Procesamiento del Lenguaje Natural, num. 29 (2002), pp.129-136 recibido 28-05-2002; aceptado 01-07-2002

Programación Dinámica y Análisis Parcial*

David Cabrero Souto
Dpto. Computación
Universidad de A Coruña
Campus de Elviña s/n
15071 A Coruña, Spain
cabrero@udc.es

Jesús Vilares Ferro
Area de Ciencias de la Computación
e Inteligencia Artificial
Universidad de Vigo
Edificio Politécnico
32004 Ourense, Spain
jvilares@uvigo.es

Manuel Vilares Ferro
Dpto. Computación
Universidad de A Coruña
Campus de Elviña s/n
15071 A Coruña, Spain
vilares@udc.es

Resumen: En los últimos años hemos observado un renovado interés en la aplicación de la programación dinámica al procesamiento del lenguaje natural (PLN). La principal ventaja es la compactación de las representaciones, lo que convierte este paradigma en un método común para el tratamiento de computaciones con un alto grado de redundancia relacionado con fenómenos como el no determinismo. El análisis sintáctico del lenguaje natural añade otro desafío, ya que a menudo la información gramatical no es suficiente. En el presente trabajo describimos una extensión de las técnicas de análisis para el caso del análisis parcial en programación dinámica. Nuestro objetivo es obtener tanta información como sea posible, esto es, análisis incompletos, al mismo tiempo que conservamos la compactación de las representaciones.
Palabras clave: Programación dinámica, análisis parcial, esquema de deducción

Abstract: The last years have seen a renewal of interest in applying dynamic programming to natural language processing. The main advantage is the compactness of the representations, which is turning this paradigm into a common way of dealing with highly redundant computations related to phenomena such as non-determinism. Natural language parsing adds another challenge, since grammatical information is often insufficient. We describe an extension of parsing techniques for partial parsing in dynamic programming. Our aim is to obtain as much information as possible, that is incomplete parses, while preserving compactness of the representations.
Keywords: Partial parsing, dynamic programming, deductive parsing scheme

1 Introducción

Las computaciones con un alto grado de redundancia son habituales cuando tratamos con formalismos gramaticales complejos. Este hecho ha motivado que las técnicas de análisis sintáctico codifiquen árboles y computaciones mediante algún tipo de estructura compartida. Un área de aplicación principal es el procesamiento del lenguaje natural, donde la programación dinámica se aplica desde hace tiempo (Earley, 1970). En particular, el análisis sintáctico del lenguaje natural presenta el problema de la información parcial. Esta falta de información se debe a errores en fases previas del análisis y al hecho de que las gramáticas y lexicones son, en la práctica, incompletos e incluso incorrectos.

Nos referiremos al análisis estándar como *análisis completo*, reservando el término *análisis parcial* para todas las subcomputaciones posibles de un análisis completo. El objetivo será obtener todos los análisis parciales correctos incluso cuando no existe un análisis completo.

Trabajos previos han ilustrado la adecuación en la práctica de la programación dinámica para el tratamiento de gramáticas independientes del contexto (GICs) (Vilares and Dion, 1994), gramáticas suavemente dependientes del contexto (Alonso et al., 1999) y gramáticas de cláusulas definidas (Vilares and Alonso, 1998). Nuestro objetivo es mostrar la validez de estos resultados para el análisis parcial. Nuestra aproximación al problema consiste en extender los modelos de análisis completo al caso del análisis parcial preservando los beneficios de la programación dinámica.

A menudo, las técnicas mencionadas mejoran la eficacia podando el espacio de búsqueda mediante un control estático. Por desgracia, cuan-

* Este trabajo ha sido parcialmente financiado por la Unión Europea, el Gobierno español y la Xunta de Galicia mediante los proyectos 1FD97-0047-C04-02, TIC2000-0370-C02-01 y PGIDT99XI10502B, respectivamente.

do tratamos análisis parciales, el control estático puede eliminar ramas de análisis que no son útiles en un análisis completo, pero son necesarias en algún análisis parcial. Para afrontar este problema trataremos las posiciones de comienzo y final de un análisis parcial como otra fuente más de no determinismo, adaptando el control estático convenientemente.

En contraste con otros trabajos en el dominio del análisis parcial, nuestra propuesta introduce un marco común basado en el concepto de esquema de deducción (Shieber, Schabes, and Pereira, 1995). Este marco diferencia claramente nuestro trabajo de otras aproximaciones orientadas a una arquitectura particular de análisis (Jacquemin, 1994; Sperber and Thiemann, 1995; Rocio and Lopes, 1998), y provee una descripción uniforme y un formalismo operacional para validar la eficacia en cada caso.

En la sección 2 de este artículo introduciremos el marco uniforme de análisis sintáctico, describiendo esquemas diferentes para el análisis completo y parcial. Estos esquemas incluyen las aproximaciones descendentes y ascendentes clásicas, y también estrategias mixtas con control dinámico y estático. La sección 3 proporciona un estudio del proceso de interpretación dinámica. En la sección 4 comparamos en la práctica los esquemas presentados, mediante resultados experimentales. Finalmente, la sección 5 es una conclusión sobre el trabajo presentado.

2 Sobre el análisis parcial

Para obtener una definición pragmática del análisis sintáctico parcial, debemos relajar las condiciones que aplicamos al concepto estándar. El concepto de gramática incluye un símbolo inicial o axioma, S, que da lugar al análisis de todas las cadenas del lenguaje generado por la gramática. En su lugar, usaremos un conjunto de símbolos iniciales, $\mathcal{S}$, siguiendo el concepto de *punto de entrada* (Jacobs, 1992), una estructura clásica en la sintaxis abstracta que permite el análisis de fragmentos de programas. Definiremos una GIC como un cuádruple $\mathcal{G} = (N, \Sigma, P, \mathcal{S})$, donde N es un conjunto finito de categorías no terminales o variables, Σ es un alfabeto finito y P es un conjunto finito de producciones independientes del contexto. Tal y como hemos mencionado, $\mathcal{S}$ es el conjunto de símbolos iniciales que dan lugar tanto a los análisis completos como a los parciales. Supondremos que $\mathcal{L}(\mathcal{G})$ es el lenguaje generado por $\mathcal{G}$, y trataremos de determinar los análisis parciales de la cadena de entrada $w_{1..n}$, de longitud n.

En concreto, discutiremos la extensión al caso del análisis parcial de los métodos de análisis independiente del contexto clásicos incluyendo las arquitecturas descendentes y ascendentes, el algoritmo de Earley (Earley, 1970) como representante de una estrategia mixta con predicción dinámica y una propuesta LR(1) como representante de una estrategia mixta con predicción estática.

En aras de una mejor exposición hemos seleccionado un marco descriptivo común, el esquema deductivo (Shieber, Schabes, and Pereira, 1995), similar a la propuesta de esquemas de análisis (Sikkel, 1993). El sistema de deducción está formado por un conjunto de *items* representativos de estados del proceso de análisis y un conjunto de pasos de deducción que se aplican sobre los items.

Como ejemplo consideraremos el lenguaje, $\mathcal{P}$, de los palíndromos no vacíos sobre el alfabeto $\Sigma = \{a, b\}$, generados por la siguiente gramática:

Palin $\rightarrow$ *a*	*Palin* $\rightarrow$ *b*
Palin $\rightarrow$ *a Palin a*	*Palin* $\rightarrow$ *b Palin b*

Nótese, por ejemplo, que aunque la cadena de entrada $aababab \notin \mathcal{P}$, contiene subcadenas que pertenecen al lenguaje, como es el caso de los árboles mostrados en la figura 1.

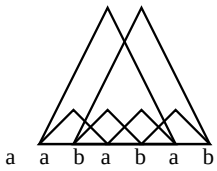

Figura 1: Análisis parciales de aababab

2.1 Un esquema descendente

El esquema para el análisis descendente completo se muestra en la figura 2. Tenemos un único *axioma* que predice el análisis del símbolo inicial, y un único *objetivo* que representa el análisis de la cadena de entrada completa. A continuación haremos una breve descripción del esquema. Los items son de la forma: $[\bullet\beta, j, T]$, indicando que hemos construido una derivación $S \stackrel{*}{\Rightarrow} w_1 \cdots w_j \beta$, y T es el árbol de análisis. El punto es una referencia a la posición j en la entrada. El análisis ha alcanzado dicha posición y debe continuar a partir de ella. Las pruebas de corrección y fiabilidad se pueden consultar en (Shieber, Schabes, and Pereira, 1995).

	Descendente	Ascendente
Items	$[\bullet\beta, j, T]$	$[\alpha\bullet, j, T]$
Invariantes	$S \overset{*}{\Rightarrow} w_1 \cdots w_j \beta$	$\alpha w_{j+1} \cdots w_n \overset{*}{\Rightarrow} w_1 \cdots w_n$
Axiomas	$[\bullet S, 0, ()]$	$[\bullet, 0, ()]$
Objetivos	$[\bullet, n, T]$	$[S\bullet, n, T]$
Scanning	$\frac{[\bullet w_{j+1}\beta, j, T_1 \text{tree}(B, \alpha \bullet w_{j+1}\beta)T_2]}{[\bullet\beta, j+1, T_1 \text{tree}(B, \alpha w_{j+1} \bullet \beta)T_2)]}$	$\frac{[\alpha\bullet, j, T]}{[\alpha w_{j+1}\bullet, j+1, T w_{j+1}]}$
Prediction	$\frac{[\bullet B\beta, j, T_1 \text{tree}(A, \alpha \bullet B\beta T_2)]}{[\bullet\gamma\beta, j, T_1 \text{tree}(A, \alpha \text{tree}(B, \bullet\gamma)\beta)T_2]}$ $\langle r_k : B \to \gamma \in R \rangle$	
Completion		$\frac{[\alpha\gamma\bullet, j, T_1 T_2]}{[\alpha B\bullet, j, T_1 \text{tree}(B, T_2)]}$ $\left\langle \begin{array}{l} r_k : B \to \gamma \in R, \\ \|\gamma\| = \|T_2\| \end{array} \right\rangle$

Figura 2: Esquemas descendente y ascendente

El análisis comienza en la posición 0 con el símbolo S, dando lugar al axioma $[\bullet S, 0, ()]$. A continuación aplicamos los siguientes pasos deductivos:

Scanning: Mueve el puntero una posición hacia adelante. Esta regla se obtiene tras observar que los items $[\bullet w_{j+1}\beta, j, T]$ y $[\bullet\beta, j+1, T w_{j+1}]$ representan el mismo estado en la derivación $S \overset{*}{\Rightarrow} w_1 \cdots w_{j+1}\beta$.

Prediction: Tomamos el siguiente símbolo no terminal a analizar, B, y lo reescribimos con la parte derecha de una producción $r_k : B \to \gamma$. Este paso predice el uso de la producción r_k.

Finalmente, $w \in \mathcal{L}(\mathcal{G})$ si y sólo si el ítem objetivo $[\bullet, n, T]$ ha sido generado. Esto significa que $S \overset{*}{\Rightarrow} w_1 \cdots w_n$, y T es el árbol de análisis.

Para adaptar este esquema al análisis parcial, consideraremos una forma modificada de los items, añadiendo una referencia a la posición de comienzo del potencial análisis parcial. Un análisis parcial cubre cualquier porción de la cadena de entrada. Por lo tanto, en lugar de un axioma comenzando en la posición 0, tendremos el siguiente conjunto de axiomas:

$$\text{Axiomas} = \{[\bullet A, i, i, ()], A \in \mathcal{S},\ 0 \leq i < n\}$$

Los pasos deductivos siguen siendo los mismos, pero manteniendo la posición de inicio. Ya que un análisis parcial pueden comprender cualquier porción de la cadena de entrada, puede comenzar y terminar en cualquier posición. Como consecuencia, tenemos que:

$$\text{Objetivos} = \{[\bullet, i, j, T], A \in \mathcal{S},\ 0 \leq j \leq n\}$$

Para el nuevo esquema, generaremos $m \times n$ axiomas, donde $m = \|\mathcal{S}\|$ y n es la longitud de la entrada. Para cada axioma generaremos una rama de análisis, y, consecuentemente, nuevos items. En nuestra gramática de ejemplo, para una cadena de entrada $aaba$ el analizador descendente completo genera 30 items, mientras que el analizador parcial genera 82.

2.2 Un esquema ascendente

Incluimos en la figura 2 el esquema para el análisis ascendente completo. Los items son de la forma $[\alpha\bullet, j, T]$, estableciendo que $\alpha w_{j+1} \cdots w_n \overset{*}{\Rightarrow} w_1 \cdots w_n$, siendo T el árbol de análisis. El punto indica que α se reduce a la subcadena de entrada hasta la posición j.

El análisis ascendente comienza en la posición 0 antes de reducir cualquier porción de la cadena de entrada. Por tanto el axioma es $[\bullet, 0, ()]$. Los pasos deductivos son:

Scanning: Desplaza el siguiente terminal y mueve el punto una posición hacia adelante.

Completion: Reduce los k_n símbolos inmediatamente anteriores al punto. Estos símbolos debe coincidir con la parte derecha de la regla r_k.

Como antes, $w \in \mathcal{L}(\mathcal{G})$ si y sólo si el ítem objetivo $[S\bullet, n, T]$ ha sido generado. Esto significa que $S \overset{*}{\Rightarrow} w_1 \cdots w_n$, donde T es el árbol de análisis.

Para el tratamiento del análisis parcial, extendemos los items con una referencia a la posición inicial. En lo que respecta a los axiomas, un análisis parcial pueden comenzar en cualquier posición de la cadena de entrada. Por lo tanto:

$$\text{Axiomas} = \{[\bullet, i, i, ()], A \in \mathcal{S},\ 0 \leq i < n\}$$

Los pasos deductivos únicamente son modificados para conservar el punto de comienzo. De nuevo, el ítem objetivo es reemplazado por un conjunto de items. Para construir este conjunto debemos tener en cuenta que un análisis parcial pueden finalizar con cualquier símbolo de $\mathcal{S}$, en cualquier posición. Así, tenemos que:

$$\text{Objetivos} = \{[A\bullet, i, j, T], A \in \mathcal{S},\ 0 \leq j \leq n\}$$

En lo que respecta a la eficiencia, la principal diferencia con el análisis completo es el conjunto de axiomas. El tamaño de este conjunto es n, la longitud de la cadena de entrada. Retomando el anterior ejemplo, esto da lugar a 29 items para el análisis completo y 54 para el análisis parcial.

2.3 Esquema de Earley

Emplearemos el algoritmo de Earley (Earley, 1970) para ilustrar la extensión al análisis parcial de una estrategia mixta con predicción dinámica. El esquema de análisis completo se muestra en la figura 3.

Items	$[A \to \alpha \bullet \beta, i, j, T],\ A \to \alpha\beta \in R$
Invariante	$S \overset{*}{\Rightarrow} w_1 \cdots w_j \beta$ $\alpha w_{j+1} \cdots w_n \overset{*}{\Rightarrow} w_i \cdots w_n$
Axiomas	$[S' \to \bullet S, 0, 0, ()]$
Objetivos	$[S' \to S\bullet, 0, n, T]$
Scanning	$\dfrac{[A \to \alpha \bullet w_{j+1}\beta, i, j, T]}{[A \to \alpha w_{j+1} \bullet \beta, i, j+1, T w_{j+1}]}$
Prediction	$\dfrac{[A \to \alpha \bullet B\beta, i, j, T]}{[B \to \bullet\gamma, j, j, ()]}\ \langle r_k : B \to \gamma \in R\rangle$
Completion	$\dfrac{[A \to \alpha \bullet B\beta, i, k, T_1][B \to \gamma\bullet, k, j, T_2]}{[A \to \alpha B \bullet \beta, i, j, T_1 \text{tree}(B, T_2)]}$

Figura 3: Esquema de *Earley*

En este caso tenemos items de la forma $[A \to \alpha \bullet \beta, i, j, T]$. De nuevo el punto es una referencia a la posición j, pero ahora el ítem representa el estado en el reconocimiento de la producción $A \to \alpha\beta$, donde α reduce una parte de la subcadena justo antes del punto, y β resta por ser analizado. En relación a i, es una referencia a la posición de comienzo de la subcadena analizada por α. Consecuentemente, los items representan un estado local en el proceso de análisis en lugar de uno global. Representar información local facilita la compartición de cálculos entre items.

La gramática es aumentada con una producción artificial $S' \to S$, donde S' es un símbolo distinguido. De esta forma se facilita la definición de axiomas y objetivos. El análisis comienza con el axioma $[S' \to \bullet S, 0, 0, ()]$, en la posición 0. Los paso deductivos son los siguientes:

Scanning: Después de reconocer el terminal w_{j+1}, movemos el puntero de la posición j a $j+1$.

Prediction: Predecimos todas las producciones $B \to \gamma$, puesto que puede reducir B desde la posición j.

Completion: Una vez finalizado el análisis de la producción $B \to \gamma$, entre las posiciones k y j, buscamos los items cuyo siguiente símbolo a analizar sea B a partir de la posición k. Para estos items generamos uno nuevo moviendo el puntero inmediatamente después de B, en la posición j.

Una vez generado el ítem objetivo$[S' \to S\bullet, 0, n, T]$, sabemos que $w_{0\ldots n}$ se reduce al símbolo inicial S, y $w \in \mathcal{L}(\mathcal{G})$.

Necesitamos modificar los axiomas y objetivos para tratar los análisis parciales. Los axiomas comenzarán con cualquier símbolo inicial en cualquier posición:

$$\text{Axiomas} = \{[S' \to \bullet A, i, i, ()], A \in \mathcal{S},\ 0 \leq i < n\}$$

y los objetivos terminan en cualquier posición después del punto de comienzo:

$$\text{Objetivos} = \{[S' \to A\bullet, i, j, T], A \in \mathcal{S},\ 0 \leq i \leq j \leq n\}$$

La comparación entre análisis completo y parcial es similar a la que hemos realizado para el esquema descendente, el número de axiomas y objetivos crece de uno a $n \times m$. Volviendo a nuestro ejemplo, este incremento significa que necesitamos 35 items para un análisis completo y 52 para uno parcial.

2.4 Una estrategia mixta con control estático

A continuación introducimos un esquema de análisis completo para un analizador tipo LR(0). La figura 4 muestra una descripción preliminar donde hemos omitido el control de estado finito para evidenciar la relación con el esquema de Earley. La diferencia entre ambos parte del significado de los items. En el caso de Earley la secuencia de símbolos α, inmediatamente a la izquierda del punto, reduce la subcadena $w_{i\ldots j}$. En LR(0), únicamente el símbolo X ($\alpha = \alpha' X$) inmediatamente a la izquierda del punto reduce la subcadena. Como consecuencia, hemos de adaptar los pasos deductivos:

Items Invariante	$[A \to \alpha \bullet \beta, i, j, T] \left\langle \begin{array}{c} A \to \alpha\beta \in R, \\ 0 \le i \le j \le n \end{array} \right\rangle$ $S \overset{*}{\Rightarrow} w_1 \cdots w_i X \beta$ $X \overset{*}{\Rightarrow} w_{i+1} \cdots w_j$ $\alpha = \alpha' X, \exists k, k \le i, \alpha' \overset{*}{\Rightarrow} w_{k+1} \cdots w_i$
Axioma	$[S' \to \bullet S, 0, 0, ()]$
Objetivo	$[S' \to S \bullet, 0, n, T]$
Shift	$\dfrac{[A \to \alpha \bullet w_j \beta, i, j, T]}{[A \to \alpha w_j \bullet \beta, j, j+1, w_j]}$
Prediction	$\dfrac{[A \to \alpha \bullet B\beta, i, j, T]}{[B \to \bullet\gamma, j, j, ()]}$
Reduce	$\dfrac{\begin{array}{l}[B \to X_1 X_2 \cdots X_m \bullet, j_{m-1}, j_m, T_m], \\ \ldots, \\ [B \to \bullet X_1 X_2 \cdots X_m, j_0, j_1, ()], \\ [A \to \alpha \bullet B\beta, i, j_0, T_0]\end{array}}{[A \to \alpha B \bullet \beta, j_0, j_m, \text{tree}(B, T_1 \ldots T_m)]}$

Figura 4: Esquema LR(0), omitiendo el control finito

Shift: Es similar al paso *scanning* en Earley. La posición de comienzo refleja el último símbolo analizado, w_j.

Reduce: Reemplaza el paso *completion* de Earley. En el esquema de Earley necesitamos un ítem $[B \to X_1 \cdots X_m \bullet, j_0, j_m, T_m]$ que establece el reconocimiento de $B \to X_1 \cdots X_m$ entre las posiciones j_0 y j_m. En LR(0) necesitamos m items de la forma $[B \to X_1 \ldots X_i \ldots Xm, j_{i-1}, j_i, T_i]$. Cada ítem restablece el reconocimiento de uno de los m símbolos de la parte derecha de la producción en posiciones consecutivas entre j_0 y j_m.

Prediction: Igual que en el esquema de Earley.

A continuación, obtendremos un esquema más eficiente añadiendo un control de estado finito (Alonso, Cabrero, and Vilares, 1998). Esta modificación implica reemplazar las producciones con punto $A \to \alpha \bullet \beta$ por un estado, st, representativo de su clase de equivalencia. Para construir el control de estado finito mencionado, inicializamos $st_0 = [S' \to \bullet S]$. Seguidamente construimos los demás estados como el cierre de sus items$[A \to \alpha \bullet B\beta]$. Más concretamente, la operación de cierre añade los items $[B \to \bullet\gamma]$ por cada producción $B \to \gamma$. El cierre es en realidad equivalente al paso de *prediction.* Por lo tanto, lo hemos eliminado del esquema de análisis LR con control de estado finito, tal y como se muestra en la figura 5. Aquí, *action(state, token)* denota una acción *shift* o *reduce* en el autómata del control de estado finito para un estado y *token* dados. De forma similar, *goto(state, variable)* recupera una acción *goto*. Finalmente, *reveals(state)* se refiere a todos aquellos estados para los cuales existe una acción *shift* o *goto* hacia *state.*

Para extender el esquema al análisis parcial, debemos permitir que el análisis comience en cualquier punto de la cadena de entrada:

$$\text{Axiomas} = \{[st_0, i, i, ()], 0 \le i \le n]\}$$

y, que termine en cualquier posición después del punto de comienzo:

$$\text{Objetivos} = \{[st_f, i, j, T], 0 \le i \le j \le n]\}$$

Necesitamos cambiar la inicialización en la construcción del control de estado finito para permitir el análisis de cualquier símbolo inicial. Así, en lugar de $st_0 = \{[S' \to \bullet S]\}$, el primer estado será $st_0 = \{[S \to \bullet A] \mid A \in \mathcal{S}\}$.

Ahora, podemos mejorar los analizadores LR(0) con un control de estado finito más eficiente. A la hora de construir sus estados, añadiremos información de los símbolos adelantados que son compatibles con las acciones. El nuevo control dará lugar a un algoritmo de análisis LR(1) o LALR(1), dependiendo de la forma en que se calculan los símbolos adelantados. A continuación, para adaptar el esquema LR(0) al control LALR(1), añadiremos precondiciones a los pasos deductivos. Las precondiciones comprobarán que los símbolos adelantados son compatibles antes de aplicar la acción correspondiente. El esquema resultante se muestra en la figura 5.

Realmente *action* y *goto* son el núcleo de la tabla del autómata LALR(1). Esta tabla cambia para un autómata LR(1), sin embargo su interpretación continúa siendo la misma. Por esta razón, podemos usar el mismo esquema para los analizadores LALR(1) y LR(1), cambiando únicamente la construcción del control de estado finito.

De nuevo, la extensión al análisis parcial implica añadir axiomas y objetivos, pero con una nueva consideración. Ya que un análisis parcial puede finalizar en cualquier posición, la operación de finalización debe ser compatible con cualquier símbolo, y no sólo con el símbolo de fin de cadena. Así el conjunto de axiomas será:

$$\text{Axiomas} = \{[st_0, \_, i, i, ()], 0 \le i \le n, A \in \mathcal{S}\}$$

y el conjunto de items objetivo:

$$\text{Objetivos} = \{[st_j, \_, i, j, T], 0 \le i \le j \le n, A \in \mathcal{S}\}$$

	LR(0)	LALR(1)
Items	$[st, i, j, T]$	$[st, b, i, j, T]$
Axiomas	$[st_0, 0, 0, ()]$	$[st_0, \$, 0, 0, ()]$
Objetivos	$[st_f, 0, n, T]$	$[st_f, \$, 0, n, T]$
Shift	$\frac{[st, i, j, T]}{[st', j, j+1, w_j]}\ \langle \text{shift}_{st'} \in \text{action}(st, w_j)\rangle$	$\frac{[st, b, i, j, T]}{[st', b, j, j+1, w_j]}\ \langle \text{shift}_{st'} \in \text{action}(st, w_j)\rangle$
Reduce	$\frac{\begin{array}{l}[st_m, j_{m-1}, j_m, T_m],\\ \ldots,\\ [st_1, j_0, j_1, ()],\\ [st_0, i, j_0, T_0]\end{array}}{[st, j_0, j_m, tree(r, T_1 \ldots T_m)]}\ \left\langle \begin{array}{l}\text{reduce}_r \in \text{action}(st_m, w_j),\\ st_i \in \text{reveals}(st_{i+1}),\\ st \in \text{goto}(st_0, \text{lhs}(r)),\\ m = \text{length}(\text{rhs}(r))\end{array} \right\rangle$	$\frac{\begin{array}{l}[st_m, b', j_{m-1}, j_m, T_m],\\ \ldots,\\ [st_1, b', j_0, j_1, ()],\\ [st_0, b, i, j_0, T_0]\end{array}}{[st, b, j_0, j_m, tree(r, T_1 \ldots T_m)]}\ \left\langle \begin{array}{l}\text{reduce}_r \in \text{action}(st_m, w_{j_m}),\\ st_i \in \text{reveals}(st_{i+1}),\\ st \in \text{goto}(st_0, \text{lhs}(r)),\\ m = \text{length}(\text{rhs}(r))\end{array} \right\rangle$

Figura 5: Esquemas LR(0) y LALR(1)

Para construir el control de estado finito podríamos emplear el concepto de símbolo comodín que sustituye cualquier terminal de la gramática. Esto puede producir un crecimiento exponencial del número de estados. En su lugar, consideraremos los símbolos de la cadena de entrada ambiguos. En cada posición, las posibles interpretaciones son el terminal original, w_i, y el símbolo de fin de cadena. La primera interpretación es compatible con la continuación del análisis completo y la segunda con la finalización de un análisis parcial.

En nuestro ejemplo, el esquema LR(0) completo necesita 35 items, mientras que el esquema parcial necesita 52. Para el esquema LALR(1) se necesitan, respectivamente, 32 y 42 items.

Items	$[A, st, i, j, T] \cup [\nabla_{r,s}, st, i, j, T]$
Axiomas	$[-, st_0, 0, 0, ()]$
Objetivos	$[S', st_f, 0, n, T]$
InitShift	$\frac{[A, st, i, j, T]}{[A_{r,1}, st', j, j+1, w_j]}\ \left\langle \begin{array}{l}A_{r,1} = w_j \wedge\\ \text{shift}_{st'} \in \text{action}(st, w_j)\end{array} \right\rangle$
Shift	$\frac{[A_{r,s}, st, i, j, T]}{[A_{r,s+1}, st', j, j+1, w_j]}\ \left\langle \begin{array}{l}A_{r,s+1} = w_j \wedge\\ \text{shift}_{st'} \in \text{action}(st, w_j)\end{array} \right\rangle$
Sel	$\frac{[A_{r,n}, st, i, j, T]}{[\nabla_{r,n}, st, j, j, ()]}\ \langle \text{reduce}_r \in \text{action}(st, w_j)\rangle$
Red	$\frac{[\nabla_{r,s}, st, k, j, T_1]\ [A_{r,s}, st, i, k, T_2]}{[\nabla_{r,s-1}, st', i, j, T_2 T_1]}\ \langle st' \in \text{reveals}(st)\rangle$
Head	$\frac{[\nabla_{r,0}, st, i, j, T]}{[A_{r,0}, st', i, j, \text{tree}(A_{r,0}, T)]}\ \langle st' \in \text{goto}(st, A_{r,0})\rangle$

Figura 6: Esquema LALR(1) con binarización implícita

3 La interpretación dinámica

Dado que las acciones del autómata dependen del primer, y posiblemente del segundo, elementos de la pila, consideraremos una binarización implícita de la gramática. Como consecuencia obtenemos dos características interesantes que no son habituales en otros algoritmos de análisis independiente del contexto:

- La complejidad temporal del analizador es $\mathcal{O}(n^3)$, donde n es la longitud de la cadena de entrada. Este resultado se logra sin necesidad de transformar la gramática a Forma Normal de Chomsky.
- La compartición de cálculos de análisis entre la cola de hijos de un nodo es posible. Más exactamente, un analizador ascendente sólo puede compartir los constituyentes de la derecha, mientras que el analizador descendente sólo puede compartir los de la izquierda. La razón es simple y se debe al tipo de búsqueda empleado para construir el bosque de análisis. La búsqueda primero en anchura da lugar a construcciones ascendentes y la búsqueda primero en profundidad, ascendentes, tal y como se muestra en la figura 7.

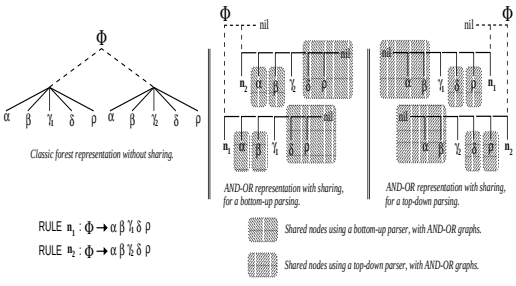

Figura 7: Compartición de la cola de hijos en un nodo

Para obtener una complejidad $\mathcal{O}(n^3)$ en el caso general, usamos una binarización implícita de las producciones. Hacemos esto dividiendo ca-

da paso de reducción, que implica m elementos, en $m+1$ pasos de reducción de producción con un máximo de 2 elementos en su parte derecha. Por tanto, la reducción de la producción $A_{r,0} \rightarrow A_{r,1} \cdots A_{r,n_r}$ se realiza de forma equivalente como la reducción de las siguientes n_r+1 producciones:

$$\begin{array}{ll} A_{r,0} \rightarrow \nabla_{r,0} & \nabla_{r,0} \rightarrow A_{r,1}\nabla_{r,1} \\ \vdots & \vdots \\ \nabla_{r,n_r-1} \rightarrow A_{r,n_r}\nabla_{r,n_r} & \nabla_{r,n_r} \rightarrow \epsilon \end{array}$$

Este tratamiento de las reducciones implica un cambio en la forma de los items. Emplearemos un nuevo elemento, representando los símbolos de una producción o el símbolo $\nabla_{r,i}$ que indica que los elementos $A_{r,i+1} \ldots A_{r,n_r}$ ya han sido reducidos[1].

Con respecto a los pasos deductivos, debemos diferenciar entre el desplazamiento del primer símbolo de la parte derecha de una producción (*InitShift*) o el desplazamiento de los otros símbolos (*Shift*).

Por su parte, el paso *Reduce* también ha sido refinado en tres pasos. La *selección* de la producción a reducir (*Sel*), la *reducción* de las producciones binarias implícitas (*Red*), y el reconocimiento del símbolo de la parte izquierda de la producción a reducir (*Head*). El esquema resultante se muestra en la figura 6. Este esquema corresponde a la interpretación dinámica de los algoritmos de análisis LR(1) o LALR(1) usando un sistema de inferencia basado en items S^1 (de la Clergerie, 1993), dependiendo de la tabla de acciones elegida.

La extensión al análisis parcial es análoga a los esquemas previos, siguiendo una aproximación idéntica para la construcción de la tabla, y añadiendo nuevos axiomas y objetivos. Así, el conjunto de axiomas viene dado por:

$$\text{Axiomas} = \{[st_0, \_, i, i, ()], 0 \leq i \leq n, A \in \mathcal{S}\}$$

y el conjunto de objetivos por:

$$\text{Objetivos} = \{[st_j, \_, i, j, T], 0 \leq i \leq j \leq n, A \in \mathcal{S}\}$$

4 Resultados experimentales

Para ilustrar los aspectos prácticos de nuestra propuesta, incluimos algunos resultados experimentales. Hemos analizado varias cadenas de entrada cuya longitud varía desde 1 a 12, considerando diversas gramáticas: una gramática de las expresiones aritméticas con y sin recursividad, la gramática de los palíndromos empleada a lo largo del presente trabajo, y una gramática para la adjunción de sintagmas preposicionales. Para parte de estas gramáticas no es posible experimentar el análisis descendente ya que presentan recursividad por la izquierda.

[1] $\nabla_{r,i}$ es equivalente a la producción con punto $A_{r,0} \rightarrow \alpha\beta$ donde $\alpha = A_{r,1} \ldots A_{r,i}$ y $\beta = A_{r,i+1} \ldots A_{r,n_r}$.

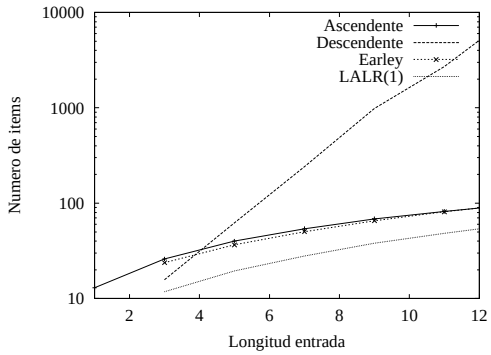

Figura 8: Num. de items en análisis completo

Dado que el número de items generados es diferente incluso para el análisis de cadenas de la misma longitud, hemos calculado la media para todas las cadenas de la misma longitud, tanto para el análisis completo como para el parcial.

En lo que respecta al análisis completo, en aquellas gramáticas donde resulta relevante, éstas parecen adecuadas para el análisis descendente, tal y como se observa en la figura 8. Por su parte, el análisis ascendente solo muestra un buen rendimiento para cadenas de entrada de poca longitud. Finalmente, las aproximaciones basadas en programación dinámica, tanto el algoritmo de Earley como la interpretación dinámica del analizador LALR(1), muestran un rendimiento similar al análisis descendente.

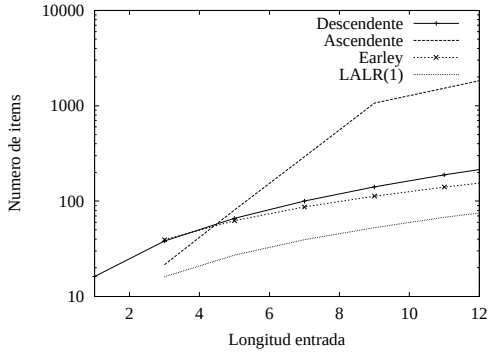

Figura 9: Num. de items en análisis parcial

En lo que atañe al análisis parcial, figura 9, el análisis descendente sufre una degradación del

rendimiento, mientras que el ascendente continúa mostrando un buen rendimiento para cadenas de entrada cortas. Las aproximaciones de programación dinámica continúan mostrando un buen comportamiento.

Finalmente, la figura 10 ilustra la relación entre los esquemas parciales y completos, sintetizando las figuras anteriores. Hemos reemplazado el número de items generados por el incremento del número de items en el análisis completo a su contrapartida en el análisis parcial. De nuevo se muestra la escasa capacidad de adaptación del esquema descendente al análisis parcial, sufriendo un crecimiento exponencial. Por su parte los esquemas basados en algún tipo de estrategia ascendente se adaptan con un incremento adecuado del coste computacional.

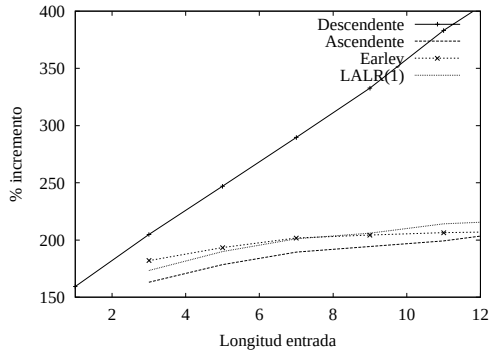

Figura 10: Relación análisis parcial/completo

5 Conclusiones

Hemos descrito una aproximación práctica al análisis parcial en el dominio de las GICs. En comparación con trabajos previos, nuestra propuesta se basa en un esquema de análisis deductivo, estableciendo un marco uniforme para comparar el rendimiento entre diferentes estrategias de análisis tanto para el caso parcial como para el completo.

Desde un punto de vista teórico, hemos presentado gradualmente cada esquema para remarcar la relación entre ellos. Esto lleva a una mejor comprensión de los mecanismos que regulan la definición de las reglas de deducción e incluso las estructuras que manipulan. La evolución del análisis completo al parcial también se desarrolla en cada caso.

6 Agradecimientos

El código necesario para interpretar los esquemas de análisis expuestos deriva del código original de (Shieber, Schabes, and Pereira, 1995), y ha sido adaptado inicialmente por V.J. Díaz Madrigal.

Bibliografía

Alonso, M.A., D. Cabrero, E. de la Clergerie, and M. Vilares. 1999. Tabular algorithms for TAG parsing. In *Proc. of EACL'99, Ninth Conference of the European Chapter of the Association for Computational Linguistics*, pages 150–157, Bergen, Norway, June. ACL.

Alonso, M.A., D. Cabrero, and M. Vilares. 1998. Construction of efficient generalized LR parsers. In Derick Wood and Sheng Yu, editors, *Automata Implementation*, volume 1436 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin-Heidelberg-New York, pages 7–24.

de la Clergerie, E. 1993. *Automates à Piles et Programmation Dynamique*. Ph.D. thesis, University of Paris VII, France.

Earley, J. 1970. An efficient context-free parsing algorithm. *Communications of the ACM*, 13(2):94–102.

Jacobs, I., 1992. *The* CENTAUR *1.2 Manual*. INRIA, Sophia-Antipolis, France.

Jacquemin, Christian. 1994. Receycling terms into a partial parser. In *Proc. of the 4^{th} Conf. on Applied Natural Language Processing. Stuttgart, DE, 13–15 Oct 1994*, pages 113–118.

Rocio, V. and J. G. Lopes. 1998. Partial parsing, deduction and tabling. In *Proceedings of Tabulation in Parsing and Deduction (TAPD'98)*, pages 52–61, Paris (FRANCE), April.

Shieber, S.M., Y. Schabes, and F.C.N. Pereira. 1995. Principles and implementation of deductive parsing. *Journal of Logic Programming*, 1-2:3–36.

Sikkel, K. 1993. *Parsing Schemata*. Ph.D. thesis, University of Twente, The Netherlands.

Sperber, Michael and Peter Thiemann. 1995. The essence of LR parsing. In *Proceedings of the ACM SIGPLAN Symposium on Partial Evaluation and Semantics-Based Program Manipulation*, pages 146–155, La Jolla, California, 21-23 June.

Vilares, M. and M.A. Alonso. 1998. An LALR extension for DCGs in dynamic programming. In Carlos Martín Vide, editor, *Mathematical and Computational Analysis of Natural Language*, volume 45 of *Studies in Functional and Structural Linguistics*. John Benjamins Publishing Company, Amsterdam & Philadelphia, pages 267–278.

Vilares, M. and B.A. Dion. 1994. Efficient incremental parsing for context-free languages. In *Proc. of the 5^{th} IEEE International Conference on Computer Languages*, pages 241–252, Toulouse, France.