_{\mathrm{SLR}(1)} = f(\mathcal{I}_{\mathrm{SLR}(1)'})$ y que $\Delta_{\mathrm{SLR}(1)} = f(\Delta_{\mathrm{SLR}(1)'})$ por inducción en la longitud de las secuencias de derivación. En consecuencia, $\mathbb{P}_{\mathrm{SLR}(1)} \stackrel{\text{ir}}{\longrightarrow} \mathbb{P}_{\mathrm{SLR}(1)'}$, con lo que hemos probado lo que pretendíamos.

Para demostrar que $\mathbf{SLR(1)'} \stackrel{\text{df}}{\Longrightarrow} \mathbf{LR}$, deberemos demostrar que para todo esquema de análisis $\mathbb{P}_{\mathrm{SLR}(1)'}$ y $\mathbb{P}_{\mathrm{LR}}$ se cumple que $\mathcal{I}_{\mathrm{SLR}(1)'} \supseteq \mathcal{I}_{\mathrm{LR}}$ y $\vdash_{SLR(1)'} \supseteq \vdash_{LR}$. Lo primero es trivial, puesto que $\mathcal{I}_{\mathrm{SLR}(1)'} = \mathcal{I}_{\mathrm{LR}}$. Para lo segundo debemos mostrar que $\vdash_{SLR(1)'} \supseteq \mathcal{D}_{\mathrm{LR}}$. Los pasos Init y Scan son idénticos en ambos sistemas de análisis. Por otra parte, es claro que $\mathcal{D}^{\mathrm{Pred}}_{\mathrm{SLR}(1)'} \supseteq \mathcal{D}^{\mathrm{Pred}}_{\mathrm{LR}}$ y $\mathcal{D}^{\mathrm{Reduce}}_{\mathrm{SLR}(1)'} \supseteq \mathcal{D}^{\mathrm{Reduce}}_{\mathrm{LR}}$ puesto que primero(βb) y primero(βb) son condiciones más restrictivas que $\emptyset$ y siguiente(B), respectivamente. $\square$

C.4 LR(1) y LALR(1) con tablas precompiladas

Se puede obtener un algoritmo más eficiente mediante la compilación de los pasos Pred en un autómata finito, llamado *autómata LR*, tal y como se hace en los algoritmos LR clásicos [6]. Dicha compilación se realiza mediante la aplicación de una función *cerradura*. Con ello se consigue evitar la aplicación de pasos Pred en tiempo de ejecución. El conjunto de ítems del nuevo esquema de análisis LR^c es equivalente al conjunto de ítems del esquema LR puesto que los ítems $[A \to \alpha \bullet \beta, b, i, j]$ son simplemente reemplazados por ítems $[st, i, j]$, donde st representa la clase de equivalencia de todos los ítems que contienen la producción con punto $A \to \alpha \bullet \beta$ y el símbolo de preanálisis b'. Además de en eficiencia, también se gana en flexibilidad, puesto que ahora para aplicar un esquema LR(1) o LALR(1) tan sólo es necesario cambiar la fase de compilación, manteniendo sin cambios los pasos deductivos del esquema de análisis. La prueba de corrección del esquema de análisis se fundamenta en la corrección de la estragia de análisis LR utilizada [6].

Esquema de análisis sintáctico C.5 El sistema de análisis $\mathbb{P}_{\text{LR}^c}$, correspondiente al algoritmo LR(1) utilizando tablas precompiladas, dada una gramática independiente del contexto $\mathcal{G}$ y una cadena de entrada $a_1 \dots a_n$ queda definido por:

$$\mathcal{I}_{\text{LR}^c} = \{\ [st, i, j] \mid st \in \mathcal{S},\ 0 \le i \le j\ \}$$

$$\mathcal{D}^{\text{Init}}_{\text{LR}^c} = \{\vdash [st_0, 0, 0]\}$$

$$\mathcal{D}^{\text{Shift}}_{\text{LR}^c} = \{\ [st, i, j], [a, j, j+1] \vdash [st', j, j+1] \mid \text{shift}_{st'} \in \text{acción}(st, a)\ \}$$

$$\mathcal{D}^{\text{Reduce}}_{\text{LR}^c} = \left\{ \begin{array}{l} [st^m, j_{m-1}, j_m], , \dots, [st^1, j_0, j_1], [st^0, i, j_0] \vdash [st, j_0, j_m] \mid \\ \exists [a, j_m, j_m+1] \in \mathcal{H}_{\text{LR}^c},\ \text{reduce}_r \in \text{acción}(st^m, a), \\ st^i \in \text{revela}(st^{i+1}),\ \ st \in \text{ir\_a}(st^0, \text{lhs}(r)), \\ m = \text{longitud}(\text{rhs}(r)) \end{array} \right\}$$

$$\mathcal{D}_{\text{LR}^c} = \mathcal{D}^{\text{Init}}_{\text{LR}^c} \cup \mathcal{D}^{\text{Shift}}_{\text{LR}^c} \cup \mathcal{D}^{\text{Reduce}}_{\text{LR}^c}$$

$$\mathcal{F}_{\text{LR}^c} = \{\ [st_f, 0, n]\ \}$$

donde $\mathcal{S}$ es el conjunto de estados en el autómata LR, $st_0 \in \mathcal{S}$ es el estado inicial, $st_f \in \mathcal{S}$ es el estado final, $\text{lhs}(r)$ es el no-terminal del lado izquierdo de la producción r, y donde $st^i \in \text{revela}(st^{i+1})$ es equivalente a $st^{i+1} \in \text{ir\_a}(st^i, X)$ cuando $X \in V_N$ y a $\text{shift}_{st^{i+1}} \in \text{acción}(st^i, X)$ cuando $X \in V_T$. Finalmente *acción* e *ir_a* se refieren a las tablas[1] en las que se ha codificado el comportamiento del autómata LR:

- La tabla de acciones determina qué acciones se deben realizar para cada combinación de estado y símbolo de preanálisis. Concretamente, en el caso de se deba realizar un desplazamiento indica el estado al que hay que desplazar y en el caso de acciones de reducción la producción que deberá ser aplicada.

[1]Se puede incrementar la velocidad del analizador sintáctico transformando dichas tablas en código, obteniendo en contrapartida un ejecutable de mayor tamaño [86].

- La tabla de ir_a determina cual será el nuevo estado del autómata LR después de realizar una reducción. Para acceder a una entrada se utiliza el estado actual y el no-terminal situado en el lado izquierdo de la producción que se ha reducido.

§

C.5 LR(1) y LALR(1) con complejidad $\mathcal{O}(n^3)$

La utilización de los pasos Reduce en los esquemas de análisis anteriores incrementa la complejidad de los algoritmos a $\mathcal{O}(n^{p+1})$, donde p es la máxima longitud de la parte derecha de una producción en la gramática que se esté considerando. Por consiguiente, sólo se obtendrá una complejidad de orden cúbico cuando toda producción tenga a lo sumo dos elementos en su parte derecha. Para obtener una complejidad de orden $\mathcal{O}(n^3)$ sin restringir la longitud de las producciones, seguiremos la sugerencia de Johnson [88] de utilizar el método denominado *binarización implícita de producciones*, descrito por Lang[2] en [107]. Básicamente, este proceso consiste en transformar una reducción de una producción con m elementos en su parte derecha en m reducciones de producciones que poseen a lo sumo 2 elementos en su parte derecha.

Siguiendo este enfoque, la reducción de la producción

$$A_{r,0} \rightarrow A_{r,1} \ldots A_{r,n_r}$$

se transformaría en la reducción de las siguientes $n_r + 1$ producciones:

$$\begin{array}{l} A_{r,0} \rightarrow \nabla_{r,0} \\ \nabla_{r,0} \rightarrow A_{r,1}\ \nabla_{r,1} \\ \vdots \\ \nabla_{r,n_r-1} \rightarrow A_{r,n_r}\ \nabla_{r,n_r} \\ \nabla_{r,n_r} \rightarrow \varepsilon \end{array}$$

donde los símbolos nabla son *frescos*, esto es, diferentes de cualquier otro símbolo de la gramática. Un aspecto importante a considerar es que no es necesario tratar explícitamente la existencia de esas $n_r + 1$ nuevas producciones. Bien al contrario, el algoritmo trabaja siempre sobre las producciones originales. Esto se consigue introduciendo los símbolos ∇ directamente en el interior de las producciones. En efecto, la producción

$$A_{r,0} \rightarrow A_{r,1} \ldots A_{r,n_r}$$

pasa a ser vista por el algoritmo como constituida por los elementos

$$A_{r,0} \rightarrow \nabla_{r,0}\ A_{r,1}\ \nabla_{r,1}\ \ldots A_{r,n_r}\ \nabla_{r,n_r}$$

de tal modo que los símbolos ∇ sirven de indicadores para señalar la parte de la producción que ha sido reducida, o equivalentemente, cuáles de las reducciones binarias han sido aplicadas. Por ejemplo, un ítem conteniendo ∇_{r,n_r} indicará que se ha reducido la producción $\nabla_{r,n_r} \rightarrow \varepsilon$, mientras que un ítem conteniendo ∇_{r,n_r-1} indicará que ya han sido reducidas las producciones $\nabla_{r,n_r} \rightarrow \varepsilon$ y $\nabla_{r,n_r-1} \rightarrow A_{r,n_r}\ \nabla_{r,n_r}$. La presencia de $\nabla_{r,0}$ indicará que toda la parte derecha de la producción original ha sido reducida y que ya sólo queda por generar el símbolo $A_{r,0}$, correspondiente al lado izquierdo de la producción[3].

[2] Un método equivalente llamado *bilinearización* es descrito por Leermakers en [111].

[3] Es interesante señalar la similitud entre $\nabla_{r,i}$ y la producción con punto $A_{r,0} \rightarrow \alpha \bullet \beta$, donde $\alpha = A_{r,1} \ldots A_{r,i}$ y $\beta = A_{r,i+1} \ldots a_{r,n_r}$.

De lo anterior se deduce que este nuevo tratamiento de las reducciones lleva aparejado un cambio en la forma de los ítems , puesto que se deberá incluir un nuevo elemento que representará o bien un símbolo ∇, indicando que los elementos $A_{r,i+1} \ldots A_{r,n_r}$ de la producción ya han sido reducidos, o bien un símbolo perteneciente a la gramática, que puede ser un terminal si el ítem ha sido generado como resultado de un desplazamiento o un no-terminal si el ítem ha sido generado al finalizar una reducción. En este último caso, dicho no-terminal se corresponderá con el símbolo situado en el lado izquierdo de la producción reducida. En resumen, siguiendo la terminología de los esquemas de análisis diremos que aplicamos un refinamiento a los ítems.

Con respecto a los pasos deductivos, aplicaremos un refinamiento a los pasos Shift, puesto que ahora deberemos diferenciar si desplazamos el primer elemento del lado derecho de una producción (InitShift) o cualquier otro elemento del lado derecho (Shift). Por su parte, los pasos Reduce también deben ser refinados y reemplazados por los siguientes tres pasos: Sel para indicar que una producción a sido seleccionada para reducción, Red para realizar la reducción implícita de una producción binaria y Head para indicar la finalización de la reducción y el reconocimiento del lado izquierdo de la producción.

Esquema de análisis sintáctico C.6 El sistema de análisis $\mathbb{P}_{\text{LR}^3}$, correspondiente al algoritmo LR(1) con complejidad cúbica, dada una gramática independiente del contexto $\mathcal{G}$ y una cadena de entrada $a_1 \ldots a_n$ queda definido por:

$$\mathcal{I}_{\text{LR}^3} = \left\{ \; [A, st, i, j] \cup [\nabla_{r,s}, st, i, j] \mid A \in V_N \cup V_T, \;\; st \in \mathcal{S}, \;\; 0 \leq i \leq j \; \right\}$$

$$\mathcal{D}^{\text{Init}}_{\text{LR}^3} = \left\{ \; \vdash [-, st_0, 0, 0] \; \right\}$$

$$\mathcal{D}^{\text{InitShift}}_{\text{LR}^3} = \left\{ \begin{array}{l} [A, st, i, j] \vdash [A_{r,1}, st', j, j+1] \mid \\ \exists [a, j, j+1] \in \mathcal{H}_{\text{LR}^3}, \;\; A_{r,1} = a, \;\; \text{shift}_{st'} \in \text{acción}(st, a), \;\; A \in V \end{array} \right\}$$

$$\mathcal{D}^{\text{Shift}}_{\text{LR}^3} = \left\{ \begin{array}{l} [A_{r,s}, st, i, j] \vdash [A_{r,s+1}, st', j, j+1] \mid \\ \exists [a, j, j+1] \in \mathcal{H}_{\text{LR}^3}, \;\; A_{r,s+1} = a, \;\; \text{shift}_{st'} \in \text{acción}(st, a) \end{array} \right\}$$

$$\mathcal{D}^{\text{Sel}}_{\text{LR}^3} = \left\{ \begin{array}{l} [A, st, i, j] \vdash [\nabla_{r,n_r}, st, j, j] \mid \\ \exists [a, j, j+1] \in \mathcal{H}_{\text{LR}^3}, \;\; \text{reduce}_r \in \text{acción}(st, a), \;\; A \in V \end{array} \right\}$$

$$\mathcal{D}^{\text{Red}}_{\text{LR}^3} = \left\{ \begin{array}{l} [\nabla_{r,s}, st, k, j], [A_{r,s}, st, i, k] \vdash [\nabla_{r,s-1}, st', i, j] \mid \\ st' \in \text{revela}(st) \end{array} \right\}$$

$$\mathcal{D}^{\text{Head}}_{\text{LR}^3} = \left\{ \; [\nabla_{r,0}, st, i, j] \vdash [A_{r,0}, st', i, j] \mid st' \in \text{ir\_a}(st, A_{r,0}) \; \right\}$$

$$\mathcal{D}_{\text{LR}^3} = \mathcal{D}^{\text{Init}}_{\text{LR}^3} \cup \mathcal{D}^{\text{InitShift}}_{\text{LR}^3} \cup \mathcal{D}^{\text{Shift}}_{\text{LR}^3} \cup \mathcal{D}^{\text{Sel}}_{\text{LR}^3} \cup \mathcal{D}^{\text{Red}}_{\text{LR}^3} \cup \mathcal{D}^{\text{Head}}_{\text{LR}^3}$$

$$\mathcal{F}_{\text{LR}^3} = \left\{ \; [S, st_f, 0, n] \; \right\}$$

§

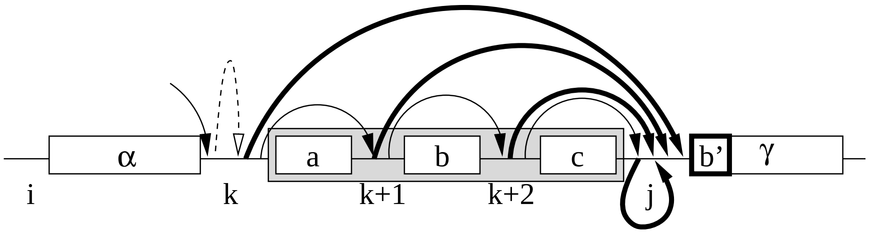

Figura C.4: Representación gráfica del algoritmo LR(1) con complejidad cúbica

En la figura C.4 se muestra cómo procede el algoritmo que se acaba de proponer en el caso del análisis de las producciones utilizadas en la figura C.1. El funcionamiento es prácticamente idéntico al de los algoritmos anteriores en lo que respecta a las acciones de desplazamiento de los terminales a, b y c. Las diferencias surgen en el proceso de la reducción de la producción $B \rightarrow abc$. Ahora, esta operación de reducción se realiza en varias fases, comenzando por la aplicación de un paso Sel que genera el ítem representado en la figura C.4 por el arco de trazo grueso de la parte inferior. Este ítem se combina con el resultante del desplazamiento del terminal c mediante un paso Red para realizar la primera reducción binaria implícita, dando lugar al arco de trazo grueso situado más a la derecha en la parte superior de la figura. Este ítem se combina a su vez con el resultante del desplazamiento del terminal b mediante otro paso Red para generar el ítem representado por el segundo arco de trazo grueso de la parte superior de la figura. Este ítem también se combina mediante un paso Red, esta vez con el ítem resultante del desplazamiento del terminal a, dando lugar al ítem correspondiente a la finalización de la reducción de la producción $B \rightarrow abc$, representado por el primer arco de trazo grueso en la figura C.4. Para finalizar, ya sólo queda aplicar un paso Head.

Proposición C.3 $\mathbf{LR^c} \stackrel{\mathbf{ir}}{\Longrightarrow} \mathbf{LR^{c'}} \stackrel{\mathbf{sr}}{\Longrightarrow} \mathbf{LR^3}$.

Demostración:

Como primer paso definiremos el sistema de análisis $\mathbb{P}_{\mathrm{LR}^{c'}}$ para una gramática $\mathcal{G}$ y una cadena de entrada $a_1 \dots a_n$:

$$\mathcal{I}_{\mathrm{LR}^{c'}} = \left\{ \ [A, st, i, j] \cup [\nabla_{r,s}, st, i, j] \mid A \in V_N \cup V_T, \ st \in \mathcal{S}, \ 0 \leq i \leq j \ \right\}$$

$$\mathcal{D}^{\mathrm{Init}}_{\mathrm{LR}^{c'}} = \{ \vdash [-, st_0, 0, 0] \}$$

$$\mathcal{D}^{\mathrm{Shift}}_{\mathrm{LR}^{c'}} = \left\{ \ [A, st, i, j], [a, j, j+1] \vdash [a, st', j, j+1] \mid \mathrm{shift}_{st'} \in \mathrm{acción}(st, a), \ A \in V \ \right\}$$

$$\mathcal{D}^{\mathrm{Reduce}}_{\mathrm{LR}^{c'}} = \left\{ \begin{array}{l} [A_{r,m}, st^m, j_{m-1}, j_m], \dots, [A_{r,1} st^1, j_0, j_1], [A, st^0, i, j_0] \vdash [A_{r,0}, st, j_0, j_m] \mid \\ \exists [a, j_m, j_m + 1] \in \mathcal{H}_{\mathrm{LR}^{c'}}, \ \mathrm{reduce}_r \in \mathrm{acción}(st^m, a), \ st^i \in \mathrm{revela}(st^{i+1}), \\ st \in \mathrm{ir\_a}(st^0, \mathrm{lhs}(r)), \ m = \mathrm{longitud}(\mathrm{rhs}(r)), \ A \in V \end{array} \right\}$$

$$\mathcal{D}_{\mathrm{LR}^{c'}} = \mathcal{D}^{\mathrm{Init}}_{\mathrm{LR}^{c'}} \cup \mathcal{D}^{\mathrm{Shift}}_{\mathrm{LR}^{c'}} \cup \mathcal{D}^{\mathrm{Reduce}}_{\mathrm{LR}^{c'}}$$

$$\mathcal{F}_{\mathrm{LR}^{c'}} = \left\{ \ [st_f, 0, n] \ \right\}$$

Para demostrar que $\mathbf{LR^c} \stackrel{\mathbf{ir}}{\Longrightarrow} \mathbf{LR^{c'}}$, definiremos la siguiente función

$$f([A, st, i, j]) = \{ \ [st, i, j] \ \}$$

de la cual se obtiene directamente que $\mathcal{I}_{\mathrm{LR}^c} = f(\mathcal{I}_{\mathrm{LR}^{c\prime}})$ y que $\Delta_{LR^c} = f(\Delta_{LR^{c\prime}})$. En consecuencia, $\mathbb{P}_{\mathrm{SLR}(1)} \xrightarrow{\mathbf{ir}} \mathbb{P}_{\mathrm{SLR}(1)'}$, con lo que hemos probado lo que pretendíamos.

Para demostrar que $\mathbf{LR^{c\prime}} \stackrel{\mathbf{ir}}{\Longrightarrow} \mathbf{LR^3}$, deberemos demostrar que para todo esquema de análisis $\mathbb{P}_{\mathrm{LR}^{c\prime}}$ y $\mathbb{P}_{\mathrm{LR}^3}$ se cumplen $\mathcal{I}_{\mathrm{LR}^{c\prime}} \subseteq \mathcal{I}_{\mathrm{LR}^3}$ y $\overset{*}{\vdash}_{LR^{c\prime}} \subseteq \overset{*}{\vdash}_{LR^3}$. Lo primero es trivial, puesto que $\mathcal{I}_{\mathrm{LR}^{c\prime}} = \mathcal{I}_{\mathrm{LR}^3}$. Para lo segundo debemos mostrar que $\mathcal{D}_{\mathrm{LR}^{c\prime}} \subseteq \overset{*}{\vdash} LR^3$. Los pasos Init son idénticos en ambos casos. Un paso Shift de $\mathbb{P}_{\mathrm{LR}^{c\prime}}$ se corresponde bien con un paso Shift o bien con un paso InitShift de $\mathbb{P}_{\mathrm{LR}^3}$. Un paso Reduce de $\mathbb{P}_{\mathrm{LR}^{c\prime}}$ es igual a la secuencia de derivaciones de $\mathbb{P}_{\mathrm{LR}^3}$ encabezada por un paso Sel, seguida de m pasos Red, uno por cada elemento del lado derecho de la producción que se está reduciendo, y finalmente un paso Head. □

C.5.1 Análisis de complejidad

Tomamos la longitud n de la cadena de entrada como parámetro de la complejidad puesto que tanto el tamaño de la gramática como el del autómata LR son fijos para una gramática dada. A partir del esquema de análisis sintáctico $\mathbf{LR^3}$ es fácil ver que la complejijdad del algoritmo con respecto a la cadena de entrada es $\mathcal{O}(n^3)$ puesto que los pasos deductivos que involucran un mayor número de posiciones de la cadena de entrada son los del conjunto $\mathcal{D}^{\mathrm{Red}}_{\mathrm{LR}^3}$, cada uno de los cuales relaciona las posiciones i, j y k.

Este resultado es equivalente al que obtendríamos si siguiesemos un método clásico de cálculo de complejidad, como el que se relata a continuación. Puesto que cada ítem posee dos posiciones de la cadena de entrada, habrá $\mathcal{O}(n^2)$ ítems . Cada paso deductivo ejecuta un número limitado de operaciones por cada ítem. A este respecto, el peor caso corresponde a los pasos $\mathcal{D}^{\mathrm{Red}}_{\mathrm{LR}^3}$, que pueden combinar $\mathcal{O}(n^2)$ ítems de la forma $[\nabla_{r,s}, st, k, j]$ con $\mathcal{O}(n)$ ítems de las forma $[A_{r,s}, st, i, k]$ y por consiguiente estos pasos deductivos presentan una complejidad $\mathcal{O}(n^3)$.

Al igual que se hacía en el caso del algoritmo de Earley original [69] podemos agrupar los ítems en conjuntos de ítems denominados *itemsets*. Existe un itemset por cada uno de los caracteres en la cadena de entrada[4]. En este caso, para la clase de las gramáticas con número limitado de ítems [5], en las cuales el número máximo de ítems que puede contener un itemset está acotado, se obtiene complejidad lineal, tanto espacial como temporal. El interés práctico de este resultado se debe a que esta clase de gramáticas incluye a la familia de las gramáticas LR y en consecuencia, es posible realizar el análisis sintáctico en tiempo lineal cuando la cadena de entrada es localmente determinista.

Si considerásemos el tamaño de la gramática como variable en el cálculo de la complejidad, los análisis efectuados por Johnson [88] con respecto al algoritmo de Tomita serían también aplicables al algoritmo propuesto.

C.6 Tabulación del autómata a pila LR

El esquema de análisis LR^3 corresponde a una interpretación en programación dinámica o tabulación de los algoritmos de análisis sintáctico LR(1) o LALR(1) utilizando un sistema de inferencia basado en ítems S^1 [52, pp. 173–175]. Para incorporar el algoritmo en la estructura común de análisis sintáctico propuesta por Lang en [107] es necesario transformarlo en un conjunto de transiciones de autómata a pila. Por [52] sabemos que utilizando ítems S^1, es decir,

[4]ítems con el cuarto componente igual a j pertenecen al itemset j.

[5]*bounded item grammars*, llamadas *bounded state grammars* en [69].

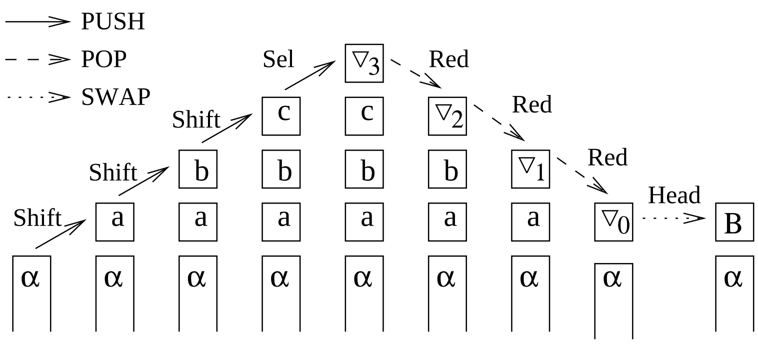

Figura C.5: Representación gráfica de la evolución de la pila en un algoritmo LR

ítems que contienen únicamente el elemento en la cima de la pila, podemos obtener una interpretación correcta en programación dinámica de los autómata débilmente predictivos, clase a la que pertenecen los autómatas a pila LR.

En la figura C.5 podemos observar cómo el algoritmo descrito en la sección C.5 trabaja sobre una pila. En esta figura, las cajas representan ítems. Para facilitar su comprensión, en cada caja hemos situado únicamente el primer elemento del ítem que representa. Por la misma razón los símbolos nabla poseen un solo índice, que indica su posición en la producción $B \rightarrow abc$. Los pasos deductivos Shift y Sel apilan un nuevo ítem en la cima de la pila. Los pasos Red desapilan los dos ítems de la cima y apilan uno en lo más alto de la pila. Los pasos Head simplemente reemplazan el ítem de la cima por otro ítem.

Las transiciones de la figura C.5 se corresponden exactamente con las transiciones de autómatas a pila descritas en [107]. En efecto, cada autómata a pila puede ser descrito utilizando las siguientes transiciones:

SWAP:	$(B \longmapsto C)(A) = C$	tal que $B = A$
PUSH:	$(B \longmapsto BC)(A) = C$	tal que $B = A$
POP:	$(DB \longmapsto C)(E, A) = C$	tal que $(D, B) = (E, A)$

donde A, B, C, D y E son ítems y donde las pilas crecen hacia la derecha.

Esquema de compilación C.7 Si consideramos los pasos deductivos Head como transiciones SWAP, los pasos Init, InitShift, Shift y Sel como transiciones PUSH y los pasos Red como transiciones POP, obtenemos un esquema de compilación que, para una gramática independiente del contexto $\mathcal{G}$ y una estrategia de análisis sintáctico LR(1) o LALR(1), queda definido por el siguiente conjunto de reglas de compilación y el elemento final $[S, st_f]$.

[INIT]	$\$_0 \longmapsto \$_0[-, st_0]$	
[INITSHIFT]	$[A, st] \overset{a}{\longmapsto} [A, st]\ [A_{r,1}, st']$	$A_{r,1} = a,\ \text{shift}_{st'} \in \text{acción}(st, a),\ A \in V$
[SHIFT]	$[A_{r,s}, st] \overset{a}{\longmapsto} [A_{r,s}, st]\ [A_{r,s+1}, st']$	$A_{r,s+1} = a,\ \text{shift}_{st'} \in \text{acción}(st, a)$
[SEL]	$[A, st] \longmapsto [A_{r,n_r}, st]\ [\nabla_{r,n_r}, st]$	$\text{reduce}_r \in \text{acción}(st, \text{lookahead}),\ A \in V$
[RED]	$[A_{r,s}, st]\ [\nabla_{r,s}, st] \longmapsto [\nabla_{r,s-1}, st']$	$st' \in \text{revela}(st)$
[HEAD]	$[\nabla_{r,0}, st] \longmapsto [A_{r,0}, st']$	$st' \in \text{ir\_a}(st, A_{r,0})$

§

Esquema de análisis sintáctico C.8 La interpretación en programación dinámica del autómata a pila correspondiente a los algoritmos LR(1) y LALR(1) para una gramática independiente del contexto $\mathcal{G}$ y una cadena de entrada $a_1 \dots a_n$ queda definido por el el sistema de análisis $\mathbb{P}_{\mathrm{LR^{S1}}}$ que se muestra a continuación:

$$\mathcal{I}_{\mathrm{LR^{S1}}} = \left\{ \ [A, st, i, j] \cup [\nabla_{r,s}, st, i, j] \mid A \in V_N \cup V_T, \ \ st \in \mathcal{S}, \ \ 0 \leq i \leq j \ \right\}$$

$$\mathcal{D}^{\mathrm{Init}}_{\mathrm{LR^{S1}}} = \left\{ \ \vdash [-, st_0, 0, 0] \ \right\}$$

$$\mathcal{D}^{\mathrm{InitShift}}_{\mathrm{LR^{S1}}} = \left\{ \begin{array}{l} [A, st, i, j] \vdash \quad [A_{r,1}, st', j, j+1] \mid \\ \exists [a, j, j+1] \in \mathcal{H}_{\mathrm{LR^{S1}}}, \ \ A_{r,1} = a, \ \ \mathrm{shift}_{st'} \in \mathrm{acción}(st, a), \ \ A \in V \end{array} \right\}$$

$$\mathcal{D}^{\mathrm{Shift}}_{\mathrm{LR^{S1}}} = \left\{ \begin{array}{l} [A_{r,s}, st, i, j] \vdash [A_{r,s+1}, st', i, j+1] \mid \\ \exists [a, j, j+1] \in \mathcal{H}_{\mathrm{LR^{S1}}}, \ \ A_{r,s+1} = a, \ \ \mathrm{shift}_{st'} \in \mathrm{acción}(st, a) \end{array} \right\}$$

$$\mathcal{D}^{\mathrm{Sel}}_{\mathrm{LR^{S1}}} = \left\{ \begin{array}{l} [A, st, i, j] \vdash [\nabla_{r,n_r}, st, i, j+1] \mid \\ \exists [a, j, j+1] \in \mathcal{H}_{\mathrm{LR^{S1}}}, \ \ \mathrm{reduce}_r \in \mathrm{acción}(st, a), \ \ A \in V \end{array} \right\}$$

$$\mathcal{D}^{\mathrm{Red}}_{\mathrm{LR^{S1}}} = \left\{ \begin{array}{l} [\nabla_{r,s}, st, i, k] \quad [A_{r,s}, st, k, j] \vdash [\nabla_{r,s-1}, st', i, j] \mid \\ st' \in \mathrm{revela}(st) \end{array} \right\}$$

$$\mathcal{D}^{\mathrm{Head}}_{\mathrm{LR^{S1}}} = \left\{ \begin{array}{l} [\nabla_{r,0}, st, i, j] \vdash [A_{r,0}, st', i, j] \mid \\ st' \in \mathrm{ir\_a}(st, A_{r,0}) \end{array} \right\}$$

$$\mathcal{D}_{\mathrm{LR^{S1}}} = \mathcal{D}^{\mathrm{Init}}_{\mathrm{LR^{S1}}} \cup \mathcal{D}^{\mathrm{InitShift}}_{\mathrm{LR^{S1}}} \cup \mathcal{D}^{\mathrm{Shift}}_{\mathrm{LR^{S1}}} \cup \mathcal{D}^{\mathrm{Sel}}_{\mathrm{LR^{S1}}} \cup \mathcal{D}^{\mathrm{Red}}_{\mathrm{LR^{S1}}} \cup \mathcal{D}^{\mathrm{Head}}_{\mathrm{LR^{S1}}}$$

$$\mathcal{F}_{\mathrm{LR^{S1}}} = \left\{ \ [S, st_f, 0, n] \ \right\}$$

§

A continuación trataremos de mostrar mediante ejemplos el funcionamiento de un analizador sintáctico que implementa el esquema de análisis LR^{S1}. En primer lugar, veremos como la utilización de símbolos de preanálisis ayuda a mejorar la eficiencia evitando la exploración de caminos infructuosos. Posteriormente veremos cómo se tratan los ciclos y las producciones recursivas.

Ejemplo C.1 Tomemos la gramática independiente del contexto $\mathcal{G}_1$, simple pero de alto valor didáctico:

$$\begin{array}{ll} (0) & \Phi \to S \\ (1) & S \to Aa \\ (2) & S \to Bb \\ (3) & A \to cd \\ (4) & B \to cd \end{array}$$

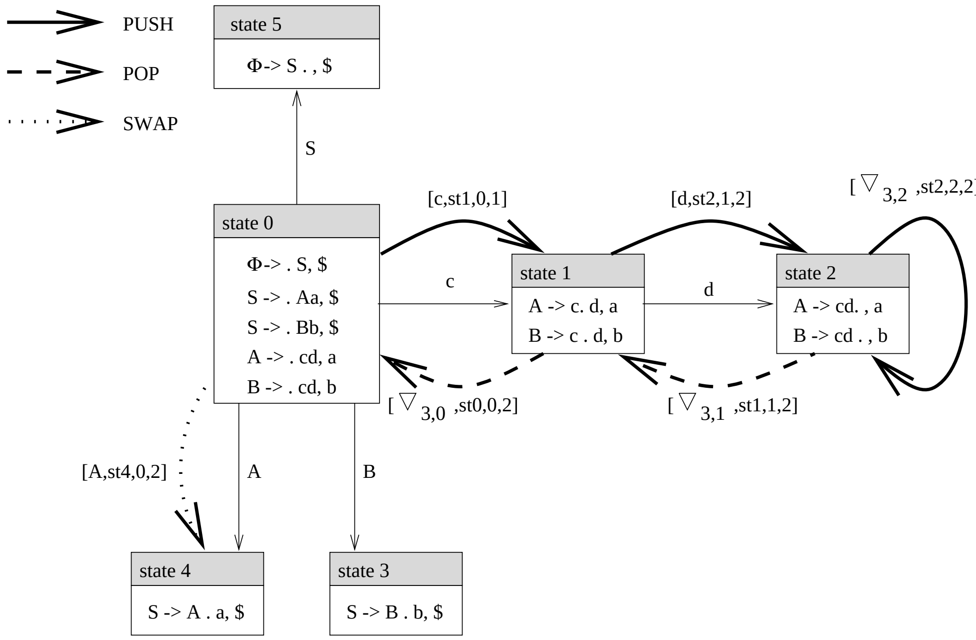

Figura C.6: Transiciones para la cadena de entrada *cda* en el autómata LALR(1) de la gramática $\mathcal{G}_1$

El lenguaje generado por $\mathcal{G}_1$ es el conjunto $\{cda, cdb\}$. En la figura C.6 mostramos el autómata LALR(1) para esta gramática y las transiciones correspondientes al análisis del prefijo *cd* de la cadena de entrada *cda*. El análisis comienza con el autómata en el estado 0. La primera acción a realizar es el desplazamiento del terminal c, aplicando para ello un paso InitShift. Como resultado, el ítem $[c, st1, 0, 1]$ es apilado y el nuevo estado del autómata LR es el 1. Una vez en este estado, sabemos que estamos intentando analizar la entrada mediante las producciones 3 y/o 4 (en un analizador Earley, esto correspondería a predecir las producciones 3 y 4). La siguiente acción a realizar en el desplazamiento del terminal d, para lo cual aplicamos un paso Shift, apilando el ítem $[d, st2, 1, 2]$ y convirtiendo al estado 2 en el nuevo estado actual del autómata LR. El estado 2 indica que tanto al producción 3 como la 4 reconocen el prefijo *cd*, pero el símbolo de preanálisis nos permite determinar que continuar el análisis por la producción 4 es infructuoso, puesto que no es compatible con el siguiente símbolo en la cadena de entrada. Un analizador sintáctico sin preanálisis se hubiese visto obligado a explorar las dos alternativas representadas por las producciones 3 y 4, para descubrir más tarde que sólo la producción 3 puede ser aplicada. La utilización de preanálisis incrementa el dominio determinista del analizador, permitiéndole obtener una mayor eficiencia.

En nuestro caso, el análisis continua mediante la reducción de la producción 3

1. Sel: PUSH $[\nabla_{3,2}, st2, 2, 2]$
2. Red: POP $[\nabla_{3,2}, st2, 2, 2]$ y $[d, st2, 1, 2]$ para dar $[\nabla_{3,1}, st1, 1, 2]$
3. Red: POP $[\nabla_{3,1}, st1, 1, 2]$ y $[c, st1, 0, 1]$ para dar $[\nabla_{3,0}, st0, 0, 2]$
4. Head: SWAP $[\nabla 3, 0, st0, 0, 2]$ para dar $[A, st4, 0, 2]$

¶

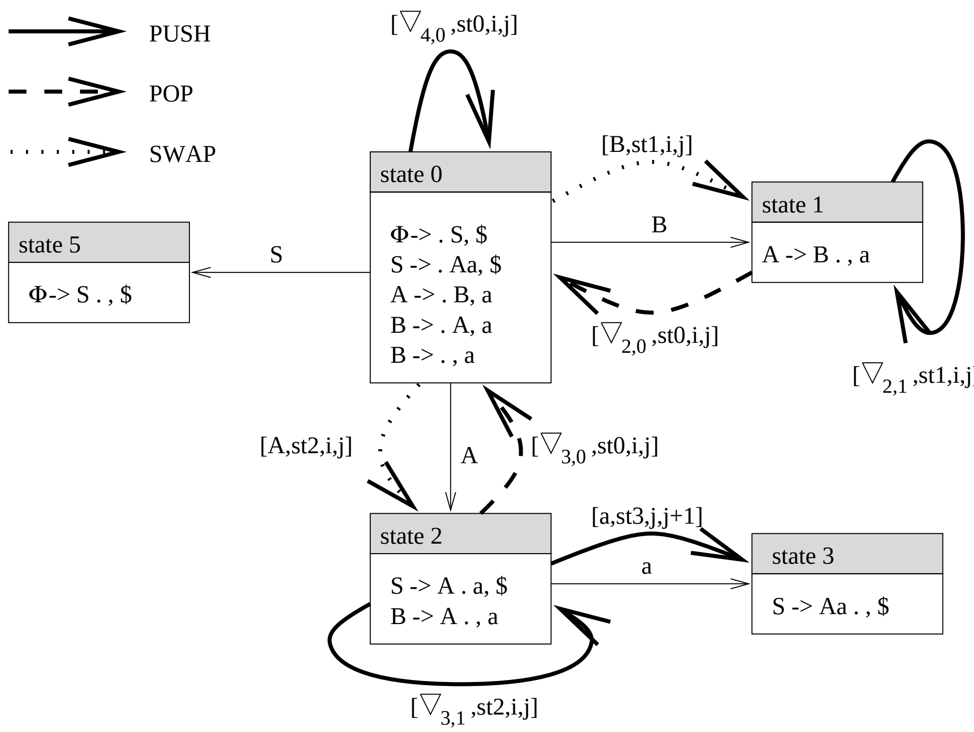

Figura C.7: Transiciones de un ciclo en el autómata LALR(1) de la gramática $\mathcal{G}_2$

Ejemplo C.2 Para mostrar el análisis de gramáticas cíclicas y con producciones recurrentes, utilizaremos la gramática $\mathcal{G}_2$, una vez más simple pero didáctica:

$$\begin{array}{ll} (0) & \Phi \rightarrow S \\ (1) & S \rightarrow Aa \\ (2) & A \rightarrow B \\ (3) & B \rightarrow A \\ (4) & B \rightarrow \varepsilon \end{array}$$

El lenguaje generado por $\mathcal{G}_2$ es $\{a\}$. En la figura C.7 podemos ver el autómata LALR(1) para esta gramática junto con las transiciones correspondientes a varios análisis cíclicos.

El análisis comienza en el estado 0. La primera acción a realizar es la reducción de la producción 4:

1. Sel: PUSH $[\nabla_{4,0}, st0, i, j]$

2. Head: SWAP $[\nabla_{4,0}, st0, i, j]$ para dar $[B, st1, i, j]$

El estado actual es el 1 y la siguiente acción a realizar la reducción de la producción 2:

1. Sel: PUSH $[\nabla_{2,1}, st1, i, j]$

2. Red: POP $[\nabla_{2,1}, st1, i, j]$ y $[B, st1, i, j]$ para dar $[\nabla_{2,0}, st0, i, j]$

3. Head: SWAP $[\nabla_{2,0}, st0, i, j]$ para dar $[A, st2, i, j]$

En este punto, el estado actual del autómata LALR(1) es el 2 y la acción a realizar es la reducción de la producción 3:

1. Sel: PUSH $[\nabla_{3,1}, st2, i, j]$

2. Red: POP $[\nabla_{3,1}, st2, i, j]$ y $[A, st2, i, j]$ para dar $[\nabla_{3,0}, st0, i, j]$

3. Head: SWAP $[\nabla_{3,0}, st0, i, j]$ para dar $[B, st1, i, j]$

El ítem resultante de la última transición, $[B, st1, i, j]$, ya había sido generado anteriormente y en consecuencia no es preciso volver a realizar las acciones derivadas a partir de él. Como resultado, todos los ítems generados por la aplicación de las producciones 2 y 3 con calculados una sola vez, en la primera iteración.

El análisis de la cadena de entrada continúa mediante la aplicación de un paso Shift en el estado 2, apilando el ítem $[a, st3, j, j+1]$. ¶

C.7 Análisis sintáctico LR Generalizado para Gramáticas de Cláusulas Definidas

La Gramáticas de Cláusulas Definidas (DCG) [143] son una extensión de las gramáticas independientes del contexto en las cuales se asocia un conjunto de atributos a los símbolos gramaticales, de tal modo que en vez de un conjunto de producciones sobre elementos de $(V_T \cup V_N)^*$ tenemos un conjunto de cláusulas sobre átomos lógicos. Frente a los enfoques que tratan de implantar analizadores LR para DCG mediante alteraciones en la estrategia de búsqueda de lenguajes lógicos [134, 196] o mediante transformaciones de la gramática [162], presentamos aquí un enfoque basado en la extensión de un algoritmo LR generalizado para gramáticas independientes del contexto.

Formalmente, una DCG está definida por la tupla $(V_N, V_T, \mathcal{P}, S, \mathcal{V}, F)$, donde $\mathcal{V}$ es un conjunto finito de variables, F es un conjunto finito de funtores y $\mathcal{P}$ es un conjunto de cláusulas definidas de la forma

$$A_{r,0}(t^1_{r,0}, \ldots, t^{m_0}_{r,0}) \to A_{r,1}(t^1_{r,1}, \ldots, t^{m_1}_{r,1}) \ \ldots \ A_{r,n_r}(t^1_{r,n_r}, \ldots, t^{m_{n_r}}_{r,n_r})$$

donde $A_{r,0} \in V_N$, $A_{r,i} \in (V_N \cup V_T)^*$ para $1 \geq i \geq n_r$ y $t^{m_j}_{r,i}$ son términos definidos inductivamente como sigue: un término es bien un funtor constante de aridad 0, bien una variable o bien un término compuesto $f(t_1, \ldots, t_l)$, donde f es un funtor de aridad l y $t_1, \ldots, t_l$ son términos. Si $A_{r,i} \in V_T$, los términos $t^1_{r,i}, \ldots, t^{m_i}_{r,i}$ se consideran relacionados con la entrada léxica $A_{r,i}$, pudiendo los valores variables ser obtenidos a partir de los valores almacenados en dicha entrada. Para cada cláusula γ_r definimos el vector $\overrightarrow{T_r}$ que contiene todas las variables que aparecen en γ_r. Cuando la mención explícita de los términos lógicos no sea precisa, denotaremos una cláusula o sus elementos mediante el símbolo de la gramática independiente del contexto que le corresponde escrito en negrita. Así, la cláusula anterior se escribirá en forma abreviada como $\mathbf{A_{r,0}} \to \mathbf{A_{r,1}} \ldots \mathbf{A_{r,n_r}}$.

Una diferencia fundamental desde el punto de visto operativo entre las gramáticas independientes del contexto y las gramáticas de cláusulas definidas es que mientras las primeras poseen un número finito de símbolos gramaticales, en las últimas el número de elementos gramaticales es potencialmente infinito ya que no existe un límite para el tamaño de los términos lógicos. En consecuencia, no se puede garantizar la terminación de las operaciones para la construcción de las tablas de análisis LR en el caso de DCG [120].

Una posible solución a este problema pasa por el uso de restrictores positivos [185] o negativos [202] con el fin de definir un número finito de clases de equivalencias en las cuales poder ordenar el número infinito de no-terminales[6]. Dichos restrictores deben aplicarse en todas aquellas

[6] Aunque los restrictores fueron originalmente definidos para ser aplicados a estructuras de rasgos [40], en general pueden ser aplicados a todos aquellos formalismos basados en restricciones [186], incluyendo las gramáticas de cláusulas definidas.

operaciones involucradas en la obtención de información precompilada a partir de la gramática: funciones primero y siguiente, cerradura de los estados del autómata LR y construcción de las tablas de acciones e ir_a.

Generalmente existen varios restrictores para gramática de cláusulas definidas pero sin embargo no puede asegurarse la terminación de las operaciones mencionadas anteriormente para cada uno de ellos. Es más, no existe ningún método automático para la selección del mejor restrictor puesto que este depende de la cantidad de información gramatical que deba ser preservada. En la práctica, con el fin e conseguir un equilibrio entre la bonanza del restrictor y la garantía de terminación, durante la fase de compilación se suele considerar únicamente la gramática independiente del contexto subyacente [226].

En el proceso de binarización de las cláusulas definidas, debemos tener en cuenta la transmisión de la información almacenada en los términos lógicos, de tal modo que una cláusula definida $\mathbf{A_{r,0}} \rightarrow \mathbf{A_{r,1}} \ldots \mathbf{A_{r,n_r}}$ se transforma en el siguiente conjunto de $n_r + 1$ cláusulas:

$$
\begin{array}{l}
\mathbf{A_{r,0}} \rightarrow \nabla_{r,0}(\overrightarrow{T_r}) \\
\nabla_{r,0}(\overrightarrow{T_r}) \rightarrow \mathbf{A_{r,1}}\ \nabla_{r,1}(\overrightarrow{T_r}) \\
\vdots \\
\nabla_{r,n_r-1}(\overrightarrow{T_r}) \rightarrow \mathbf{A_{r,n_r}}\ \nabla_{r,n_r}(\overrightarrow{T_r}) \\
\nabla_{r,n_r}(\overrightarrow{T_r}) \rightarrow \varepsilon
\end{array}
$$

Como mecanismo operacional utilizaremos los *autómatas lógicos a pila*[56, 52], esencialmente autómatas a pila que almacenan en la pila predicados lógicos en vez de simples elementeos gramaticales. A continuación definimos el esquema de compilación.

Esquema de compilación C.9 El esquema de compilación para una gramática de cláusulas definidas $\mathcal{G}$ y una estrategia de análisis sintáctico LR(1) o LALR(1) queda definido por el siguiente conjunto de producciones y el elemento final $[\mathbf{S}, st_f]$.

[INIT]	$\$_0 \longmapsto \$_0[-, st_0]$	
[INITSHIFT]	$[\mathbf{A}, st] \overset{a}{\longmapsto} [\mathbf{A}, st]\ [\mathbf{A_{r,1}}, st']$	$\mathbf{A_{r,1}} = a(t^1_{r,1}, \ldots, t^{m_1}_{r,1})$ $\text{shift}_{st'} \in \text{acción}(st, a)$
[SHIFT]	$[\mathbf{A_{r,s}}, st] \overset{a}{\longmapsto} [\mathbf{A_{r,s}}, st]\ [\mathbf{A_{r,s+1}}, st']$	$\mathbf{A_{r,s+1}} = a(t^1_{r,1}, \ldots, t^{m_1}_{r,1})$ $\text{shift}_{st'} \in \text{acción}(st, a)$
[SEL]	$[\mathbf{A}, st] \longmapsto [\mathbf{A_{r,n_r}}, st]\ [\nabla_{r,n_r}(\overrightarrow{T_r}), st]$	$\text{reduce}_r \in \text{acción}(st, \text{lookahead})$
[RED]	$[\mathbf{A_{r,s}}, st]\ [\nabla_{r,s}(\overrightarrow{T_r}), st] \longmapsto [\nabla_{r,s-1}(\overrightarrow{T_r}), st']$	$st' \in \text{revela}(st)$
[HEAD]	$[\nabla_{r,0}(\overrightarrow{T_r}), st] \longmapsto [\mathbf{A_{r,0}}, st']$	$st' \in \text{ir\_a}(st, A_{r,0})$

§

Esquema de análisis sintáctico C.10 La interpretación en programación dinámica del autómata lógico a pila correspondiente a los algoritmos LR(1) y LALR(1) para una gramática de cláusulas definidas $\mathcal{G}$ y una cadena de entrada $a_1 \dots a_n$ queda definido por el el sistema de análisis $\mathbb{P}_{\mathrm{LR}^{\mathrm{S1}}(\mathrm{DCG})}$ que se muestra a continuación:

$$\mathcal{I}_{\mathrm{LR}^{\mathrm{S1}}(\mathrm{DCG})} = \left\{ \ [\mathbf{A}, st, i, j] \cup [\nabla_{r,s}(\overrightarrow{T_r}), st, i, j] \ \right\}$$

$$\mathcal{D}^{\mathrm{Init}}_{\mathrm{LR}^{\mathrm{S1}}(\mathrm{DCG})} = \{\vdash [-, st_0, 0, 0]\}$$

$$\mathcal{D}^{\mathrm{InitShift}}_{\mathrm{LR}^{\mathrm{S1}}(\mathrm{DCG})} = \left\{ \begin{array}{l} [\mathbf{A}, st, i, j] \vdash [\mathbf{A_{r,1}}, st', j, j+1] \mid \\ \exists [a, j, j+1] \in \mathcal{H}_{\mathrm{LR}^{\mathrm{S1}}(\mathrm{DCG})}, \ \mathbf{A_{r,1}} = a(t^1_{r,1}, \dots, t^{m_1}_{r,1}), \ \mathrm{shift}_{st'} \in \mathrm{acción}(st, a) \end{array} \right\}$$

$$\mathcal{D}^{\mathrm{Shift}}_{\mathrm{LR}^{\mathrm{S1}}(\mathrm{DCG})} = \left\{ \begin{array}{l} [\mathbf{A_{r,s}}, st, i, j] \vdash [\mathbf{A_{r,s+1}}, st', i, j+1] \mid \\ \exists [a, j, j+1] \in \mathcal{H}_{\mathrm{LR}^{\mathrm{S1}}(\mathrm{DCG})}, \ \mathbf{A_{r,s+1}} = a(t^1_{r,s+1}, \dots, t^{m_{s+1}}_{r,s+1}), \\ \mathrm{shift}_{st'} \in \mathrm{acción}(st, a) \end{array} \right\}$$

$$\mathcal{D}^{\mathrm{Sel}}_{\mathrm{LR}^{\mathrm{S1}}(\mathrm{DCG})} = \left\{ \begin{array}{l} [\mathbf{A_{r,n_r}}, st, i, j] \vdash [\nabla_{r,n_r}(\overrightarrow{T_r}), st, j, j] \mid \\ \exists [a, j, j+1] \in \mathcal{H}_{\mathrm{LR}^{\mathrm{S1}}(\mathrm{DCG})}, \ \mathrm{reduce}_r \in \mathrm{acción}(st, a) \end{array} \right\}$$

$$\mathcal{D}^{\mathrm{Red}}_{\mathrm{LR}^{\mathrm{S1}}(\mathrm{DCG})} = \left\{ \ [\mathbf{A_{r,s}}, st, i, k] \ \ [\nabla_{r,s}(\overrightarrow{T_r}), st, k, j] \ \ \vdash [\nabla_{r,s-1}(\overrightarrow{T_r}), st', i, j] \mid st' \in \mathrm{revela}(st) \ \right\}$$

$$\mathcal{D}^{\mathrm{Head}}_{\mathrm{LR}^{\mathrm{S1}}(\mathrm{DCG})} = \left\{ \ [\nabla_{r,0}(\overrightarrow{T_r}), st, i, j] \vdash [\mathbf{A_{r,0}}, st', i, j] \mid st' \in \mathrm{ir\_a}(st, A_{r,0}) \ \right\}$$

$$\mathcal{D}_{\mathrm{LR}^{\mathrm{S1}}(\mathrm{DCG})} = \mathcal{D}^{\mathrm{Init}}_{\mathrm{LR}^{\mathrm{S1}}(\mathrm{DCG})} \cup \mathcal{D}^{\mathrm{InitShift}}_{\mathrm{LR}^{\mathrm{S1}}(\mathrm{DCG})} \cup \mathcal{D}^{\mathrm{Shift}}_{\mathrm{LR}^{\mathrm{S1}}(\mathrm{DCG})} \cup \mathcal{D}^{\mathrm{Sel}}_{\mathrm{LR}^{\mathrm{S1}}(\mathrm{DCG})} \cup \mathcal{D}^{\mathrm{Red}}_{\mathrm{LR}^{\mathrm{S1}}(\mathrm{DCG})} \cup \mathcal{D}^{\mathrm{Head}}_{\mathrm{LR}^{\mathrm{S1}}(\mathrm{DCG})}$$

$$\mathcal{F}_{\mathrm{LR}^{\mathrm{S1}}(\mathrm{DCG})} = \left\{ \ [\mathbf{S}, st_f, 0, n] \ \right\}$$

§

C.8 Análisis sintáctico LR Generalizado para Gramáticas Lineales de Índices

Las técnicas creadas para el desarrollo de analizadores sintácticos de tipo LR generalizado para DCG pueden aplicarse al caso de las gramáticas lineales de índices, puesto que estas pueden verse como un caso particular de DCG en el que a cada no terminal de una CFG se le añade un único atributo en forma de pila y se restringe la posibilidad de copia de su contenido (véase la sección 8.1 para más detalles). A continuación mostramos cómo especializar las técnicas desarrolladas en la sección precedente para la construcción de analizadores sintácticos LR para gramáticas lineales de índices.

El primer paso para realizar un analizador de tipo LR consiste en construir el autómata LR, un autómata finito en el cual se almacena cierta información obtenida mediante un análisis estático de la gramática. Al igual que en el caso de las gramáticas de cláusulas definidas, existen dos opciones técnicas para construir dicho autómata:

1. Considerar únicamente el esqueleto independiente del contexto de la gramática lineal de índices.
2. Incluir información relativa a la evolución de las pilas de índices en el autómata LR.

La primera opción es más simple pero menos efectiva puesto que no se considera toda la información disponible en la gramática durante la construcción del autómata LR y por consiguiente habrá más conflictos en la fase de ejecución del algoritmo de análisis LR. Estos conflictos se hubiesen podido evitar de haber considerado la información acerca de la constitución de las pilas durante la construcción del autómata LR. Para ello sería necesario sustituir las funciones *primero* y *siguiente* [6] por las que se muestran a continuación.

Definición C.3 *Un elemento* $a \in V_T \cup \{\epsilon\}$ *pertenece a* $\text{primero}(\Gamma)$*, donde* $\Gamma \in V_T \cup V_N[V_I{}^*]$*, si se cumple alguna de las condiciones siguientes:*

- $\Gamma = a$
- $\Gamma' \rightarrow \epsilon \in P$*, existe* $\sigma = \text{mgu}(\Gamma, \Gamma')$ *y* $a = \epsilon$
- $\Gamma' \rightarrow \Gamma_1 \ldots \Gamma_i \ldots \Gamma_n \in P$*, existe* $\sigma = \text{mgu}(\Gamma, \Gamma')$ *y* $a \in \text{primero}(\Gamma_i \sigma)$ *y* $\forall_{j=1}^{i-1} \epsilon \in \text{primero}(\Gamma_j \sigma)$

donde mgu se refiere al unificador más general.

A partir de la definición anterior obtenemos el siguiente método de cálculo de la función $\text{primero}(\Gamma)$:

1. Si $\Gamma = a \in V_T$ entonces $\text{primero}(a) = \{a\}$.
2. Si $\Gamma \rightarrow \varepsilon$ entonces $\varepsilon \in \text{primero}(\Gamma)$.
3. Si $\Gamma' \rightarrow \Gamma_1 \ldots \Gamma_i \ldots \Gamma_n$ y existe $\sigma = \text{mgu}(\Gamma, \Gamma')$, entonces $\text{primero}(\Gamma_1 \sigma) \subseteq \text{primero}(\Gamma)$ y $\forall_{j=1}^{j=i-1} \Gamma_j$ tal que $\varepsilon \in \text{primero}(\Gamma_j)$ tenemos que $\text{primero}(\Gamma_{j+1} \sigma) \subseteq \text{primero}(\Gamma)$

Definición C.4 *Un elemento* $a \in V_T \cup \{\$\}$ *pertenece a* $\text{siguiente}(\Gamma)$*, donde* $\Gamma \in V_T \cup V_N[V_I{}^*]$ *y \$ es un carácter especial que no pertenece a* V_T *que indica que se ha alcanzado el final de la cadena de entrada, si se cumple alguna de las siguientes condiciones:*

- $a = \$$ *y* $\Gamma = S[\,]$
- $\Gamma'' \rightarrow \Upsilon_1 \Gamma' \Upsilon_2$*, existe* $\sigma = \text{mgu}(\Gamma, \Gamma')$ *y* $a \in \text{primero}(\Upsilon_2 \sigma) - \{\epsilon\}$
- $\Gamma'' \rightarrow \Upsilon_1 \Gamma' \Upsilon_2$*, existe* $\sigma = \text{mgu}(\Gamma, \Gamma')$ *y* $\epsilon \in \text{primero}(\Upsilon_2 \sigma)$ *y* $a \in \text{siguiente}(\Gamma'' \sigma)$

A partir de la definición anterior obtenemos el siguiente método de cálculo de la función $\text{siguiente}(\Gamma)$:

1. Si $\Gamma = S[\,]$ entonces $\$ \in \text{siguiente}(\Gamma)$.
2. Si $\Gamma'' \rightarrow \Upsilon_1 \Gamma' \Upsilon_2$ y $\sigma = mgu(\Gamma, \Gamma')$ entonces $\text{primero}(\Upsilon_2 \sigma) \subseteq \text{siguiente}(\Gamma)$.
3. Si $\Gamma'' \rightarrow \Upsilon_1 \Gamma' \Upsilon_2$ y $\sigma = mgu(\Gamma, \Gamma')$ y $\varepsilon \in \text{primero}(\Upsilon_2 \sigma)$ entonces $\text{siguiente}(\Gamma'' \sigma) \subseteq \text{siguiente}(\Gamma)$.

El cierre de los estados del autómata se efectúa como en el algoritmo clásico [6] con la salvedad de que las funciones *primero* y *siguiente* son reemplazadas por las que acabamos de definir. Puesto que la operación de cierre es esencialmente predictiva pueden surgir problemas de no terminación si el proceso de unificación introduce un número infinito de nuevos elementos a considerar. Este es un problema bien conocido en el ámbito de la programación lógica y para solucionarlo se han propuesto varias alternativas. De entre todas ellas, la mejor adaptada a las características de LIG consiste en aplicar la noción de *restrictor* propuesta por Shieber en [185]. En efecto, puesto que únicamente es necesario considerar un número acotado de elementos de la cima de cada pila de índices para determinar si una producción puede ser o no utilizada en una derivación, podemos restringir el alcance de la unificación a los primeros elementos de la pila de índices y considerar el resto como una variable lógica. En la tabla C.1 se define la operación de unificación para el caso de que sólo se consulte el primer elemento de la cima de las pilas LIG. Para todos aquellos casos no mostrados en el tabla, la unificación fracasa. En la tabla C.2 se define la operación de subsumción para el mismo caso.

$\text{mgu}(A[\,], A[\,]) = A[\,]$
$\text{mgu}(A[\,], A[\circ\circ]) = A[\,]$
$\text{mgu}(A[\circ\circ], A[\,]) = A[\,]$
$\text{mgu}(A[\circ\circ], A[\circ\circ]) = A[\circ\circ]$
$\text{mgu}(A[\circ\circ], A[\circ\circ\gamma]) = A[\circ\circ\gamma]$
$\text{mgu}(A[\circ\circ\gamma], A[\circ\circ]) = A[\circ\circ\gamma]$
$\text{mgu}(A[\circ\circ\gamma_1], A[\circ\circ\gamma_2]) = A[\circ\circ\gamma_1]$ si y sólo si $\gamma_1 = \gamma_2$

Tabla C.1: Unificación mediantes restrictores de símbolos LIG

$A[\,] \preceq A[\,]$
$A[\gamma] \preceq A[\gamma]$
$A[\circ\circ] \preceq A[\,]$
$A[\circ\circ] \preceq A[\circ\circ]$
$A[\circ\circ] \preceq A[\circ\circ\gamma]$
$A[\circ\circ\gamma] \preceq A[\gamma]$
$A[\circ\circ\gamma] \preceq A[\circ\circ\gamma]$
$A[\circ\circ\gamma] \preceq A[\circ\circ\gamma'\gamma]$

Tabla C.2: Subsumción mediante restrictores de símbolos LIG

Mediante la utilización de esta técnica sólo se pueden generar un número finito de elementos durante el proceso de construcción de los estados del autómata LR. Una vez que el autómata ha sido construido, se construyen las tablas de *acción* e *ir_a* como para los algoritmos LR clásicos, teniendo en cuenta que las transiciones entre estados están etiquetadas no ya por un símbolo terminal o no-terminal sino por un terminal o por un elemento del conjunto

$V_N[\,] \cup V_N[\circ\circ] \cup V_N[\circ\circ V_I]$. Dependiendo del tratamiento del símbolo de preanálisis obtendremos un autómata finito LR(1) o LALR(1).

A continuación se detalla el esquema de compilación que dada una gramática lineal de índices genera el conjunto de transiciones del autómata lógico a pila que aplica el autómata finito LR previamente construido junto con un mecanismo de desplazamiento y reducción para analizar sintácticamente una cadena de entrada de acuerdo con la gramática. Los elementos de la pila del autómata son pares $\langle A, st\rangle$, donde A es un elemento de la gramática y st es un estado del autómata LR. En este esquema se ha aplicado una binarización implícita de las producciones de la gramática de tal modo que una producción

$$A_{r,0}[\circ\circ\gamma] \to A_{r,1}[\,] \ldots A_{r,l}[\circ\circ\gamma'] \ldots A_{r,n_r}[\,]$$

se ha descompuesto en las siguientes $n_r + 1$ producciones

$$\begin{array}{l}
A_{r,0}[\circ\circ\gamma] \to \nabla_{r,0}[\circ\circ\gamma] \\
\nabla_{r,0}[\circ\circ\gamma] \to A_{r,1}[\,]\ \nabla_{r,1}[\circ\circ\gamma] \\
\vdots \\
\nabla_{r,l-1}[\circ\circ\gamma] \to A_{r,l}[\circ\circ\gamma']\ \nabla_{r,l}[\,] \\
\nabla_{r,l}[\circ\circ] \to A_{r,l+1}[\,]\ \nabla_{r,l+1}[\circ\circ] \\
\vdots \\
\nabla_{r,n_r-1}[\circ\circ] \to A_{r,n_r}[\,]\ \nabla_{r,n_r}[\circ\circ] \\
\nabla_{r,n_r}[\,] \to \epsilon
\end{array}$$

donde la pila de índices asociada los $\nabla_{r,i}$ con $i \in [l \ldots n_r]$ está vacía al ser heredada de $\nabla_{r,n_r}[\,]$, puesto que el algoritmo LR reduce las producciones de derecha a izquierda.

Esquema de compilación C.11 El esquema de compilación para una gramática lineal de índices $\mathcal{G}$ y una estrategia de análisis sintáctico LR(1) o LALR(1) queda definido por el siguiente conjunto de reglas de compilación y el elemento final $\langle S, st_f\rangle$.

$$\begin{array}{lll}
\textbf{[INIT]} & \langle \$_0[\circ\circ], -\rangle \longmapsto \langle \$_0[\circ\circ], -\rangle\ \langle -, st_0\rangle & \\
\textbf{[SHIFT]} & \langle A[\circ\circ], st\rangle \stackrel{a}{\longmapsto} \langle A[\circ\circ], st\rangle\ \langle A_{r,1}[\,], st'\rangle & A_{r,1} = a,\ \text{shift}_{st'} \in \text{acción}(st, a) \\
\textbf{[SEL]} & \langle A_{r,n_r}[\circ\circ], st\rangle \longmapsto \langle A_{r,n_r}[\circ\circ], st\rangle\ \langle \nabla_{r,n_r}[\,], st\rangle & \text{reduce}_r \in \text{acción}(st, \text{lookahead}) \\
\textbf{[RED]} & \langle A_{r,s}[\,], st\rangle\langle \nabla_{r,s}[\circ\circ], st\rangle\ \longmapsto \langle \nabla_{r,s-1}[\circ\circ], st'\rangle & st \in \text{ir_a}(st', A_{r,s}) \\
\textbf{[SRED]} & \langle A_{r,s}[\circ\circ\gamma], st\rangle\langle \nabla_{r,s}[\,], st\rangle\ \longmapsto \langle \nabla_{r,s-1}[\circ\circ\gamma'], st'\rangle & st \in \text{ir_a}(st', A_{r,s}) \\
\textbf{[HEAD]} & \langle \nabla_{r,0}[\circ\circ], st\rangle \longmapsto \langle A_{r,0}[\circ\circ], st'\rangle & st' \in \text{ir_a}(st, A_{r,0})
\end{array}$$

donde, si $A_{r,s}$ es el hijo dependiente de la producción r se aplica **[SRED]** en el proceso de reducción, en otro caso se aplica **[RED]**. El estado inicial del autómata LR es st_0 y el final es st_f. §

Las transiciones del esquema de compilación C.11 son un subconjunto de las transiciones de los autómatas lógicos a pila restringidos al caso de LIG que incorporan estrategias $*$-ascendentes [14]. Puesto que la estrategia LR es completamente ascendente al no realizar la fase de llamada ninguna predicción sobre la parte independiente del contexto, podemos utilizar la técnica de tabulación para la estrategia ascendente-ascendente propuesta en la sección 8.5.2, obteniendo la interpretación tabular del autómata que se muestra en el esquema de compilación C.12. La complejidad temporal con respecto a la longitud de la cadena de entrada es $\mathcal{O}(n^6)$ y la complejidad espacial es $\mathcal{O}(n^4)$.

Esquema de análisis sintáctico C.12 La interpretación en programación dinámica del autómata lógico a pila correspondiente a los algoritmos LR(1) y LALR(1) para una gramática lineal de índices $\mathcal{G}$ y una cadena de entrada $a_1 \ldots a_n$ queda definido por el el sistema de análisis $\mathbb{P}_{\mathrm{LR}^{S1}(\mathrm{LIG})}$ que se muestra a continuación:

$$\mathcal{I}_{\mathrm{LR}} = \left\{ \begin{array}{l} [i, A, st_1, j, \gamma_1 \mid p, B, st_2, q] \mid \\ A, B \in V_N,\ \gamma \in V_I,\ st_1, st_2 \in \mathcal{A},\ 0 \leq i \leq j,\ (p, q) \leq (i, j) \end{array} \right\}$$

$$\mathcal{D}_{\mathrm{LR}}^{\mathrm{Init}} = \frac{}{[0, -, st_0, 0, - \mid -, -, -, -]}$$

$$\mathcal{D}_{\mathrm{LR}}^{\mathrm{Shift}} = \frac{[i, A, st_1, j, \gamma \mid p, B, st_2, q]}{[j, A_{r,1}, st_3, j+1, - \mid -, -, -, -]} \quad \begin{array}{l} \exists [a, j, j+1] \in \mathcal{H}_{\mathrm{LR}},\ A_{r,1} = a, \\ \mathrm{shift}_{st_3} \in \mathrm{acción}(st_1, a),\ A \in V \end{array}$$

$$\mathcal{D}_{\mathrm{LR}}^{\mathrm{Sel}} = \frac{[i, A_{r,n_r}, st_1, j, \gamma \mid p, B, st_2, q]}{[j, \nabla_{r,n_r}, st_1, j, - \mid -, -, -, -]} \quad \begin{array}{l} \exists [a, j, j+1] \in \mathcal{H}_{\mathrm{LR}}, \\ \mathrm{reduce}_r \in \mathrm{acción}(st_1, a_j), \end{array}$$

$$\mathcal{D}_{\mathrm{LR}}^{\mathrm{Red}} = \frac{\begin{array}{c} [k, \nabla_{r,s}, st_1, k, j, \gamma \mid p, B, st_2, q], \\ [A_{r,s}, st_1, i, k, - \mid -, -, -, -] \end{array}}{[i, \nabla_{r,s-1}, st_3, j, \gamma \mid p, B, st_2, q]} \quad \begin{array}{l} \boldsymbol{A}_{r,0} \to \Upsilon_1\ A_{r,s}[\,]\ \Upsilon_2, \\ st_3 \in \mathrm{revela}(st_1) \end{array}$$

$$\mathcal{D}_{\mathrm{LR}}^{\mathrm{SRed}_1} = \frac{\begin{array}{c} [k, \nabla_{r,s}, st_1, j, -, \mid -, -, -, -], \\ [i, A_{r,s}, st_1, k, \gamma \mid p, B, st_2, q] \end{array}}{[i, \nabla_{r,s-1}, st_3, j, \gamma \mid p, B, st_2, q]} \quad \begin{array}{l} A_{r,0}[\circ\circ] \to \Upsilon_1\ A_{r,s}[\circ\circ]\ \Upsilon_2, \\ st_1 \in \mathrm{ir\_a}(st_3, A_{r,s}) \end{array}$$

$$\mathcal{D}_{\mathrm{LR}}^{\mathrm{SRed}_2} = \frac{\begin{array}{c} [k, \nabla_{r,s}, st_1, j, -, \mid -, -, -, -], \\ [i, A_{r,s}, st_1, k, \gamma' \mid p, B, st_2, q] \end{array}}{[i, \nabla_{r,s-1}, st_3, j, \gamma \mid i, A_{r,s}, st_1, k]} \quad \begin{array}{l} A_{r,0}[\circ\circ\gamma] \to \Upsilon_1\ A_{r,s}[\circ\circ]\ \Upsilon_2, \\ st_1 \in \mathrm{ir\_a}(st_3, A_{r,s}) \end{array}$$

$$\mathcal{D}_{\mathrm{LR}}^{\mathrm{Red}_2} = \frac{\begin{array}{c} [k, \nabla_{r,s}, st_1, j, - \mid -, -, -, -], \\ [i, A_{r,s}, st_1, k, \gamma' \mid p, B, st_2, q], \\ [p, B, st_2, q, \gamma'' \mid p', C, st_3, q'] \end{array}}{[i, \nabla_{r,s-1}, st_4, j, \gamma'' \mid p', C, st_3, q']} \quad \begin{array}{l} A_{r,0}[\circ\circ] \to \Upsilon_1\ A_{r,s}[\circ\circ\gamma']\ \Upsilon_2, \\ st_4 \in \mathrm{revela}(st_1) \end{array}$$

$$\mathcal{D}_{\mathrm{LR}}^{\mathrm{Head}} = \frac{[i, \nabla_{r,0}, st_1, j, \gamma, \mid p, B, st_2, q]}{[i, A_{r,0}, st_3, j, \gamma \mid p, B, st_2, q]} \quad st_1 \in \mathrm{revela}(st_3)$$

$$\mathcal{D}_{\mathrm{LR}} = \mathcal{D}_{\mathrm{LR}}^{\mathrm{Init}} \cup \mathcal{D}_{\mathrm{LR}}^{\mathrm{Shift}} \cup \mathcal{D}_{\mathrm{LR}}^{\mathrm{Sel}} \cup \mathcal{D}_{\mathrm{LR}}^{\mathrm{Red}} \cup \mathcal{D}_{\mathrm{LR}}^{\mathrm{SRed}_1} \cup \mathcal{D}_{\mathrm{LR}}^{\mathrm{SRed}_2} \cup \mathcal{D}_{\mathrm{LR}}^{\mathrm{SRed}_3} \cup \mathcal{D}_{\mathrm{LR}}^{\mathrm{Head}}$$

$$\mathcal{F}_{\mathrm{LR}} = \{\ [0, S, st_f, -, n \mid -, -, -, -]\ \}$$

donde $\mathcal{A}$ es el conjunto de estados del autómata LR, con st_0 su estado inicial y st_f su estado final. §

C.9 Conclusiones

Hemos mostrado como el algoritmo de Earley es un punto de arranque adecuado para el diseño de otros algoritmos de análisis sintáctico más complejos. En esta dirección, hemos derivado en sucesivas etapas un analizador LR generalizado capaz de analizar gramáticas independientes del contexto sin restricciones. Cada algoritmo intermedio ha sido descrito utilizando un esquema de análisis y hemos pasado de uno a otro aplicando transformaciones que podemos calificar de sencillas. Como resultado hemos obtenido un algoritmo en programación dinámica que se integra perfectamente en la estructura general de análisis propuesta por Lang, mostrando una complejidad con respecto a la cadena de entrada del orden de $\mathcal{O}(n^3)$ en el peor caso, aunque puede llegar a ser lineal en ciertos casos, tal y como sucede en el algoritmo de Earley.

La técnica propuesta ha sido también aplicada a formalismos gramaticales que aun no siendo independientes del contexto poseen un esqueleto independiente del contexto. Es el caso de las gramáticas de cláusulas definidas y de las gramáticas lineales de índices, ya que se ha mostrado que existe una extensión directa de las técnicas de análisis independientes del contexto que permite la evaluación de cláusulas de Horn [52]. Vilares Ferro y Alonso Pardo describen en [222, 221] la implementación de un analizador de gramáticas de cláusulas definidas basada en la especificación del algoritmo LALR(1) generalizado mostrada en este capítulo. Vilares Ferro et al. muestran en [223] una extensión de dicho algoritmo que permite representar de forma finita ciertas clases de términos lógicos infinitos, ampliando de esta manera el dominio de aplicación.

En los analizadores de lenguajes naturales resulta de gran interés disponer de la posibilidad de revisar dinámicamente la corrección sintáctica de un texto cuando ha sido editado sin necesidad de rehacer totalmente el análisis sintáctico sino cambiando únicamente las partes de las estructuras de análisis que se vean afectadas por el cambio. En este contexto, el algoritmo presentado aquí puede ser extendido con el fin de obtener un analizador sintáctico totalmente incremental, esto es, un analizador sintáctico que recupera partes del análisis anterior siempre que se realiza un nuevo análisis. Mediante el concepto de totalmente incremental se quiere indicar que se permiten modificaciones en cualquier punto de la cadena de entrada [232]. En [219, 227, 220] se describe un analizador sintáctico incremental basado en nuestra especificación del algoritmo LALR(1) para gramáticas independientes del contexto mientras que en [7] se analiza la integración de dicho analizador incremental en el entorno Galena [224] de desarrollo de herramientas para lenguaje natural, incluyendo la interacción con los etiquetadores diseñados en dicho enotrno [76].

Finalmente, señalar que en ciertos casos es posible aumentar la eficiencia del algoritmo obtenido aplicando técnicas para la compartición de ítems que se corresponden con el análisis de un mismo sufijo común a varias producciones gramaticales [122, 121]. Nederhof y Satta han estudiado esta vía en [130] para el caso de analizadores tabulares LR. El algoritmo resultante es tan complejo, ya que además del mecanismo de tabulación requiere la realización de un transformación a la gramática y de la aplicación de una función de filtrado, que el valor práctico de tales optimizaciones depende en gran medida de las características propias de las gramáticas utilizadas.

Bibliografía

[1] Anne Abeillé and Marie-Hélène Candito. FTAG: a lexicalized tree adjoining grammar for French. In Anne Abeillé and Owen Rambow, editors, *Tree Adjoining Grammars : Formal Properties, Linguistic Theory and Applications*. CSLI, Stanford, CA, USA, 2000.

[2] Anne Abeillé, Marie-Hélène Candito, and Alexandra Kinyon. The current status of FTAG. In *Proc. of 5th International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+5)*, pages 11–18, Paris, France, May 2000.

[3] Alexandre Agustini and Vera Lúcia Strube de Lima. An experiment on synchronous TAGs for the construction of a transfer module. In *Proc. of Fourth International Workshop on Tree-Adjoining Grammars and Related Frameworks (TAG+4)*, pages 1–4, Philadelphia, PA, USA, August 1998.

[4] Alfred V. Aho. Indexed grammars — an extension of context-free grammars. *Journal of the Association for Computer Machinery*, 15(4):647–671, October 1968.

[5] Alfred V. Aho, Ravi Sethi, and Jeffrey D. Ullman. *Compilers: Principles, Techniques and Tools*. Addison-Wesley Publishing Company, Inc., Reading, Massachusetts, USA, 1986.

[6] Alfred V. Aho and Jeffrey D. Ullman. *The Theory of Parsing, Translation and Compiling*. Prentice Hall, Englewood Cliffs, New Jersey, USA, 1972.

[7] Miguel A. Alonso Pardo. Edición interactiva en entornos incrementales. Master's thesis, Facultade de Informática, Universidade da Coruña, Corunna, Spain, October 1994.

[8] Miguel A. Alonso Pardo, David Cabrero Souto, Eric de la Clergerie, and Manuel Vilares Ferro. Tabular algorithms for TAG parsing. In *Proc. of EACL'99, Ninth Conference of the European Chapter of the Association for Computational Linguistics*, pages 150–157, Bergen, Norway, June 1999. ACL.

[9] Miguel A. Alonso Pardo, David Cabrero Souto, Eric de la Clergerie, and Manuel Vilares Ferro. Algoritmos tabulares para el análisis de TAG. *Procesamiento del Languaje Natural*, 23:157–164, September 1998.

[10] Miguel A. Alonso Pardo, David Cabrero Souto, and Manuel Vilares Ferro. Construction of efficient generalized LR parsers. In *Proc. of Second International Workshop on Implementing Automata (WIA'97)*, pages 131–140, London, Ontario, Canada, September 1997.

[11] Miguel A. Alonso Pardo, David Cabrero Souto, and Manuel Vilares Ferro. A new approach to the construction of Generalized LR parsing algorithms. In Ruslan Mitkov, Nicolas Nicolov, and Nikolai Nikolov, editors, *Proc. of Recent Advances in Natural Language Processing (RANLP'97)*, pages 171–178, Tzigov Chark, Bulgaria, September 1997.

[12] Miguel A. Alonso Pardo, David Cabrero Souto, and Manuel Vilares Ferro. Construction of efficient generalized LR parsers. In Derick Wood and Sheng Yu, editors, *Automata Implementation*, volume 1436 of *Lecture Notes in Computer Science*, pages 7–24. Springer-Verlag, Berlin-Heidelberg-New York, 1998.

[13] Miguel A. Alonso Pardo, David Cabrero Souto, and Manuel Vilares Ferro. Generalized LR parsing for extensions of context-free grammars. In Nicolas Nicolov and Ruslan Mitkov, editors, *Recent Advances in Natural Language Processing II*, volume 189 of *Current Issues in Linguistic Theory*. John Benjamins Publishing Company, Amsterdam & Philadelphia, 1999.

[14] Miguel A. Alonso Pardo, Eric de la Clergerie, and David Cabrero Souto. Tabulation of automata for tree adjoining languages. In *Proc. of the Sixth Meeting on Mathematics of Language (MOL 6)*, pages 127–141, Orlando, Florida, USA, July 1999.

[15] Miguel A. Alonso Pardo, Eric de la Clergerie, Jorge Graña Gil, and Manuel Vilares Ferro. New tabular algorithms for LIG parsing. In *Proc. of the Sixth International Workshop on Parsing Technologies (IWPT 2000)*, pages 29–40, Trento, Italy, February 2000.

[16] Miguel A. Alonso Pardo, Eric de la Clergerie, and Manuel Vilares Ferro. Automata-based parsing in dynamic programming for Linear Indexed Grammars. In A. S. Narin'yani, editor, *Proc. of DIALOGUE'97 Computational Linguistics and its Applications International Workshop*, pages 22–27, Moscow, Russia, June 1997.

[17] Miguel A. Alonso Pardo, Eric de la Clergerie, and Manuel Vilares Ferro. A formal definition of Bottom-up Embedded Push-Down Automata and their tabulation technique. In *Proc. of Second Interbnational Workshop on Tabulation in Parsing and Deduction (TAPD 2000)*, Vigo, Spain, September 2000.

[18] Miguel A. Alonso Pardo, Eric de la Clergerie, and Manuel Vilares Ferro. A redefinition of Embedded Push-Down Automata. In *Proc. of 5th International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+5)*, pages 19–26, Paris, France, May 2000.

[19] Miguel A. Alonso Pardo, Jorge Graña Gil, and Manuel Vilares Ferro. Nuevos algoritmos tabulares para el análisis de LIG. *Procesamiento del Lenguaje Natural*, 25:7–14, September 1999.

[20] Miguel A. Alonso Pardo, Mark-Jan Nederhof, and Eric de la Clergerie. Tabulation of automata for tree adjoining languages. *Grammars*, Forthcoming.

[21] Miguel A. Alonso Pardo, Djamé Seddah, and Eric de la Clergerie. Practical aspects in compiling tabular TAG parsers. In *Proc. of 5th International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+5)*, pages 27–32, Paris, France, May 2000.

[22] Srinivas Bangalore. Transplanting supertags from English to Spanish. In *Proc. of Fourth International Workshop on Tree-Adjoining Grammars and Related Frameworks (TAG+4)*, pages 5–8, Philadelphia, PA, USA, August 1998.

[23] François Barthélemy. Un analyseur syntaxique pour les grammaires d'arbres adjoints. Rapport de stage de DEA, Université d'Orleans and INRIA, France, September 1989.

[24] François Barthélemy. *Outils pour l'analyse syntaxiqye contextuelle.* PhD thesis, Université d'Orléans, Orleans, France, February 1993.

[25] Tilman Becker. A new automaton model for TAGs: 2-SA. *Computational Intelligence*, 10(4):422–430, 1994.

[26] Tilman Becker and Dominik Heckmann. Recursive matrix systems (RMS) and TAG. In *Proc. of Fourth International Workshop on Tree Adjoining Grammars and Related Frameworks (TAG+4)*, pages 9–12, Philadelphia, PA, USA, August 1998.

[27] Tilman Becker and Dominik Heckmann. Parsing mildly context-sensitive RMS. In *Proc. of the Sixth International Workshop on Parsing Technologies (IWPT 2000)*, pages 293–294, Trento, Italy, February 2000.

[28] Tilman Becker, Owen Rambow, and Michael Niv. The derivational generative power of formal systems or scrambling is beyond LCFRS. Technical Report IRCS-92-38, Institute for Research in Cognitive Science, University of Pennsylvania, 1992.

[29] R.E. Bellman. *Dynamic Programming.* Princeton University Press, Princeton, New Jersey, 1957.

[30] Sylvie Billot and Bernard Lang. The structure of shared forest in ambiguous parsing. In *Proc. of the 27th Annual Meeting of the Association for Computational Linguistics*, pages 143–151, Vancouver, British Columbia, Canada, June 1989. ACL.

[31] Pierre Boullier. Yet another $\mathcal{O}(n^6)$ recognition algorithm for mildly context-sensitive languages. In *Proc. of the Fourth International Workshop on Parsing Technologies*, pages 34–47, 1995. Extended version as INRIA Rapport de Recherche 2730.

[32] Pierre Boullier. Another facet of LIG parsing. In *Proc. of 34th Annual Meeting of the Association for Computational Linguistics*, Santa Cruz, CA, USA, June 1996. ACL. Extended version as INRIA Rapport de Recherche 2858.

[33] Pierre Boullier. A generalization of mildly context-sensitive formalisms. In *Proc. of Fourth International Workshop on Tree Adjoining Grammars and Related Frameworks (TAG+4)*, pages 17–20, Philadelphia, PA, USA, August 1998.

[34] Pierre Boullier. Proposal for a natural language processing syntactic backbone. Rapport de recherche 3342, INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 Le Chesnay Cedex, France, January 1998.

[35] Pierre Boullier. Chinese numbers, MIX, scrambling, and range concatenation grammars. In *Proc. of EACL'99, Ninth Conference of the European Chapter of the Association for Computational Linguistics*, pages 53–60, Bergen, Norway, June 1999. ACL.

[36] Pierre Boullier. A cubic time extension of context-free grammars. In *Proc. of the Sixth Meeting on Mathematics of Language (MOL 6)*, pages 37–50, Orlando, Florida, USA, July 1999.

[37] Pierre Boullier. Range concatenation grammars. In *Proc. of the Sixth International Workshop on Parsing Technologies (IWPT 2000)*, pages 53–64, Trento, Italy, February 2000.

[38] Marie-Hélène Candito. Building parallel LTAG for French and Italian. In *COLING-ACL'98, 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Proceedings of the Conference*, volume I, pages 211–217, Montreal, Quebec, Canada, August 1998. ACL.

[39] Marie-Hélène Candito and Sylvain Kahane. Defining DTG derivations to get semantic graphs. In *Proc. of Fourth International Workshop on Tree-Adjoining Grammars and Related Frameworks (TAG+4)*, pages 25–28, Philadelphia, PA, USA, August 1998.

[40] Bob Carpenter. *The Logic of Typed Feature Structures.* Number 32 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, Cambridge/New York/Melbourne, 1992.

[41] Vicente Carrillo Montero, Víctor J. Díaz Madrigal, and A. Gómez Ojeda. Gramática FTAG del castellano: árboles elementales asociados a los adjetivos. *Procesamiento del Lenguaje Natural*, 17:130–141, September 1995.

[42] Vicente Carrillo Montero, Víctor J. Díaz Madrigal, and Miguel Toro Bonilla. Un método general de transformación de CFG a TAG. In C. Matín Vide, editor, *Lenguajes Naturales y Lenguajes Formales*, volume XII, pages 415–422, Barcelona, Spain, September 1996. PPU.

[43] John Carrol, Nicolas Nicolov, Olga Shaumyan, Martine Smets, and David Weir. Grammar compaction and computation sharing in automaton-based parsing. In *Proc. of First Workshop on Tabulation in Parsing and Deduction (TAPD'98)*, pages 16–25, Paris, France, April 1998.

[44] John Carrol, Nicolas Nicolov, Olga Shaumyan, Martine Smets, and David Weir. The LEXSYS project. In *Proc. of Fourth International Workshop on Tree-Adjoining Grammars and Related Frameworks (TAG+4)*, pages 29–33, Philadelphia, PA, USA, August 1998.

[45] John Carrol and David Weir. Encoding frequency information in lexicalized grammars. In *Proc. of the 5th International Workshop on Parsing Technologies (IWPT-97)*, Boston/Cambridge, MA, USA, 1997. ACL/SIGPARSE.

[46] Marc Cavazza. Synchronous TFG for speech translation. In *Proc. of Fourth International Workshop on Tree-Adjoining Grammars and Related Frameworks (TAG+4)*, pages 38–41, Philadelphia, PA, USA, August 1998.

[47] John Chen and K. Vijay-Shanker. Automated extraction of TAGs from the Penn treebank. In *Proc. of the Sixth International Workshop on Parsing Technologies (IWPT 2000)*, pages 65–76, Trento, Italy, February 2000.

[48] David Chiang, William Schuler, and Mark Dras. Some remarks on an extension of synchronous TAG. In *Proc. of 5th International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+5)*, pages 61–66, Paris, France, May 2000.

[49] Noam Chomsky. On certain formal properties of grammars. *Information and Control*, 2(2):137–167, 1959.

[50] Jürgen Dassow, Gheorghe Păun, and Arto Salomaa. Grammars with controlled derivations. In Grzegorz Rozenberg and Arto Salomaa, editors, *Handbook of Formal Languages. Vol 2: Linear Modelling: Background and Application*, chapter 3, pages 101–154. Springer-Verlag, Berlin/Heidelberg/New York, 1997.

[51] Yannick de Kercadio. An improved Earley parser with LTAG. In *Proc. of Fourth International Workshop on Tree Adjoining Grammars and Related Frameworks (TAG+4)*, pages 84–87, Philadelphia, PA, USA, August 1998.

[52] Eric de la Clergerie. *Automates à Piles et Programmation Dynamique. DyALog : Une Application à la Programmation en Logique.* PhD thesis, Université Paris 7, Paris, France, 1993.

[53] Eric de la Clergerie and Miguel A. Alonso Pardo. A tabular interpretation of a class of 2-Stack Automata. In *COLING-ACL'98, 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Proceedings of the Conference*, volume II, pages 1333–1339, Montreal, Quebec, Canada, August 1998. ACL.

[54] Eric de la Clergerie, Miguel A. Alonso Pardo, and David Cabrero Souto. A tabular interpretation of bottom-up automata for TAG. In *Proc. of Fourth International Workshop on Tree-Adjoining Grammars and Related Frameworks (TAG+4)*, pages 42–45, Philadelphia, PA, USA, August 1998.

[55] Eric de la Clergerie and François Barthélemy. Information flow in tabular interpretations for generalized Push-Down Automata. *Theoretical Computer Science*, 199(1–2):167–198, 1998.

[56] Eric de la Clergerie and Bernard Lang. LPDA: Another look at tabulation in logic programming. In Van Hentenryck, editor, *Proc. of the 11th International Conference on Logic Programming (ICLP'94)*, pages 470–486. MIT Press, June 1994.

[57] J. P. M. de Vreught and H. J. Honig. A tabular bottom-up recognizer. Technical Report 89-78, Department of Applied Mathematics and Informatics, Delft University of Technology, Delft, The Netherlands, 1989.

[58] Víctor J. Díaz Madrigal. *Gramáticas de adjunción de árboles: un enfoque deductivo en el análisis sintáctico.* PhD thesis, Universidad de Sevilla, Seville, Spain, 2000.

[59] Víctor J. Díaz Madrigal and Miguel A. Alonso Pardo. Análisis sintáctico bidireccional de TAGs. *Procesamiento del Lenguaje Natural*, 27, September 2000.

[60] Víctor J. Díaz Madrigal and Miguel A. Alonso Pardo. Comparing tabular parsers for tree adjoining grammars. In *Proc. of Second Interbnational Workshop on Tabulation in Parsing and Deduction (TAPD 2000)*, Vigo, Spain, 2000.

[61] Víctor J. Díaz Madrigal, Miguel A. Alonso Pardo, and Vicente Carrillo Montero. Bidirectional parsing of TAG without heads. In *Proc. of 5th International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+5)*, pages 67–72, Paris, France, May 2000.

[62] Víctor J. Díaz Madrigal, Vicente Carrillo Montero, and Miguel A. Alonso Pardo. A bidirectional bottom-up parser for TAG. In *Proc. of the Sixth International Workshop on Parsing Technologies (IWPT 2000)*, pages 299–300, Trento, Italy, February 2000.

[63] Víctor J. Díaz Madrigal, Vicente Carrillo Montero, and Miguel Toro Bonilla. Análisis sintáctico de TAG usando analizadores deductivos. *Procesamiento del Lenguaje Natural*, 23:126–131, September 1998.

[64] Víctor J. Díaz Madrigal, Vicente Carrillo Montero, and Miguel Toro Bonilla. Elementary tree representation. In *Proc. of First Workshop on Tabulation in Parsing and Deduction (TAPD'98)*, pages 10–15, Paris, France, April 1998.

[65] Víctor J. Díaz Madrigal, Vicente Carrillo Montero, and Miguel Toro Bonilla. Revisando el reconocedor con prefijo válido para TAGs de Schabes. *Procesamiento del Lenguaje Natural*, 25:59–65, September 1999.

[66] Víctor J. Díaz Madrigal and Miguel Toro Bonilla. Parsing TAGs with Prolog. In M. Falaschi, M. Navarro, and A. Policriti, editors, *Proc. of APPIA-GULP-PRODE'97 Joint Conference on Declarative Programming*, pages 359–367, Grado, Italy, June 1997.

[67] Christy Doran, Dania Egedi, Beth Ann Hockey, B. Srinivas, and Martin Zaidel. XTAG system — a wide coverage grammar for English. In *Proc. of the 15th International Conference on Computational Linguistics (COLING'94)*, pages 922–928, Kyoto, Japan, August 1994.

[68] J. Duske and R. Parchmann. Linear indexed languages. *Theoretical Computer Science*, 32:47–60, 1984.

[69] J. Earley. An efficient context-free parsing algorithm. *Communications of the ACM*, 13(2):94–102, 1970.

[70] Albert Einstein. *Sobre la teoría de la relatividad especial y general*, volume 6 of *Siete libros para entender el siglo XX*. Editorial Debate, S.A., Madrid, Spain, September 1998. Traducción de *Über die spezielle und allgemeine Relativitätstheorie*, 1916.

[71] Jason Eisner and Giorgio Satta. A faster parsing algorithm for lexicalized tree-adjoining grammars. In *Proc. of 5th International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+5)*, pages 79–84, Paris, France, May 2000.

[72] Roger Evans, Gerald Gazdar, and David Weir. Encoding lexicalized tree adjoining grammars with a nonmonotonic inheritance hierarchy. In *Proc. of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 77–84, Cambridge, Massachusetts, 1995. ACL.

[73] Roger Evans and David Weir. A structure-sharing parser for lexicalized grammars. In *COLING-ACL'98, 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Proceedings of the Conference*, volume I, pages 372–378, Montreal, Quebec, Canada, August 1998. ACL.

[74] Robert Frank, K. Vijay-Shanker, and John Chen. Dominance, precedence and C-command in description-based parsing. In C. Matín Vide, editor, *Lenguajes Naturales y Lenguajes Formales*, volume XII, pages 61–74, Barcelona, Spain, September 1996. PPU.

[75] Gerald Gazdar. Applicability of indexed grammars to natural languages. In U. Reyle and C. Rohrer, editors, *Natural Language Parsing and Linguistic Theories*, pages 69–94. D. Reidel Publishing Company, 1987.

[76] Jorge Graña Gil. *Técnicas de Análisis Sintáctico Robusto para la Etiquetación del Lenguaje Natural*. PhD thesis, Departamento de Computación, Universidade da Coruña, A Coruña, Spain, 2000.

[77] Andrew Haas. A parsing algorithm for unification grammar. *Computational Linguistics*, 15(4):219–232, 1989.

[78] Ariane Halber. Tree-grammar linear typing for unified super-tagging/probabilistic parsing models. In *Proc. of Fourth International Workshop on Tree Adjoining Grammars and Related Frameworks (TAG+4)*, pages 54–57, Philadelphia, PA, USA, August 1998.

[79] Chung-hye Han and Owen Rambow. The Sino-Korean light verb construction and lexical argument structure. In *Proc. of 5th International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+5)*, pages 93–100, Paris, France, May 2000.

[80] Karin Harbusch. An efficient parsing algorithm for tree adjoining grammars. In *Proc. of 28th Annual Meeting of the Association for Computational Linguistics*, pages 284–291, Pennsylvania, USA, 1990. ACL.

[81] Dominik Heckmann. Recursive matrix systems (RMS) — a highly parametrizable formal rewriting system. In Ivana Kruijff-Korbayová, editor, *Proc. of the Third ESSLLI Student Session*, pages 1–9, Saarbrücken, Germany, August 1998.

[82] Jan Heering, Paul Klint, and Jan Rekers. Incremental generation of parsers. *ACM SIGPLAN Notices (SIGPLAN'89 Conference on Programming Language Design and Implementation)*, 24(7):179–191, 1989.

[83] Jan Heering, Paul Klint, and Jan Rekers. Incremental generation of parsers. *IEEE Transactions on Software Engineering*, 16(12):1344–1350, 1990.

[84] Mark Hepple. On some similarities between D-tree grammars and type-logical grammars. In *Proc. of Fourth International Workshop on Tree-Adjoining Grammars and Related Frameworks (TAG+4)*, pages 66–69, Philadelphia, PA, USA, August 1998.

[85] John E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Series in Computer Science. Addison-Wesley Publishing Company, Reading, Massachussetts, USA, 1979.

[86] R. Nigel Horspool and Michael Whitney. Even faster LR parsing. *Software — Practice and Experience*, 20(6):515–535, 1990.

[87] Rebecca Hwa. An empirical evaluation of probabilistic lexicalized tree insertion grammars. In *COLING-ACL'98, 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Proceedings of the Conference*, volume I, pages 557–563, Montreal, Quebec, Canada, August 1998. ACL.

[88] Mark Johnson. The computational complexity of GLR parsing. In Masaru Tomita, editor, *Generalized LR Parsing*, chapter 3, pages 35–42. Kluwer Academic Publishers, Boston/Dordrecht/London, 1991.

[89] Mark Johnson. Logical embedded push-down automata in tree-adjoining grammar parsing. *Computational Intelligence*, 10(4):495–505, 1994.

[90] Aravind K. Joshi. Tree adjoining grammars: How much context-sensitivity is required to provide reasonable structural descriptions? In David R. Dowty, Lauri Karttunen, and Arnold M. Zwicky, editors, *Natural Language Parsing. Psychological, Computational and Theoretical Perspectives*, chapter 6, pages 206–250. Cambridge University Press, 1985.

[91] Aravind K. Joshi. An introduction to tree adjoining grammars. In Alexis Manaster-Ramer, editor, *Mathematics of Language*, pages 87–115. John Benjamins Publishing Co., Amsterdam/Philadelphia, 1987.

[92] Aravind K. Joshi. Relationship between strong and weak generative power of formal systems. In *Proc. of 5th International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+5)*, pages 107–113, Paris, France, May 2000.

[93] Aravind K. Joshi, Leon S. Levy, and M. Takahashi. Tree adjunt grammars. *Journal of Computer and System Sciences*, 10(1):136–162, February 1975.

[94] Aravind K. Joshi and Yves Schabes. Tree-adjoining grammars. In Grzegorz Rozenberg and Arto Salomaa, editors, *Handbook of Formal Languages. Vol 3: Beyond Words*, chapter 2, pages 69–123. Springer-Verlag, Berlin/Heidelberg/New York, 1997.

[95] Aravind K. Joshi, K. Vijay-Shanker, and David Weir. The convergence of mildly context-sensitive grammar formalisms. In P. Sells, Shieber S. M., and T. Warsow, editors, *Foundational Issues in Natural Language Processing*, pages 31–81. MIT Press, Cambridge, MA, USA, 1991.

[96] Ronald M. Kaplan. The formal architecture of lexical-functional grammar. In Mary Darymple, Ronald M. Kaplan, John T. Maxwell III, and Annie Zaenen, editors, *Formal Issues in Lexical-Functional Grammar*. Stanford University, 1994.

[97] T. Kasami. An efficient recognition and syntax algorithm for context-free languages. Scientific Report AFCRL-65-758, Air Force Cambridge Research Lab., Bedford, Massachussetts, 1965.

[98] Bill Keller and David Weir. A tractable extension of linear indexed grammars. In *Proc. of the 7th Conference of the European Chapter of the ACL*, pages 75–86. EACL, 1995.

[99] Alexandra Kinyon. Un algorithme d'analyse LR(0) pour les Grammaires d'Arbres Adjoints Lexicalisées. In D. Genthial, editor, *Actes de la quatrième conférence annuelle sur Le Traitement Automatique du Langage Naturel*, pages 93–102, Grenoble, France, June 1997.

[100] James R. Kipps. GLR parsing in time $\mathcal{O}(n^3)$. In Masaru Tomita, editor, *Generalized LR Parsing*, chapter 4, pages 43–59. Kluwer Academic Publishers, Boston/Dordrecht/London, 1991.

[101] Anthony S. Kroch. Unbounded dependencies and subjacency in a tree adjoining grammar. In Alexis Manaster-Ramer, editor, *Mathematics of Language*, pages 143–172. John Benjamins Publishing Company, Amsterdam/Philadelphia, 1987.

[102] S. Kulkarni Kulekha and Priti Shankar. Linear time parsers for classes of non context free languages. *Theoretical Computer Science*, 165:355–390, 1996.

[103] Seth Kulick. Clitic climbing and tree adjoining grammar. TAG+4 Tutorial, IRCS, Philadelphia, PA, USA, July 1998.

[104] Bernard Lang. Deterministic techniques for efficient non-deterministic parsers. In *Proc. of 2nd Colloquium on Automata, Languages and Programming (ICALP'74), Saarbrüecken, Germany*, volume 14 of *Lecture Notes in Computer Science*, pages 255–269. Springer Verlag, Berlin-Heidelberg-New York, 1974.

[105] Bernard Lang. Complete evaluation of Horn Clauses, an automata theoretic approach. Rapport de Recherche 913, INRIA, Rocquencourt, France, November 1988.

[106] Bernard Lang. The systematic construction of Earley parsers: Application to the production of $\mathcal{O}(n^6)$ Earley parsers for tree adjoining grammars. In *Proc. of the 1st International Workshop on Tree Adjoining Grammars*, August 1990.

[107] Bernard Lang. Towards a uniform formal framework for parsing. In Masaru Tomita, editor, *Current Issues in Parsing Technology*, pages 153–171. Kluwer Academic Publishers, Norwell, MA, USA, 1991.

[108] Bernard Lang. Recognition can be harder than parsing. *Computational Intelligence*, 10(4):486–494, 1994.

[109] Alberto Lavelli and Giorgio Satta. Bidirectional parsing of lexicalized tree adjoining grammars. In *Proceedings of the 5th Conference of the European Chapter of the Association for Computational Linguistics (EACL'91)*, Berlin, Germany, April 1991. ACL.

[110] Manuela Leahu. Wh-dependecies in Romanian and TAG. In *Proc. of Fourth International Workshop on Tree Adjoining Grammars and Related Frameworks (TAG+4)*, pages 92–95, Philadelphia, PA, USA, August 1998.

[111] René Leermakers. *The Functional Treatment of Parsing.* The Kluwer International Series in Engineering and Computer Science. Natural Language Processing and Machine Translation. Kluwer Academic Publishers, Boston/Dordrecht/London, 1993.

[112] Patrice Lopez. *Analyse d'énoncés oraux pour le Dialogue Homme-Machine à l'aide de Grammaires Lexicalisées d'Arbres.* PhD thesis, Université Henri Poincaré – Nancy 1, Nancy, France, October 1999.

[113] Patrice Lopez. Repair strategies for lexicalized tree grammars. In *Proc. of EACL'99, Ninth Conference of the European Chapter of the Association for Computational Linguistics*, pages 249–252, Bergen, Norway, June 1999. ACL.

[114] Patrice Lopez. Extended partial parsing for lexicalized tree grammars. In *Proc. of the Sixth International Workshop on Parsing Technologies (IWPT 2000)*, pages 159–170, Trento, Italy, February 2000.

[115] Patrice Lopez and David Roussel. Which rules for the robust parsing of spoken utterances with Lexicalized Tree Adjoining Grammars ? In *Proc. of Fourth International Workshop on Tree Adjoining Grammars and Related Frameworks (TAG+4)*, pages 96–99, Philadelphia, PA, USA, August 1998.

[116] Solomon Marcus, Carlos Martín-Vide, and Gheorghe Păun. Contextual Grammars as generative models of natural languages. *Computational Linguistics*, 24(2):245–274, June 1998.

[117] David D. McDonald and James D. Pustejovsky. TAG's as a grammatical formalism for generation. In *23rd Annual Meeting of the Association for Computational Linguistics*, pages 94–103, Chicago, IL, USA, July 1985. ACL.

[118] Philippe McLean and R. Nigel Horspool. A faster Earley parser. In *Proc. of International Conference on Compiler Construction (CC'96)*, pages 281–293, Linkopen, Sweden, 1996.

[119] Guido Minnen. Predictive left-to-right parsing of a restricted variant of TAG(LD/LP). *Computational Intelligence*, 10(4):535–546, 1994.

[120] Tsuneko Nakazawa. Construction of LR parsing tables for grammars using feature-based syntactic categories. In Jennifer Cole, Georgia M. Green, and Jerry L. Morgan, editors, *Linguistics and Computation*, number 52 in CSLI Lecture Notes, pages 199–219. CSLI Publications, Stanford, CA, USA, 1995.

[121] Mark-Jan Nederhof. *Linguistic Parsing and Program Transformations.* PhD thesis, Katholieke Universiteit Nijmegen, Nijmegen, The Netherlands, October 1994.

[122] Mark-Jan Nederhof. An optimal tabular parsing algorithm. In *Proc. of 32nd Annual Meeting of the Association for Computational Linguistics*, pages 117–124, Las Cruces, NM, USA, June 1994. ACL.

[123] Mark-Jan Nederhof. An alternative LR algorithm for TAGs. In *COLING-ACL'98, 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Proceedings of the Conference*, volume II, pages 946–952, Montreal, Quebec, Canada, August 1998. ACL.

[124] Mark-Jan Nederhof. Linear indexed automata and tabulation of TAG parsing. In *Proc. of First Workshop on Tabulation in Parsing and Deduction (TAPD'98)*, pages 1–9, Paris, France, April 1998.

[125] Mark-Jan Nederhof. The computational complexity of the correct-prefix property for TAGs. *Computational Linguistics*, 25(3):345–360, 1999.

[126] Mark-Jan Nederhof. Models of tabulation for TAG parsing. In *Proc. of the Sixth Meeting on Mathematics of Language (MOL 6)*, pages 143–158, Orlando, Florida, USA, July 1999.

[127] Mark-Jan Nederhof and J. J. Sarbo. Increasing the applicability of LR parsing. In Harry Bunt and Masaru Tomita, editors, *Recent Advances in Parsing Technology*, volume 1 of *Text, Speech and Language Technology*, chapter 3, pages 35–57. Kluwer Academic Publishers, Dordrecht/Boston/London, 1996.

[128] Mark-Jan Nederhof, Anoop Sarkar, and Giorgio Satta. Prefix probabilistic from stochastic tree adjoining grammars. In *COLING-ACL'98, 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Proceedings of the Conference*, volume II, pages 953–959, Montreal, Quebec, Canada, August 1998. ACL.

[129] Mark-Jan Nederhof, Anoop Sarkar, and Giorgio Satta. Prefix probabilities for linear indexed grammars. In *Proc. of Fourth International Workshop on Tree Adjoining Grammars and Related Frameworks (TAG+4)*, pages 116–119, Philadelphia, PA, USA, August 1998.

[130] Mark-Jan Nederhof and Giorgio Satta. Efficient tabular LR parsing. In *Proc. of 34th Annual Meeting of the Association for Computational Linguistics*, Santa Cruz, CA, USA, June 1996. ACL.

[131] Mark-Jan Nederhof and Giorgio Satta. A variant of Earley parsing. In *AI+IA 97: Advances in Artificial Intelligence, 5th Congress of the Italian Association for Artificial intelligence*, volume 1321 of *Lecture Notes in Artificial Intelligence*, pages 84–95. Springer-Verlag, New York-Heidelberg-Berlin, 1997.

[132] Carol Neidle. Lexical functional grammars. In *Encyclopaedia of Language and Linguistics.* Pergamon Press, New York, NY, USA, 1994.

[133] Günter Neumann. Automatic extraction of stochastic lexicalized tree grammars from treebaks. In *Proc. of Fourth International Workshop on Tree Adjoining Grammars and Related Frameworks (TAG+4)*, pages 120–123, Philadelphia, PA, USA, August 1998.

[134] Ulf Nilsson. AID: An alternative implementation of DCG. *New Generation Computing*, 4:383–399, 1986.

[135] Rahman Nozohoor-Farshi. GLR parsing for ϵ-grammers. In Masaru Tomita, editor, *Generalized LR Parsing*, chapter 5, pages 61–75. Kluwer Academic Publishers, Boston/Dordrecht/London, 1991.

[136] Tom Nurkkala and Vipin Kumar. A parallel parsing algorithm for natural language using Tree Adjoining Grammar. In *Proc. of the 8th International Parallel Processing Symposium*, 1994.

[137] Tom Nurkkala and Vipin Kumar. The performance of a highly unstructured parallel algorithm on the KSRI. In *Scalable High Performance Computing Conference*, Knoxville, May 1994.

[138] Rudolf Ortega i Robert. Gramáticas suavemente dependientes del contexto (MCSG). In C. Matín Vide, editor, *Lenguajes Naturales y Lenguajes Formales*, volume XII, pages 103–119, Barcelona, Spain, September 1996. PPU.

[139] Michael A. Palis, Sunil Shende, and David S. L. Wei. An optimal linear-time parallel parser for Tree Adjoining Languages. *SIAM Journal of Computing*, 19(1):1–31, 1990.

[140] Michael A. Palis and David S. L. Wei. Parallel parsing of tree adjoining grammars on the connection machine. *Internatonal journal of Parallel Programming*, 21(1):1–38, May 1992.

[141] Michael A. Palis and David S. L. Wei. Massively parallel parsing algorithms for natural language. In L. Kanal, V. Kumar, C. Suttner, and H. Kitano, editors, *Parallel Processing for Artificial Intelligence*, pages 365–407. North-Holland, 1994.

[142] Adi Palm. A logical approach to structure sharing in TAGs. In *Proc. of 5th International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+5)*, pages 171–176, Paris, France, May 2000.

[143] Fernando C. N. Pereira and David H. D. Warren. Definite Clause Grammars for language analysis — a survey of the formalism and a comparison with Augmented Transition Networks. *Artificial Intelligence*, 13:231–278, 1980.

[144] Fernando C. N. Pereira and David H. D. Warren. Parsing as deduction. In *Proc. of the 21st Annual Meeting of the Association for Computational Linguistics*, pages 137–144. ACL, June 1983.

[145] Gisela Pitsch. $LL(k)$ parsing of coupled-context-free grammars. *Computational Intelligence*, 10(4):563–578, 1994.

[146] C. Pollard. *Generalized Phrase Structure Grammars, Head Grammars and Natural Language*. PhD thesis, Stanford University, 1984.

[147] Carl Pollard and Ivan A. Sag. *Head-Driven Phrase Structure Grammar*. Studies in Contemporary Linguistics. The University of Chicago Press, Chicago & London, 1994.

[148] Peter Poller. Incremental parsing with LD/TLP-TAGs. *Computational Intelligence*, 10(4):549–562, 1994.

[149] Peter Poller and Tilman Becker. Two-step TAG parsing revisited. In *Proc. of Fourth International Workshop on Tree Adjoining Grammars and Related Frameworks (TAG+4)*, pages 143–146, Philadelphia, PA, USA, August 1998.

[150] Carlos A. Prolo. An efficient LR parser generator for tree adjoining grammars. In *Proc. of the Sixth International Workshop on Parsing Technologies (IWPT 2000)*, pages 207–218, Trento, Italy, February 2000.

[151] Sanguthevar. Rajasekaran. TAL parsing in $\mathcal{O}(n^6)$ time. *SIAM Journal on Computing*, 25(4):862–873, 1996.

[152] Owen Rambow. *Formal and Computational Aspects of Natural Language Syntax.* PhD thesis, University of Pennsylvania, 1994. Available as IRCS Report 94-08 of the Institute of Research in Cognitive Science, University of Pennsylvania.

[153] Owen Rambow. Multiset-Valued Linear Index Grammars: Imposing dominance constraints on derivations. In *Proc. of 32nd Annual Meeting of the Association for Computational Linguistics*, Las Cruces, New Mexico, USA, June 1994. ACL.

[154] Owen Rambow. The German "coherent construction". TAG+4 Tutorial, IRCS, Philadelphia, PA, USA, July 1998.

[155] Owen Rambow, K. Vijay-Shanker, and David Weir. D-Tree grammars. In *Proc. of 33rd Annual Meeting of the Association for Computational Linguistics*, Cambridge, MA, USA, June 1995. ACL.

[156] Owen Rambow, K. Vijay-Shanker, and David Weir. Parsing D-Tree grammars. In *Proc. of the International Workshop on Parsing Technologies*, 1995.

[157] Jan Rekers. *Parsing Generation for Interactive Environments.* PhD thesis, University of Amsterdam, Amsterdam, The Netherlands, 1992.

[158] Philip Resnik. Probabilistic tree-adjoining grammar as a framework for natural language processing. In *Proc. of Fifteenth International Conference on Computational Linguistics (COLING'92)*, pages 418–424, Nantes, France, August 1992.

[159] Kelly Roach. Formal properties of head grammars. In Alexis Manaster-Ramer, editor, *Mathematics of Language*, pages 293–347. John Benjamins Publishing Company, Amsterdam/Philadelphia, 1987.

[160] James Rogers. Capturing CFLs with tree adjoining grammars. In *Proc. of 32nd Annual Meeting of the Association for Computational Linguistics*, Las Cruces, New Mexico, USA, June 1994. ACL.

[161] James Rogers. A unified notion of derived and derivation structures in TAG. In T. Becker and H.-V. Krieger, editors, *Proc. of the Fifth Meeting on Mathematics of Language*, Schloss Dagstuhl, Saarbruecken, Germany, 1997.

[162] David. A. Rosenblueth and Julio C. Peralta. LR inference: Inference systems for fixed-mode logic programs based on LR parsing. In Maurice Bruynooghe, editor, *Logic Programming. Proceedings of the 1994 International Symposium (ILPS'94)*, pages 439–453, Cambridge, MA, USA, 1994. MIT Press.

[163] Anoop Sarkar. Incremental parser generation for tree adjoining grammars. In *Proceedings of the 34th Meeting of the ACL, Student Session*, Santa Cruz, California, USA, June 1996. ACL.

[164] Anoop Sarkar. Conditions on consistency of probabilistic tree adjoining grammars. In *COLING-ACL'98, 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Proceedings of the Conference*, volume II, pages 1164–1170, Montreal, Quebec, Canada, August 1998. ACL.

[165] Anoop Sarkar. Practical experiments in parsing using tree adjoining grammars. In *Proc. of 5th International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+5)*, pages 193–198, Paris, France, May 2000.

[166] Giorgio Satta. Tree-adjoining grammar parsing and boolean matrix multiplication. *Computational Linguistics*, 20(2):173–191, 1994.

[167] Giorgio Satta and William Schuler. Restrictions on tree adjoining languages. In *COLING-ACL'98, 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Proceedings of the Conference*, volume II, pages 1176–1182, Montreal, Quebec, Canada, August 1998. ACL.

[168] Yves Schabes. *Mathematical and Computational Aspects of Lexicalized Grammars*. PhD thesis, University of Pennsylvania, 1990. Available as Technical Report MS-CIS-90-48 LINC LAB 179 of the Department of Computer and Information Science, University of Pennsylvania.

[169] Yves Schabes. The valid prefix property and left to right parsing of tree-adjoining grammar. In *Proc. of II International Workshop on Parsing Technologies, IWPT'91*, pages 21–30, Cancún, Mexico, 1991.

[170] Yves Schabes. Stochastic lexicalized tree-adjoining grammars. In *Proc. of Fifteenth International Conference on Computational Linguistics (COLING'92)*, pages 426–432, Nantes, France, August 1992.

[171] Yves Schabes. Lexicalized context-free grammars. In *Proceedings of the 31st Meeting of the Association for Computational Linguistics*, pages 121–129, Columbus, Ohio, USA, June 1993. ACL. Also as Technical Report TR-93-01, January 1993, Mitsubishi Electric Research Laboratories, Cambridge, MA, USA.

[172] Yves Schabes. Left to right parsing of lexicalized tree-adjoining grammars. *Computational Intelligence*, 10(4):506–515, 1994.

[173] Yves Schabes and Aravind K. Joshi. An Earley-type parsing algorithm for tree adjoining grammars. In *Proc. of 26th Annual Meeting of the Association for Computational Linguistics*, pages 258–269, Buffalo, NY, USA, June 1988. ACL.

[174] Yves Schabes and Aravind K. Joshi. Parsing with lexicalized tree adjoining grammar. In Masaru Tomita, editor, *Current Issues in Parsing Technologies*, chapter 3, pages 25–47. Kluwer Academic Publishers, Norwell, MA, USA, 1991.

[175] Yves Schabes and Stuart M. Shieber. An alternative conception of tree-adjoining derivation. *Computational Linguistics*, 20(1):91–124, 1994.

[176] Yves Schabes and K. Vijay-Shanker. Deterministic left to right parsing of tree adjoining languages. In *Proc. of 28th Annual Meeting of the Association for Computational Linguistics*, pages 276–283, Oittsburgh, Pennsylvania, USA, June 1990. ACL.

[177] Yves Schabes and Richard C. Waters. Lexicalized context-free grammar: A cubic-time parsable, lexicalized normal form for context-free grammar that preserves tree structure. Technical Report 93-04, Mitsubishi Electric Research Laboratories. Cambridge Research Center, 201 Broadway, Cambridge, Massachusetts 02139, USA, June 1993.

[178] Yves Schabes and Richard C. Waters. Stochastic lexicalized context-free grammar. In *Proc. of the Third International Workshop on Parsing Technologies (IWPT'93)*, pages 257–266, Tilburg (The Netherlands) and Durbuy (Belgium), August 1993. Also as Technical Report TR-93-12, July 1993, Mitsubishi Electric Research Laboratories, Cambridge, MA, USA.

[179] Yves Schabes and Richard C. Waters. Tree insertion grammar: A cubic-time parsable formalism that lexicalizes context-free grammar without changing the trees produced. *Computational Linguistics*, 21(4):479–513, December 1995. Also as Technical Report TR-94-13, June 1994, Mitsubishi Electric Research Laboratories, Cambridge, MA, USA.

[180] Yves Schabes and Richard C. Waters. Stochastic lexicalized tree-insertion grammar. In Harry Bunt and Masaru Tomita, editors, *Recent Advances in Parsing Technology*, volume 1 of *Text, Speech and Language Technology*, chapter 15, pages 281–294. Kluwer Academic Publishers, Dordrecht/Boston/London, 1996.

[181] Karl-Michael Schneider. Algebraic construction of parsing schemata. In *Proc. of the Sixth International Workshop on Parsing Technologies (IWPT 2000)*, pages 242–253, Trento, Italy, February 2000.

[182] Karl-Michael Schneider. Tabular parsing and algebraic transformations. In Dirk Heylen, Anton Nijholt, and G. Scollo, editors, *Proc. of 16th Twente Workshop on Language Technology: Algebraic Methods in Language Processing (TWLT 16/AMiLP 2000)*, pages 233–250, Iowa City, Iowa, USA, 2000.

[183] Robert Sedgewick. *Algorithms.* Addison-Wesley Series in Computer Sience. Addison-Wesley Publishing Company, Inc., 1988.

[184] B. A. Sheil. Observations on context-free grammars. In *Statistical Methods in Linguistics*, pages 71–109, Stockholm, Sweden, 1976.

[185] Stuart M. Shieber. Using restriction to extend parsing algorithms for complex-feature-based formalisms. In *Proc. of the 23th Annual Meeting of the Association for Computational Linguistics*, pages 145–152. ACL, June 1985.

[186] Stuart M. Shieber. *Constraint Based Grammar Formalisms.* MIT Press, Cambridge, MA, USA, 1992.

[187] Stuart M. Shieber and Yves Schabes. Synchronous tree-adjoining grammars. In *Proc. of the 13th International Conference on Computational Linguistics (COLING'90)*, Helsinki, Finland, August 1990.

[188] Stuart M. Shieber, Yves Schabes, and Fernando C. N. Pereira. Principles and implementation of deductive parsing. *Journal of Logic Programming*, 24(1–2):3–36, July-August 1995.

[189] Klaas Sikkel. *Parsing Schemata — A Framework for Specification and Analysis of Parsing Algorithms.* Texts in Theoretical Computer Science — An EATCS Series. Springer-Verlag, Berlin/Heidelberg/New York, 1997.

[190] Klaas Sikkel. Parsing schemata and correctness of parsing algorithms. *Theoretical Computer Science*, 199(1–2):87–103, 1998.

[191] Klaas Sikkel and Anton Nijholt. Parsing of context-free languages. In G. Rozenberg and A. Salomaa, editors, *The Handbook of Formal Languages*, volume 2: Linear Modeling: Background and Application, pages 61–100. Springer Verlag, Berlin-Heidelberg-New York, 1997. Also available as Technical Report 96-32, Center for Telematics and Information Technology, University of Twente, Enschede, The Netherlands, 1996.

[192] Martine Smets. Comparison of XTAG and LEXSYS grammars. In *Proc. of Fourth International Workshop on Tree-Adjoining Grammars and Related Frameworks (TAG+4)*, pages 159–163, Philadelphia, PA, USA, August 1998.

[193] Martine Smets and Roger Evans. A compact encoding of a DTG grammar. In *Proc. of Fourth International Workshop on Tree-Adjoining Grammars and Related Frameworks (TAG+4)*, pages 164–165, Philadelphia, PA, USA, August 1998.

[194] M. Steedman. Combinators and grammars. In R. Oehrle, E. Bach, and D Wheeler, editors, *Categorial Grammars and Natural Language Structures*, pages 417–442. Foris, Dordrecht, 1986.

[195] Hisao Tamaki and Taisuke Sato. OLD resolution with tabulation. In *Proc. of Third International Conference on Logic Programming*, pages 84–98, London, U.K., 1986. Springer-Verlag.

[196] Hozumi Tanaka and Hiroaki Numazaki. Parallel GLR parsing based on logic programming. In Masaru Tomita, editor, *Generalized LR Parsing*, chapter 6, pages 67–91. Kluwer Academic Publishers, Boston/Dordrecht/London, 1991.

[197] Frédéric Tendeau. *Analyse syntaxique et sémantique avec évaluation d'attributs dans un demi-anneau. Applications à la linguistique calculatoire.* PhD thesis, Université d'Orléans, France, June 1997.

[198] The XTAG Research Group. A lexicalized tree adjoining grammar for English. Technical Report IRCS 95-03, Institute for Research in Cognitive Science, University of Pennsylvania, 3401 Walnut ST., Suite 400C, Philadelphia, PA 19104-6228, March 1995.

[199] Masaru Tomita. *Efficient Parsing for Natural Language. A Fast Algorithm for Practical Systems.* The Kluwer International Series in Engineering and Computer Science. Natural Language Processing and Machine Translation. Kluwer Academic Publishers, Boston/Dordrecht/Lancaster, 1986.

[200] Masaru Tomita. An efficient augmented context-free parsing algorithm. *Computational Linguistics*, 13(1–2):31–46, 1987.

[201] Masaru Tomita and See-Kiong Ng. The generalized LR parsing algorithm. In Masaru Tomita, editor, *Generalized LR Parsing*, chapter 1, pages 1–16. Kluwer Academic Publishers, Boston/Dordrecht/London, 1991.

[202] Arturo Trujillo. Computing FIRST and FOLLOW functions for feature-theoretic grammars. In *Proc. of the fifteenth International Conference on Computational Linguistics (COLING'94)*, Kyoto, Japan, August 1994.

[203] Kuniaki Uehora, Ryo Ochitani, Osamu Kakusho, and Junichi Toyoda. A bottom-up parser based on predicate logic: A survey of the formalism and its implementation technique. In *Proc. of the 1984 INternational Symposium on Logic Programming*, pages 220–227, 1984.

[204] Gertjan van Noord. *Reversibility in Natural Language Processing.* PhD thesis, Rijksuniversiteit te Utrecht, The Netherlands, January 1993.

[205] Gertjan van Noord. Head-corner parsing for TAG. *Computational Intelligence*, 10(4):525–534, 1994.

[206] K. Vijay-Shanker. *A Study of Tree Adjoining Grammars.* PhD thesis, University of Pennsylvania, January 1988. Available as Technical Report MS-CIS-88-03 LINC LAB 95 of the Department of Computer and Information Science, University of Pennsylvania.

[207] K. Vijay-Shanker. The use of domination statements in TAG and related formalisms. In C. Matín Vide, editor, *Lenguajes Naturales y Lenguajes Formales*, volume XII, pages 327–350, Barcelona, Spain, September 1996. PPU.

[208] K. Vijay-Shanker. D-tree grammars. TAG+4 Tutorial, IRCS, Philadelphia, PA, USA, July 1998.

[209] K. Vijay-Shanker and Aravind K. Joshi. Some computational properties of tree adjoining grammars. In *23rd Annual Meeting of the Association for Computational Linguistics*, pages 82–93, Chicago, IL, USA, July 1985. ACL.

[210] K. Vijay-Shanker and Aravind K. Joshi. Feature structures based tree adjoining grammars. In Dénes Vargha, editor, *Proc. of the 12nd International Conference on Computational Linguistics (COLING'88)*, pages 714–719, Budapest, Hungary, 1988.

[211] K. Vijay-Shanker and Aravind K. Joshi. Unification-based tree adjoining grammars. In J. Wedekind, editor, *Unification Based Grammars.* MIT Press, Cambridge, MA, USA, 1991.

[212] K. Vijay-Shanker and Yves Schabes. Structure sharing in lexicalized tree-adjoining grammars. In *Proc. of Fifteenth International Conference on Computational Linguistics (COLING'92)*, pages 205–211, Nantes, France, August 1992.

[213] K. Vijay-Shanker and David J. Weir. Polynomial parsing of extensions of context-free grammars. In Masaru Tomita, editor, *Current Issues in Parsing Technology*, chapter 13, pages 191–206. Kluwer Academic Publishers, Norwell, MA, USA, 1991.

[214] K. Vijay-Shanker and David J. Weir. Parsing some constrained grammar formalisms. *Computational Linguistics*, 19(4):591–636, 1993.

[215] K. Vijay-Shanker and David J. Weir. The use of shared forest in tree adjoining grammar parsing. In *Proc. of the 6th Conference of the European Chapter of ACL*, pages 384–393. ACL, 1993.

[216] K. Vijay-Shanker and David J. Weir. The equivalence of four extensions of context-free grammars. *Mathematical Systems Theory*, 27:511–545, 1994.

[217] K. Vijay-Shanker, David J. Weir, and Aravind K. Joshi. Characterizing structural descriptions produced by various grammatical formalisms. In *Proc. of the 25th Annual Meeting of the Association for Computational Linguistics*, pages 104–111, Buffalo, NY, USA, June 1987. ACL.

[218] K. Vijay-Shanker, David J. Weir, and Aravind K. Joshi. On the progression from context-free to tree adjoining languages. In Alexis Manaster-Ramer, editor, *Mathematics of Language*, pages 389–401. John Benjamins Publishing Company, Amsterdam/Philadelphia, 1987.

[219] Manuel Vilares Ferro. *Efficient Incremental Parsing for Context-Free Languages.* PhD thesis, Université de Nice, France, 1992.

[220] Manuel Vilares Ferro and Miguel A. Alonso Pardo. Exploring interactive chart parsing. *Procesamiento del Lenguaje Natural*, 17:158–170, September 1995.

[221] Manuel Vilares Ferro and Miguel A. Alonso Pardo. A predictive bottom-up evaluator. *Logic Programming Newsletter*, 8(4):9–10, 1995.

[222] Manuel Vilares Ferro and Miguel A. Alonso Pardo. An LALR extension for DCGs in dynamic programming. In Carlos Martín Vide, editor, *Mathematical and Computational Analysis of Natural Language*, volume 45 of *Studies in Functional and Structural Linguistics*, pages 267–278. John Benjamins Publishing Company, Amsterdam & Philadelphia, 1998.

[223] Manuel Vilares Ferro, Miguel A. Alonso Pardo, and David Cabrero Souto. An operational model for parsing definite clause grammars with infinite terms. In Alain Lecomte, François Lamarche, and Guy Perrier, editors, *Logical Aspects of Computational Linguistics*, volume 1582 of *Lecture Notes in Artificial Intelligence.* Springer-Verlag, Berlin-Heidelberg-New York, 1999.

[224] Manuel Vilares Ferro, Miguel A. Alonso Pardo, Jorge Graña Gil, and David Cabrero Souto. GALENA: Tabular DCG parsing for natural languages. In *Proc. of First Workshop on Tabulation in Parsing and Deduction (TAPD'98)*, pages 44–51, Paris, France, April 1998.

[225] Manuel Vilares Ferro, Miguel A. Alonso Pardo, and Alberto Valderruten Vidal. *Programación Lógica.* Editorial Tórculo, Santiago de Compostela, Spain, 2nd edition, 1996.

[226] Manuel Vilares Ferro, David Cabrero Souto, and Miguel A. Alonso Pardo. Dynamic programming as frame for efficient parsing. In *SCCC'98, XVIII International Conference of the Chilean Computer Science Society, November 9-14, 1998. Antofagasta, Chile*, pages 68–75, Los Alamitos, CA, November 1998. IEEE Computer Society Press.

[227] Manuel Vilares Ferro and Bernard A. Dion. Efficient incremental parsing for context-free languages. In *Proc. of the* 5^{th} *IEEE International Conference on Computer Languages*, pages 241–252, Toulouse, France, 1994.

[228] Christian Wartena. Extending linear indexed grammars. In *Proc. of 5th International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+5)*, pages 207–214, Paris, France, May 2000.

[229] Éric Wehrli. *L'analyse syntaxique des langues naturelles.* Masson, Paris, 1997.

[230] David J. Weir. *Characterizing Mildly Context-Sensitive Grammar Formalisms.* PhD thesis, University of Pennsylvania, 1988. Available as Technical Report MS-CIS-88-74 of the Department of Computer and Information Sciences, University of Pennsylvania.

[231] David J. Weir. Linear iterated pushdowns. *Computational Intelligence*, 10(4):422–430, 1994.

[232] Mats Wirén and Ralph Rönnquist. Fully incremental parsing. In Harry Bunt and Masaru Tomita, editors, *Recent Advances in Parsing Technology*, volume 1 of *Text, Speech and Language Technology*, chapter 2, pages 11–34. Kluwer Academic Publishers, Dordrecht/Boston/London, 1996.

[233] Juntae Yoon, Chung-hye Han, Nari Kim, and Meesook Kim. Customizing the XTAG system for efficient grammar development for Korean. In *Proc. of 5th International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+5)*, pages 221–226, Paris, France, May 2000.

[234] D. H. Younger. Recognition and parsing of context-free languages in time n^3. *Information and Control*, 10(2):189–208, 1967.

Índice Onomástico

Índice de Materias

E

F

U

W