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. ¶

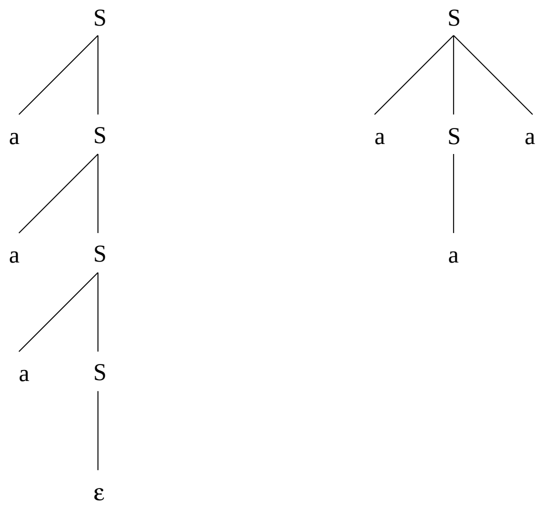

Figura 2.1: Estruturas asociadas a la cadena aaa por $\mathcal{G}_1$ y por $\mathcal{G}_2$

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.

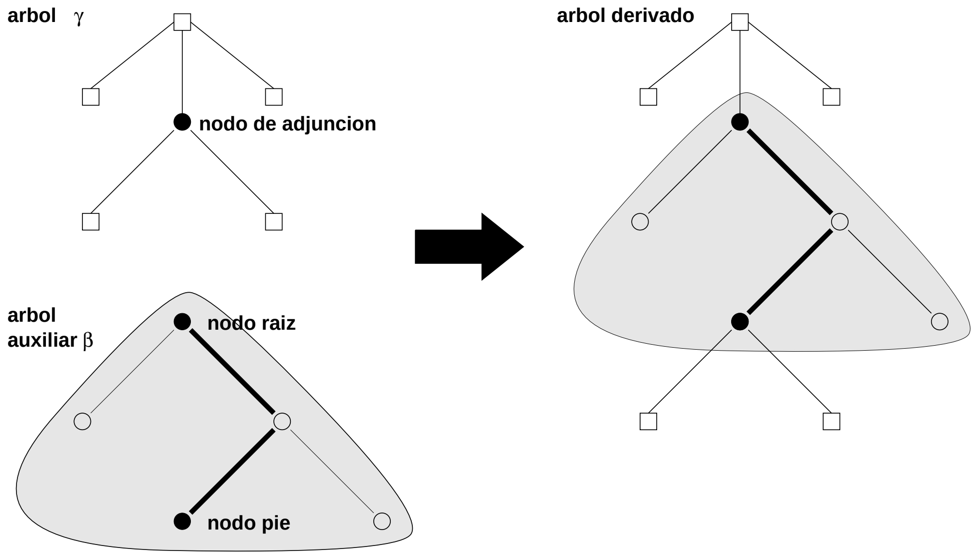

Figura 2.2: Operación de adjunción

- 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.

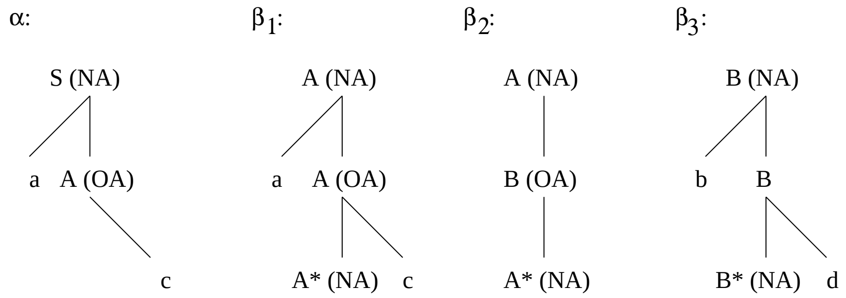

Figura 2.3: Gramática de adjunción de árboles que genera el lenguaje $a^n b^m c^n d^m$

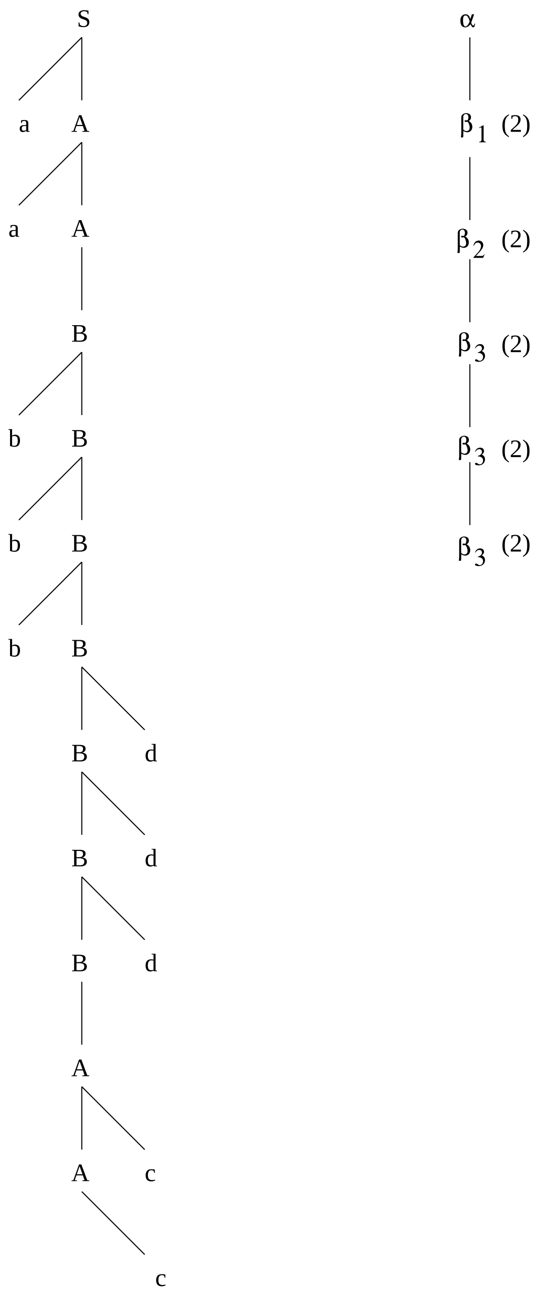

Figura 2.4: Árbol derivado (izquierda) y de derivación (derecha) en TAG para $aabbbccddd$

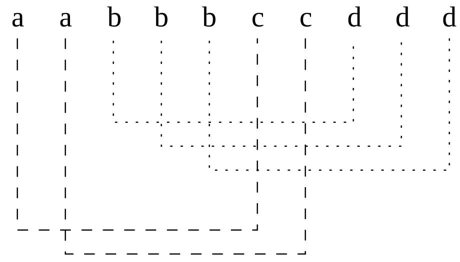

Figura 2.5: Relaciones cruzadas en la cadena *aabbbccddd*

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. ¶

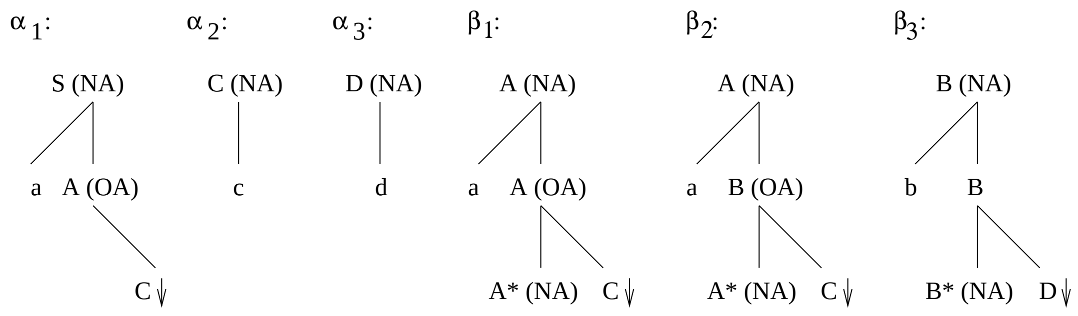

Figura 2.6: TAG lexicalizada que genera el lenguaje $a^n b^m c^n d^m$

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.

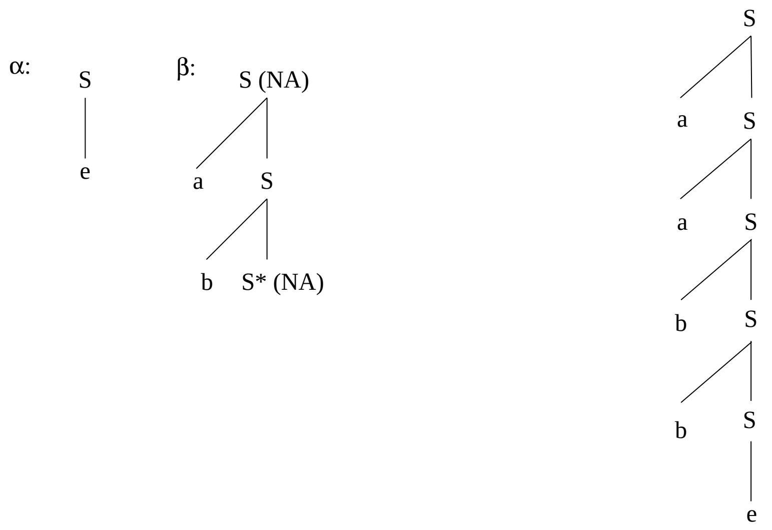

Figura 2.7: Gramática de adjunción de árboles para $a^n b^n e$ y árbol derivado para *aabbe*

- 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:

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

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 $S \rightarrow NP \quad V \quad NP$ 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. ¶

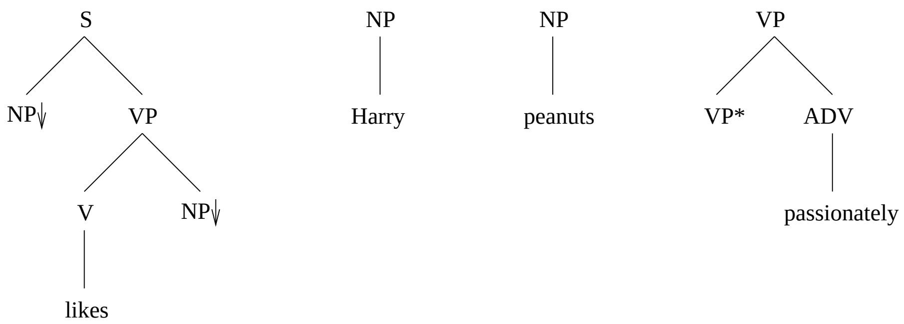

Figura 2.8: Dominio extendido de localidad de las TAG

- 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

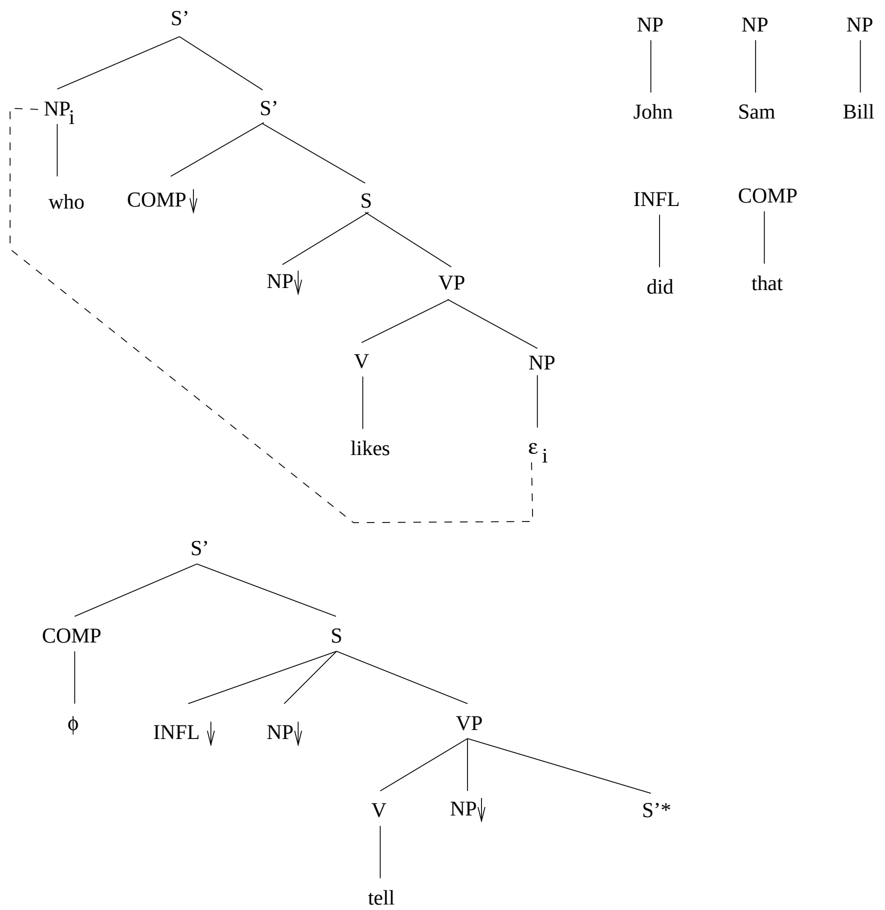

Figura 2.9: Dominio de localidad de la dependencia entre who_i y ϵ_i

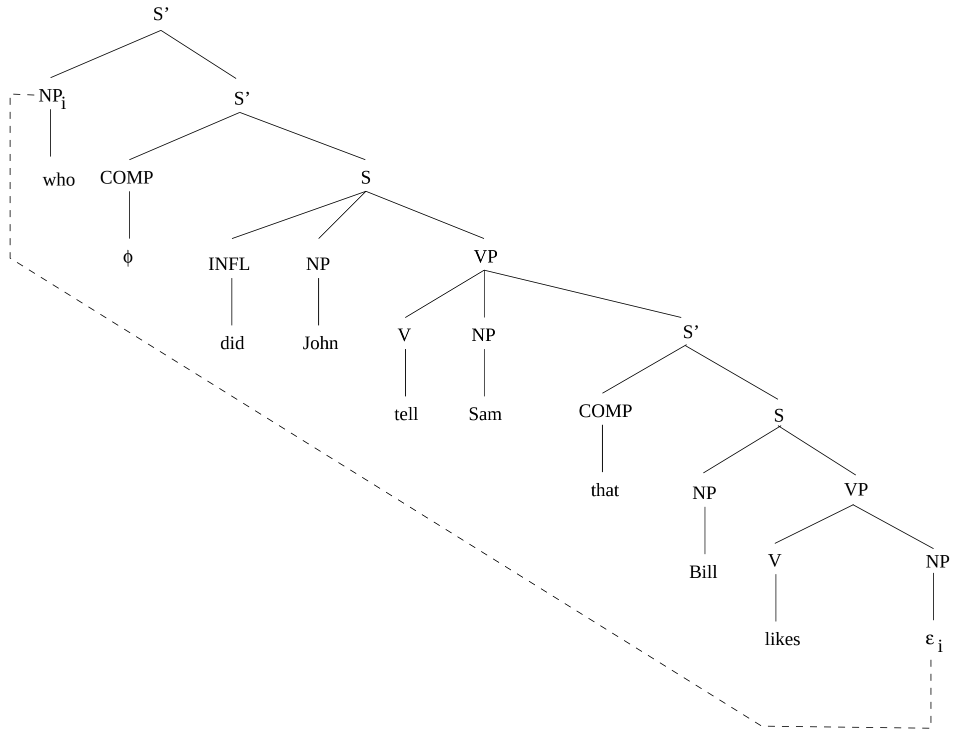

Figura 2.10: Dependencia de larga distancia entre who_i y ϵ_i

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:

$$A_1 \rightarrow A_2 A_2$$

$$A_2 \rightarrow A_1 A_2$$

$$A_2 \rightarrow A_2 A_1$$

$$A_2 \rightarrow a$$

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. ¶

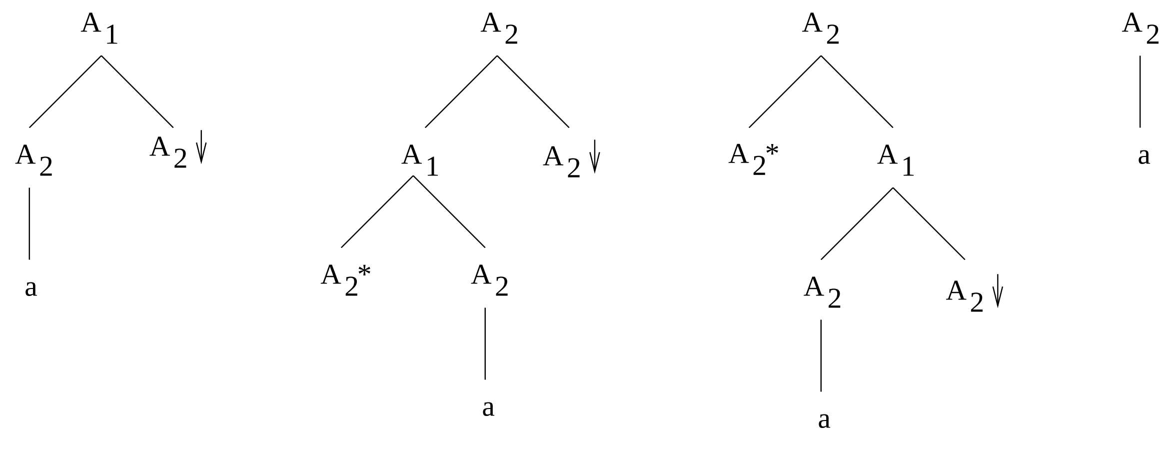

Figura 2.11: Gramática de inserción de árboles

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. ¶

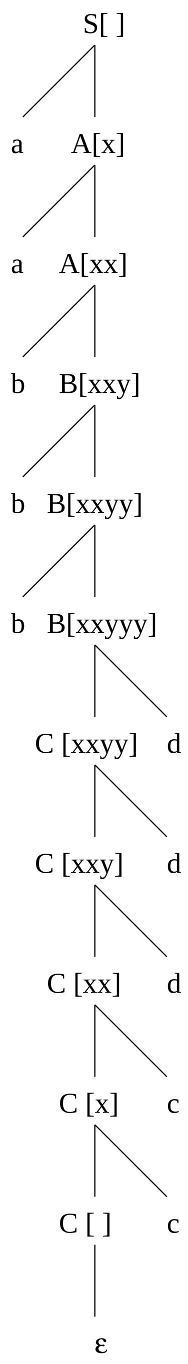

Figura 2.12: Árbol derivado en LIG para la cadena *aabbbccddd*

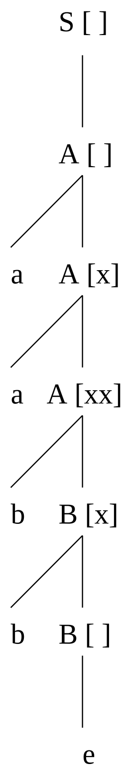

Figura 2.13: Árbol derivado en LIG para la cadena *aabbe*

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*

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

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*

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

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

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

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:

$$\begin{gathered} A_{r,0}[\circ\circ] \to A'_{r,0}[\circ\circ\gamma_r] \\ A'_{r,0}[\circ\circ\gamma_r] \to A''_{r,0}[\circ\circ]\ A_{r,1}[\,] \dots A_{r,m}[\,] \\ A''_{r,0}[\,] \to \epsilon \end{gathered}$$

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

$$\begin{gathered} S[\circ\circ] \to c\ C[\circ\circ\gamma]\ e \\ C[\circ\circ] \to c\ C[\circ\circ\gamma]\ e \\ C[\circ\circ] \to D[\circ\circ] \\ D[\circ\circ\gamma] \to d\ D[\circ\circ] \\ D[\,] \to \epsilon \end{gathered}$$

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

$$\begin{gathered} S'[\circ\circ] \to A[\,]\ S[\circ\circ] \\ S[\circ\circ] \to c\ C[\circ\circ\gamma]\ e \\ C[\circ\circ] \to c\ C[\circ\circ\gamma]\ e \\ C[\circ\circ] \to D[\circ\circ] \\ D[\circ\circ\gamma] \to d\ D[\circ\circ] \\ D[\,] \to \epsilon \\ A[\,] \to a\ A[\,]\ b \\ A[\,] \to \epsilon \end{gathered}$$

¶

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

$$\langle \beta_1, 2.1\rangle^t[\circ\circ] \rightarrow \langle \beta_1, 2.1\rangle^b[\circ\circ]$$
$$\langle \beta_1, 2.1\rangle^b[\circ\circ\langle \alpha, 2\rangle] \rightarrow \langle \alpha, 2\rangle^b[\circ\circ]$$
$$\langle \beta_1, 2.1\rangle^b[\circ\circ\langle \beta_1, 2\rangle] \rightarrow \langle \beta_1, 2\rangle^b[\circ\circ]$$
$$\langle \beta_1, 2.2\rangle^t[\circ\circ] \rightarrow \langle \beta_1, 2.2\rangle^b[\circ\circ]$$
$$\langle \beta_1, 2.2\rangle^b[\,] \rightarrow c$$

- al árbol β_2:

$$\langle \beta_2, 0\rangle^t[\circ\circ] \rightarrow \langle \beta_2, 0\rangle^b[\circ\circ]$$
$$\langle \beta_2, 0\rangle^b[\circ\circ] \rightarrow \langle \beta_2, 1\rangle^t[\circ\circ]$$
$$\langle \beta_2, 1\rangle^t[\circ\circ] \rightarrow \langle \beta_3, 0\rangle^t[\circ\circ\langle \beta_2, 1\rangle]$$
$$\langle \beta_2, 1\rangle^b[\circ\circ] \rightarrow \langle \beta_2, 1.1\rangle^t[\circ\circ]$$
$$\langle \beta_2, 1.1\rangle^t[\circ\circ] \rightarrow \langle \beta_2, 1.1\rangle^b[\circ\circ]$$
$$\langle \beta_2, 1.1\rangle^b[\circ\circ\langle \alpha, 2\rangle] \rightarrow \langle \alpha, 2\rangle^b[\circ\circ]$$
$$\langle \beta_2, 1.1\rangle^b[\circ\circ\langle \beta_1, 2\rangle] \rightarrow \langle \beta_1, 2\rangle^b[\circ\circ]$$

- y al árbol β_3:

$$\langle \beta_3, 0\rangle^t[\circ\circ] \rightarrow \langle \beta_3, 0\rangle^b[\circ\circ]$$
$$\langle \beta_3, 0\rangle^b[\circ\circ] \rightarrow \langle \beta_3, 1\rangle^t[\,] \;\; \langle \beta_3, 2\rangle^t[\circ\circ]$$
$$\langle \beta_3, 1\rangle^t[\circ\circ] \rightarrow \langle \beta_3, 1\rangle^b[\circ\circ]$$
$$\langle \beta_3, 1\rangle^b[\,] \rightarrow b$$
$$\langle \beta_3, 2\rangle^t[\circ\circ] \rightarrow \langle \beta_3, 0\rangle^t[\circ\circ\langle \beta_3, 2\rangle]$$
$$\langle \beta_3, 2\rangle^t[\circ\circ] \rightarrow \langle \beta_3, 2\rangle^b[\circ\circ]$$
$$\langle \beta_3, 2\rangle^b[\circ\circ] \rightarrow \langle \beta_3, 2.1\rangle^t[\circ\circ] \;\; \langle \beta_3, 2.2\rangle^t[\,]$$
$$\langle \beta_3, 2.1\rangle^t[\circ\circ] \rightarrow \langle \beta_3, 2.1\rangle^b[\circ\circ]$$
$$\langle \beta_3, 2.1\rangle^b[\circ\circ\langle \beta_3, 2\rangle] \rightarrow \langle \beta_3, 2\rangle^b[\circ\circ]$$
$$\langle \beta_3, 2.2\rangle^t[\circ\circ] \rightarrow \langle \beta_3, 2.2\rangle^b[\circ\circ]$$
$$\langle \beta_3, 2.2\rangle^b[\,] \rightarrow d$$

¶

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 $\{a_1^n \dots a_k^n \mid k \geq 1, n \geq 0\}$.

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:
  $$(x/y)\ (y|_1 z_1|_2 \ldots |_m z_m) \rightarrow (x|_1 z_1|_2 \ldots |_m z_m)$$
  - *Reglas hacia atrás* de la forma
  $$(y|_1 z_1|_2 \ldots |_m z_m)\ (x\backslash y)\ \rightarrow (x|_1 z_1|_2 \ldots |_m z_m)$$

  donde $m \geq 0$, $x, y, z_1, \ldots, z_m$ son metavariables de categorías y $|_1, \ldots, |_m \in \{\backslash, /\}$. En dichas reglas (x/y) y $(x\backslash y)$ reciben el nombre de *constituyentes primarios* mientras que $(y|_1 z_1|_2 \ldots |_m z_m)$ 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 $S \stackrel{*}{\Rightarrow} w$.

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:

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

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.

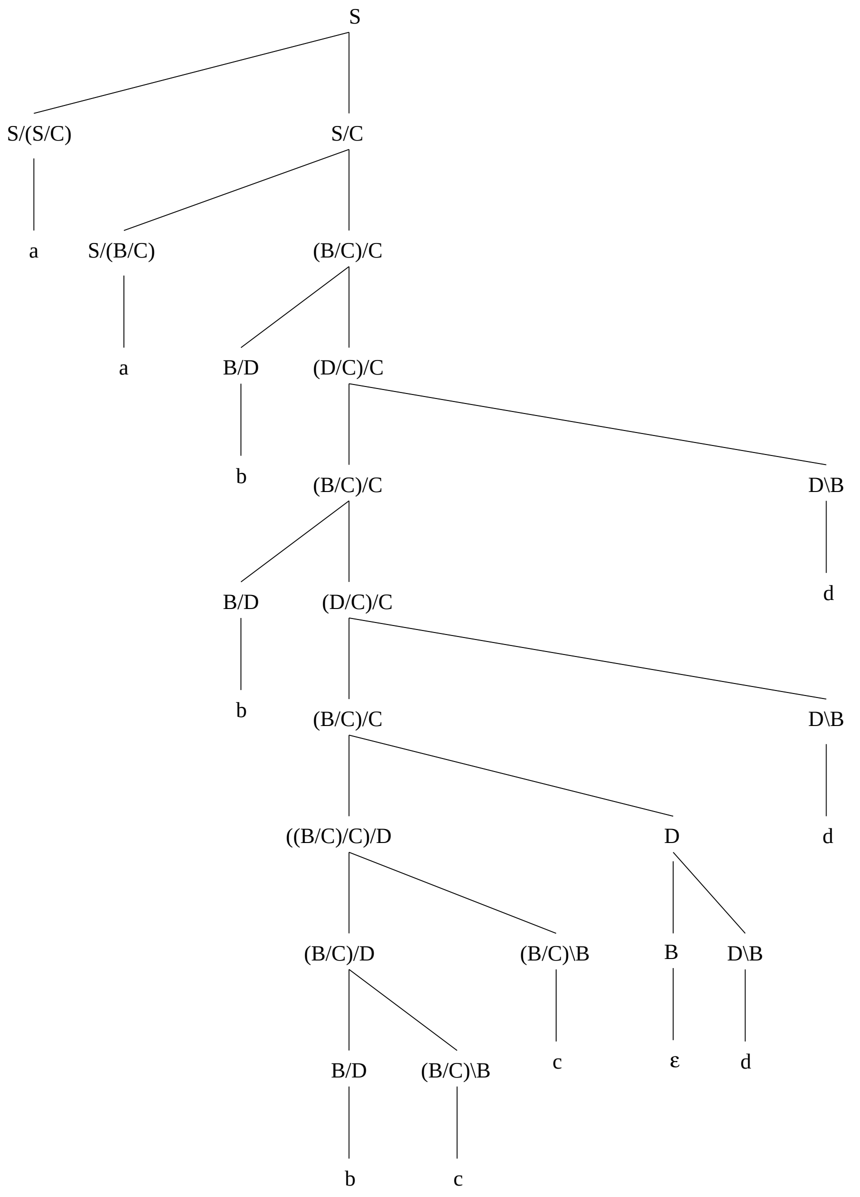

Figura 2.14: Árbol derivado en CCG para la cadena *aabbbccddd*

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 $A \to f(\sigma_1, \ldots, \sigma_m)$, con $A \in V_N$, $m \geq 1$, $f \in \{W, C_{1,m}, C_{2,m}, \ldots, C_{m,m}\}$, $\sigma_1, \ldots, \sigma_m \in V_N \cup (V_T^* \times V_T^*)$ 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:

  $$C_{i,m}(u_{1\uparrow}v_1, \ldots, u_{i\uparrow}v_i, \ldots, u_{m\uparrow}v_m) = u_1v_1 \ldots u_{i\uparrow}v_i \ldots u_mv_m$$

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

  $$W(u_{1\uparrow}v_1, u_{2\uparrow}v_2) = u_1u_{2\uparrow}v_2v_1$$

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 $A \to f(\sigma_1, \ldots, \sigma_m) \in P$ entonces $A \overset{k}{\Rightarrow} f(u_{1\uparrow}v_1, \ldots, u_{m\uparrow}v_m)$, 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 $\{a^nb^mc^nd^m\}$ 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:

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

¶

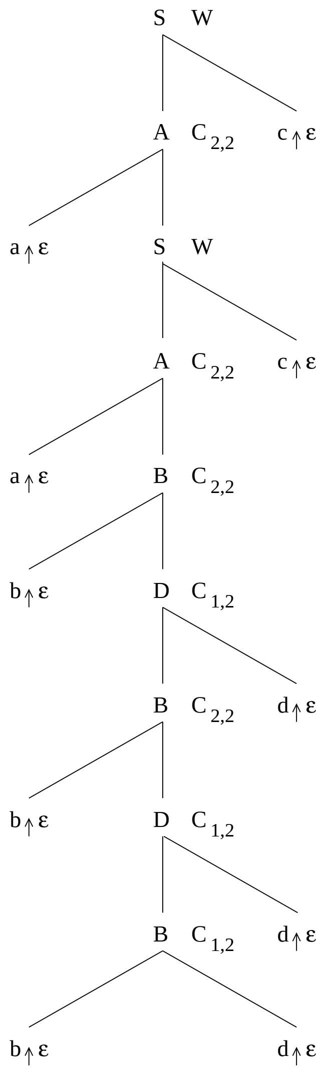

Figura 2.15: Árbol de derivación en HG para la cadena *aabbbccddd*

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 $\stackrel{*}{\Rightarrow}$ el cierre reflexivo y transitivo de $\Rightarrow$. El lenguaje $L(G)$ generado por G es el conjunto de matrices recurrentes r tal que $S \stackrel{*}{\Rightarrow} r$.

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*. ¶

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

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

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

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

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

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

$$\begin{array}{|c|}\hline \\ S \\ \hline\end{array} \Rightarrow \begin{array}{|c|c|c|c|}\hline a & & b & \\ \cline{1-1}\cline{3-3} c & A & b & B \\ \hline\end{array} \Rightarrow \begin{array}{|c|c|c|c|c|}\hline a & a & & b & \\ \cline{1-2}\cline{4-4} c & c & A & d & B \\ \hline\end{array} \Rightarrow \begin{array}{|c|c|c|c|c|}\hline a & a & \epsilon & b & \\ \cline{1-4} a & c & \epsilon & d & B \\ \hline\end{array}$$

$$\Rightarrow \begin{array}{|c|c|c|c|c|c|}\hline a & a & \epsilon & b & b & \\ \cline{1-5} c & c & \epsilon & d & d & B \\ \hline\end{array} \Rightarrow \begin{array}{|c|c|c|c|c|c|c|}\hline a & a & \epsilon & b & b & b & \\ \cline{1-6} c & c & \epsilon & d & d & d & B \\ \hline\end{array} \Rightarrow \begin{array}{|c|c|c|c|c|c|c|}\hline a & a & \epsilon & b & b & b & \epsilon \\ \hline c & c & \epsilon & d & d & d & \epsilon \\ \hline\end{array}$$

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

$$\psi_0 \rightarrow \psi_1 \ldots \psi_m$$

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:

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

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

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

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

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

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 $(k_1, \ldots, k_r) \to (\alpha_1, \ldots, \alpha_r)$ 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 $(k_1, \ldots, k_r) \to (\alpha_1, \ldots, \alpha_r) \in P$ 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:

$$(S) \to (aAbBcCdD)$$

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

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

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

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

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

  $$l : A_0 \rightarrow A_1 A_2 \dots \bar{A}_i \dots A_m$$

  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 $\langle Y_1, w_1 \rangle \dots \langle Y_m, w_m \rangle$, 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 $l : A \rightarrow A_1 A_2 \dots \bar{A}_i \dots A_m$ es una producción de G_1 entonces

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

El lenguaje generado por G es el conjunto de cadenas $a_1 \dots a_n \in V_T^*$ tal que $\langle S, \epsilon \rangle \overset{*}{\Rightarrow} \langle a_1, w_1 \rangle \dots \langle a_n, w_n \rangle$ 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:

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

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

$$l_3 : A \to \bar{B}$$

$$l_4 : B \to b\bar{B}$$

$$l_5 : B \to \bar{D}$$

$$l_6 : D \to \bar{D}d$$

$$l_7 : D \to \bar{C}$$

$$l_8 : C \to \bar{C}c$$

$$l_9 : C \to \epsilon$$

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_1 l_2\rangle \\
& \Rightarrow \langle a, \epsilon\rangle\langle a, \epsilon\rangle\langle A, l_1 l_2 l_2\rangle \\
& \Rightarrow \langle a, \epsilon\rangle\langle a, \epsilon\rangle\langle B, l_1 l_2 l_2 l_3\rangle \\
& \Rightarrow \langle a, \epsilon\rangle\langle a, \epsilon\rangle\langle b, \epsilon\rangle\langle B, l_1 l_2 l_2 l_3 l_4\rangle \\
& \Rightarrow \langle a, \epsilon\rangle\langle a, \epsilon\rangle\langle b, \epsilon\rangle\langle b, \epsilon\rangle\langle B, l_1 l_2 l_2 l_3 l_4 l_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_1 l_2 l_2 l_3 l_4 l_4 l_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_1 l_2 l_2 l_3 l_4 l_4 l_4 l_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_1 l_2 l_2 l_3 l_4 l_4 l_4 l_5 l_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_1 l_2 l_2 l_3 l_4 l_4 l_4 l_5 l_6 l_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_1 l_2 l_2 l_3 l_4 l_4 l_4 l_5 l_6 l_6 l_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_1 l_2 l_2 l_3 l_4 l_4 l_4 l_5 l_6 l_6 l_6 l_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_1 l_2 l_2 l_3 l_4 l_4 l_4 l_5 l_6 l_6 l_6 l_7 l_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_1 l_2 l_2 l_3 l_4 l_4 l_4 l_5 l_6 l_6 l_6 l_7 l_8 l_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_1 l_2 l_2 l_3 l_4 l_4 l_4 l_5 l_6 l_6 l_6 l_7 l_8 l_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_1 l_2 l_2 l_3 l_4 l_4 l_4 l_5 l_6 l_6 l_6 l_7 l_8 l_8 l_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.