Capítulo 1

Introducción

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

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

1.1. El lenguaje natural

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

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

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

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

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

1.2. La programación dinámica en el análisis sintáctico

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

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

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

[2] Un ejemplo típico consiste en buscar la distancia mínima entre dos puntos de un grafo.

[3] Dividir un problema complejo en varios más simples que son útiles para la solución del problema original.

[4] Un problema con estas características lo constituye la búsqueda dicotómica.

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

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

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

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

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

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

1.3. Formalismos gramaticales

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

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

El conjunto de los lenguajes generados por las gramáticas de adjunción de árboles constituye la clase de los lenguajes de adjunción de árboles. Esta clase también puede ser generada por

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

1.4. Ámbito de la tesis

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

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

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

En relación a los compiladores y procesadores del lenguaje:

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

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

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

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

1.5. Estructura de la memoria

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

Parte I. Lenguajes de adjunción de árboles

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

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

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

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

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

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

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

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

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

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

Capítulo 11. Una vez definidos los diferentes modelos de autómata, llega el momento de analizarlos conjuntamente, percibiéndose la existencia de tres grandes grupos de autómatas: los autómatas generales, entre los que se incluyen los autómatas lineales de índices fuertemente

dirigidos y los autómatas con dos pilas fuertemente dirigidos; los autómatas descendentes, entre los que se encuadran los autómatas a pila embebidos y los autómatas lineales de índices orientados a la izquierda; y los autómatas ascendentes, que incluyen los autómatas a pila embebidos ascendentes, los autómatas lineales de índices orientados a la derecha y los autómatas con dos pilas ascendentes.

Parte III. Apéndices

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

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

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

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

1.6. Difusión de resultados

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Los resultados de la sección 10.4 han sido publicados en:

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

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

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

cuya versión previa apareció en:

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

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

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

cuya versión previa apareció en:

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

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

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

1.7. Comunicación con el autor

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

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

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

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