Procesamiento del Lenguaje Natural, num. 30 (2003), pp.107-113 recibido 1-03-2003; aceptado 20-05-2003

Análisis sintáctico de sentencias incompletas*

Manuel Vilares Ferro y **Víctor M. Darriba Bilbao**
Departamento de Informática, Universidade de Vigo
Campus As Lagoas s/n, 32004 - Ourense
{vilares,darriba}@uvigo.es

Jesús Vilares Ferro
Departamento de Computación, Universidade da Coruña
Campus de Elviña s/n, 15071 - A Coruña
jvilares@udc.es

Resumen: Describimos un algoritmo de análisis sintáctico para gramáticas independientes del contexto (GICs), capaz de procesar entradas incompletas, incluyendo secuencias desconocidas de longitud igualmente desconocida. El analizador descrito genera como salida un bosque compartido finito que compila todos los análisis posibles de la entrada, a menudo infinitos en número. En contraste con trabajos anteriores, nuestra propuesta hace uso de técnicas avanzadas de programación dinámica que se traducen en una notable mejora del rendimiento computacional del sistema. Introducimos una construcción deductiva basada en el formalismo conocido como *parsing schemata*, lo que nos permite simplificar considerablemente la fase descriptiva.
Palabras clave: Análisis sintáctico, Sentencias incompletas, Autómatas de pila.

Abstract: We describe a context-free parsing algorithm to deal with ill-formed input, including also unknown parts of unknown length. The parser produces a finite shared-forest compiling all parses, often infinite in number. In contrast to previous works, our proposal derives profite from a finer dynamic programming construction, resulting on an improved computational behavior. We also introduce a deductive construction based on the parsing schemata formalism, which is on the advantage of simplification on the description task.
Keywords: Parsing, Incomplete sentences, Push-down automaton.

1. *Introducción*

Una cuestión clásica en el diseño de sistemas de diálogo es cómo lograr maximizar la cobertura y comprensión del lenguaje de entrada, obteniendo las interpretaciones de umbral máximo, cuando el proceso de cálculo debe ser devuelto inmediatamente ante la llegada de nuevos datos. A menudo esto es debido a que el usuario desconoce el tipo de preguntas a las que el sistema puede atender. De este modo, es aconsejable disponer de un sistema que, ante entradas ambiguas, intente adivinar una interpretación concreta más que interaccionar con el usuario. Para ello el análisis debería ser capaz de anticiparse de forma sistemática a la interacción con el usuario, en base a las expectativas del propio sistema.

Para cumplir tales requisitos, es necesario que el sistema sea capaz de analizar la entrada conforme ésta llega, incluso cuando los datos disponibles son sólo parcialmente consistentes. Dos factores están en el origen de dicho comportamiento en el caso de los interfaces hombre-máquina, tanto textuales como conversacionales. En aquéllos, el lenguaje de entrada puede aproximarse sólo de forma parcial, ya que las entradas pueden variar ampliamente respecto a la norma (Stainton, 2000) debido a fenómenos no gramaticales espontáneos. En los últimos (Stolcke, 1998), a menudo las

* Este trabajo ha sido parcialmente financiado por el Gobierno español mediante los proyectos TIC2000-0370-C02-01 y HP2001-0044, y por el Gobierno autonómico de Galicia a través de los proyectos PGIDT01PXI10506PN, PGIDIT02PXIB30501PR y PGIDIT02SIN01E.

ISSN 1135-5948

entradas pueden ser consideradas únicamente como versiones de uno de varios patrones posibles, distorsionadas debido a errores en el proceso de reconocimiento.

En este contexto, nuestro objetivo es computacional. Restringiremos los tipos de interacción a sólo aquellos necesarios para el entendimiento inmediato, empleando un modelo predictivo basado en el algoritmo de análisis de GICs propuesto por Vilares en (Vilares, 1992). Con respecto a trabajos anteriores (Tomita y Saito, 1988; Lang, 1988), nuestra propuesta nos provee de un marco de trabajo formal y un mejor comportamiento computacional.

2. El analizador estándar

Nuestro objetivo consiste en analizar una sentencia $w_{1\ldots n} = w_1 \ldots w_n$ de acuerdo con una GIC sin restricciones $\mathcal{G} = (N, \Sigma, P, S)$, donde la cadena vacía se representa como ε. Partiendo de $\mathcal{G}$ se obtiene un *autómata de pila* (AP) para el lenguaje $\mathcal{L}(\mathcal{G})$. En la práctica, se ha elegido un esquema LALR(1) generado por ICE (Vilares, 1992), si bien cualquier estrategia salto-reducción resulta adecuada. Formalmente un AP es una 7-tupla $\mathcal{A} = (\mathcal{Q}, \Sigma, \Delta, \delta, q_0, Z_0, \mathcal{Q}_f)$ donde: $\mathcal{Q}$ es el conjunto de estados, Σ el conjunto de símbolos de entrada, Δ el conjunto de símbolos de la pila, q_0 el estado inicial, Z_0 el símbolo de pila inicial, $\mathcal{Q}_f$ el conjunto de estados finales, y δ un conjunto finito de transiciones de la forma $\delta(p, X, a) \ni (q, Y)$ tales que $p, q \in \mathcal{Q}$, $a \in \Sigma \cup \{\varepsilon\}$ y $X, Y \in \Delta \cup \{\varepsilon\}$. Sea la configuración actual del AP de la forma $(p, X\alpha, ax)$, donde p es el estado actual, $X\alpha$ es el contenido de la pila con X en la cima, y ax la entrada restante donde a es la siguiente palabra a leer y $x \in \Sigma^*$. La aplicación de $\delta(p, X, a) \ni (q, Y)$ resulta en una nueva configuración $(q, Y\alpha, x)$ donde a ha sido leído, X ha sido extraído de la pila, e Y ha sido apilado.

Para obtener una complejidad polinómica, evitamos duplicar el contenido de la pila cuando surgen ambigüedades. En lugar de almacenar toda la información de la configuración, mantendremos localizada sólo aquella necesaria para la continuación del análisis. Esta se almacena en una tabla $\mathcal{I}$ de *items*, $\mathcal{I} = \{[q, X, i, j],\ q \in \mathcal{Q},\ X \in \{\varepsilon\} \cup \{\nabla_{r,s}\},\ 0 \leq i \leq j\}$; donde q es el estado actual, X es la cima de la pila, y sendas posiciones i y j delimitan la subcadena $w_{i+1} \ldots w_j$ abarcada por el último terminal reconocido en la pila o por la última producción reducida. El símbolo $\nabla_{r,s}$ indica que la parte $A_{r,s+1} \ldots A_{r,n_r}$ de la regla $A_{r,0} \rightarrow A_{r,1} \ldots A_{r,n_r}$ ha sido reconocida.

Describiremos el analizador mediante el formalismo *parsing schemata* (Sikkel, 1993). Un *parsing schema* es una terna $\langle \mathcal{I}, \mathcal{H}, \mathcal{D} \rangle$, donde $\mathcal{H} = \{[a, i, i+1],\ a = w_i\}$ es el conjunto inicial de items denominados *hipótesis* que codifica la sentencia a analizar[1], y $\mathcal{D}$ un conjunto de *pasos de deducción* que permite derivar nuevos items a partir de los ya conocidos. Los pasos de deducción son de la forma $\{\eta_1, \ldots, \eta_k \vdash \xi \,/\, conds\}$, lo que significa que si todo antecedente η_i está presente y las condiciones *conds* se satisfacen, entonces el consecuente ξ debería ser generado. En el caso de ICE, $\mathcal{D} = \mathcal{D}^{\text{Init}} \cup \mathcal{D}^{\text{Shift}} \cup \mathcal{D}^{\text{Sel}} \cup \mathcal{D}^{\text{Red}} \cup \mathcal{D}^{\text{Head}}$, donde:

$$\mathcal{D}^{\text{Shift}} = \{[q, X, i, j] \vdash [q', \varepsilon, j, j+1] \,/\, \exists\, [a, j, j+1] \in \mathcal{H},\ shift_{q'} \in action(q, a)\}$$
$$\mathcal{D}^{\text{Sel}} = \{[q, \varepsilon, i, j] \vdash [q, \nabla_{r,n_r}, j, j] \,/\, \exists\, [a, j, j+1] \in \mathcal{H},\ reduce_r \in action(q, a)\}$$
$$\mathcal{D}^{\text{Red}} = \{[q, \nabla_{r,s}, k, j][q, \varepsilon, i, k] \vdash [q', \nabla_{r,s-1}, i, j] \,/\, q' \in reveal(q)\}$$

$$\mathcal{D}^{\text{Init}} = \{\vdash [q_0, \varepsilon, 0, 0]\}$$
$$\mathcal{D}^{\text{Head}} = \{[q, \nabla_{r,0}, i, j] \vdash [q', \varepsilon, i, j] \,/\, q' \in goto(q, A_{r,0})\}$$

siendo $q_0 \in \mathcal{Q}$ el estado inicial, y donde *action* y *goto* son entradas en las tablas del AP. Decimos que $q' \in reveal(q)$ si y sólo si $\exists\ Y \in N \cup \Sigma$ tal que $shift_q \in action(q', Y)$ ó $q \in goto(q', Y)$, es decir, cuando existe una transición de q' a q en $\mathcal{A}$. Este conjunto equivale a la interpretación dinámica de APs no deterministas:

- El inicio del proceso de análisis viene dado por un paso de deducción *Init*.
- Un paso *Shift* se corresponde con la introducción de un terminal a en la cima de la pila cuando la acción a realizar es un salto a un estado st'.
- Un paso *Sel* se corresponde con la introducción de un símbolo ∇_{r,n_r} en la cima de la pila con objeto de iniciar la reducción de una regla r.
- La reducción de una regla de longitud $n_r > 0$ es realizada mediante una

[1] La cadena vacía, ε, se representa por el conjunto de hipótesis vacío, $\emptyset$. Una cadena de entrada $w_{1\ldots n}$, $n \geq 1$ se representa por $\{[w_1, 0, 1],\ [w_2, 1, 2],\ \ldots,\ [w_n, n-1, n]\}$.

serie de n_r pasos *Red*, en el que cada uno de los cuales se corresponde con una transición que reemplaza los dos elementos $\nabla_{r,s}\, X_{r,s}$ situados en la cima de la pila por el elemento $\nabla_{r,s-1}$.

- La reducción de una regla r finaliza mediante un paso *Head* correspondiente a una transición que reconoce el elemento cima $\nabla_{r,0}$ como equivalente a la parte izquierda $A_{r,0}$ de dicha regla, realizando el correspondiente cambio de estado.

Estos pasos son aplicados repetidamente hasta que no se pueden generar nuevos items. La división de las reducciones en una serie de pasos *Red* permite la compartición de cálculos correspondientes a reducciones parciales, en el peor caso con una complejidad temporal (resp. espacial) $\mathcal{O}(n^3)$ (resp. $\mathcal{O}(n^2)$) en relación a la longitud n de la cadena de entrada (Vilares, 1992). La cadena de entrada es reconocida si y sólo si el ítem final $[q_f, \nabla_{0,0}, 0, n+1]$, $q_f \in \mathcal{Q}_f$, es generado.

3. *El análisis de sentencias incompletas*

El alfabeto de entrada ha sido extendido para poder manejar sentencias incompletas. De acuerdo con Lang (Lang, 1988), dos nuevos símbolos han sido introducidos: "?", representando un único símbolo de palabra desconocida, y "$*$", representando una secuencia desconocida de palabras de entrada. Una vez que el analizador sintáctico detecta que el siguiente símbolo de entrada es uno de dichos símbolos extra, se aplicará un conjunto de pasos de deducción en modo que denominaremos *incompleto*, $\mathcal{D}_{\text{incomplete}}$, que incluyen las dos siguientes hipótesis:

$$\mathcal{D}^{\text{Shift}}_{\text{incomplete}} = \{[q,\varepsilon,i,j] \vdash [q',a,j,j+1] \;/\; w_{j+1} = ?, \mathit{shift}_{q'} \in \mathit{action}(q,a)\}$$
$$\mathcal{D}^{\text{Loop\_shift}}_{\text{incomplete}} = \{[q,\varepsilon,i,j] \vdash [q',X,j,j] \;/\; w_{j+1} = *, \mathit{shift}_{q'} \in \mathit{action}(q,X)\}$$

a la vez que se mantienen los pasos de deducción siguientes:

$$\mathcal{D}^{\text{Init}}_{\text{incomplete}} = \mathcal{D}^{\text{Init}}, \quad \mathcal{D}^{\text{Sel}}_{\text{incomplete}} = \mathcal{D}^{\text{Sel}},$$
$$\mathcal{D}^{\text{Red}}_{\text{incomplete}} = \mathcal{D}^{\text{Red}}, \quad \mathcal{D}^{\text{Head}}_{\text{incomplete}} = \mathcal{D}^{\text{Head}}$$

Desde un punto de vista intuitivo, los pasos en $\mathcal{D}^{\text{Shift}}_{\text{incomplete}}$ aplican cualquier transición de salto independientemente del símbolo de preanálisis actual, a condición de que la transición sea aplicable con respecto a la configuración del AP y de que el siguiente símbolo de entrada sea una palabra desconocida. En relación a $\mathcal{D}^{\text{Loop\_shift}}_{\text{incomplete}}$, se aplica a items correspondientes a configuraciones del AP para las cuales el siguiente símbolo de entrada denota una secuencia desconocida de palabras, lo que se podría traducir en cualquier acción de salto sobre terminales o variables. Dado que en este último caso se crean nuevos items en el propio conjunto inicial, las transiciones de salto pueden aplicarse cualesquier número de veces al mismo proceso de cálculo, sin tener que leer nuevos símbolos de la cadena de entrada.

Los pasos en $\mathcal{D}_{\text{incomplete}}$ se aplican hasta que una rama de análisis pueda enlazar con el contexto derecho mediante una acción de salto, volviendo al modo de análisis estándar. En este proceso, cuando tratamos con secuencias de palabras desconocidas, podemos generar nodos que únicamente deriven símbolos "$*$". De este modo, trabajos anteriores (Lang, 1988) incluyen una fase complementaria de simplificación donde estas variables son reemplazadas por el terminal de secuencia desconocida, "$*$". Además, los autores no detallan cómo tratar tales nodos cuando son generados a partir de más de una rama de análisis, situación habitual en un ámbito no determinista. Tal sobregeneración no tiene interés en la mayoría de aplicaciones prácticas e introduce un coste computacional adicional, lo que supone una pérdida extra de eficiencia en el análisis.

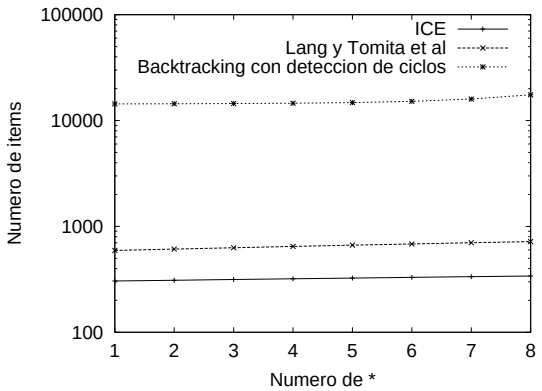

Figura 1: Número de items en modo incompleto para el ejemplo de *nombres*.

En nuestro caso ambos problemas son eliminados, tanto en lo referente a la eliminación de la fase extra de simplificación

como a la sobregeneración de secuencias desconocidas, simplemente extendiendo la estructura de los items para considerar un contador de inserciones que tabula el número de categorías sintácticas y léxicas usadas en la reconstrucción de la sentencia incompleta. Cuando se generen varios items que representen un mismo nodo, sólo se almacenarán aquellos cuyo número de inserciones es mínimo, eliminando el resto, podándolos del bosque compartido que constituye la salida del análisis.

Formalmente, los items extendidos mediante contadores de error, e, son de la forma $[p, X, i, j, \mathrm{e}]$ y, para tratarlos, debemos redifinir el conjunto de pasos de deducción $\mathcal{D}_{\text{incomplete}}$ de la siguiente forma:

$$\mathcal{D}^{\text{Shift}}_{\text{incomplete}} = \{[q, \varepsilon, i, j, e] \vdash [q', a, j, j+1, e + I(a)] \\ / w_{j+1} = ?, \mathit{shift}_{q'} \in \mathit{action}(q, a)\}$$

$$\mathcal{D}^{\text{Loop\_shift}}_{\text{incomplete}} = \{[q, \varepsilon, i, j, e] \vdash [q', X, j, j, e + I(X)] \\ / w_{j+1} = *, \mathit{shift}_{q'} \in \mathit{action}(q, X)\}$$

$$\mathcal{D}^{\text{Red}}_{\text{incomplete}} = \{[q, \nabla_{r,s}, k, j, e][q, \varepsilon, i, k, e'] \\ \vdash [q', \nabla_{r,s-1}, i, j, e + e'] / q' \in \mathit{reveal}(q)\}$$

donde $I(X)$ es el coste de inserción para $X \in N \cup \Sigma$, manteniendo el dominio de definición considerado previamente para $\mathcal{D}^{\text{Init}}_{\text{incomplete}}$, $\mathcal{D}^{\text{Sel}}_{\text{incomplete}}$ y $\mathcal{D}^{\text{Head}}_{\text{incomplete}}$.

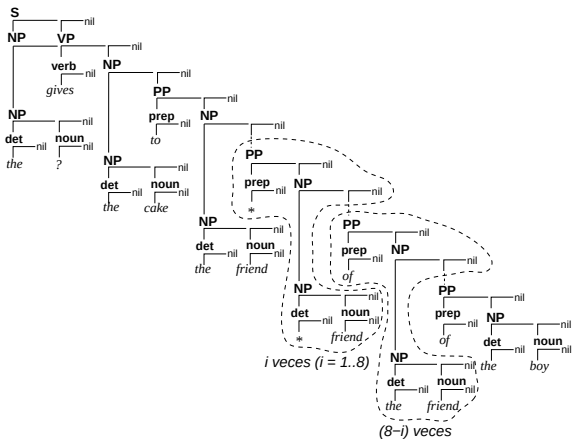

Figura 2: Bosque compartido en modo incompleto para el ejemplo de *nombres*.

Ambos, Tomita *et al.* (Tomita y Saito, 1988) y Lang (Lang, 1988), aplican técnicas de programación dinámica para manejar el no determinismo y así reducir la complejidad espacial y mejorar la eficiencia de los cálculos. Sin embargo, la aproximación práctica varía en cada caso:

- Desde el punto de vista del formalismo descriptivo, la propuesta de Lang es una generalización de la de Tomita *et al.* En efecto, para resolver los problemas derivados de las restricciones gramaticales, Lang extiende a los APs la construcción clásica de Earley (Earley, 1970), separando la estrategia de ejecución de la implementación del intérprete. El trabajo de Tomita *et al.* puede ser interpretado como una simple especificación del trabajo de Lang para los APs de tipo LR(0).

- Desde el punto de vista del formalismo operacional, Lang introduce items como fragmentos de los cálculos posibles del AP que son independientes del contenido inicial de la pila, excepto para sus dos elementos de la cima, permitiendo la compartición parcial de fragmentos comunes en caso de ambigüedad. Esto se basa en el concepto de *entorno dinámico* para GICs (Vilares, 1992), para las cuales el mecanismo de transición es adaptado y así poder ser aplicado sobre tales items. En su propuesta, Tomita *et al.* emplean una estructura basada en un grafo compartido para representar el bosque de pilas. En cuanto a la aproximación de Lang, los grafos mejoran la eficiencia computacional a expensas del coste espacial.

- Finalmente, la representación sintáctica empleada por Tomita *et al.* es un bosque compartido clásico, mientras que Lang emplea GICs como estructura de salida. Cuando la sentencia tiene diferentes análisis posibles, el conjunto de todas las posibles cadenas de análisis se representa en forma de compartición finita mediante una GIC que genera tal conjunto, posiblemente infinito. Llegados a este punto debemos recordar que las GICs pueden representarse mediante grafos AND-OR que, para Lang, componen el grafo del bosque compartido. En dicho grafo, los nodos AND se corresponden con los nodos de un árbol de análisis usual, mientras que los nodos OR se corresponden con ambigüedades. La compartición de estructuras está representada por los nodos a los que se accede a través de más de un nodo, y que se pueden corresponder tanto con la compartición de un subárbol completo, como con

la compartición de una parte de los descendientes de un nodo dado. Ésta es una ventaja suplementaria respecto a las representaciones clásicas, como es el caso de Tomita *et al.*

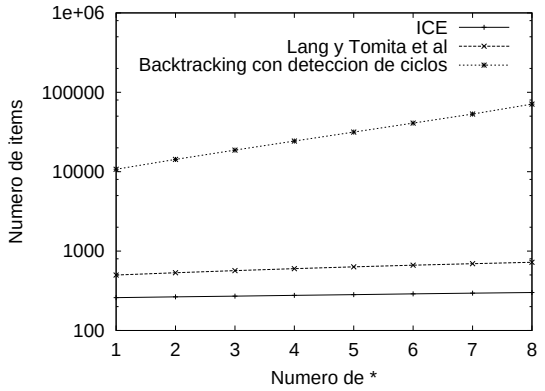

Figura 3: Número de items en modo incompleto para el ejemplo de *preposiciones*.

Nuestra propuesta emplea el formalismo descriptivo de Lang al caso particular de un esquema de análisis LALR(1), el cual facilita el cálculo del símbolo de preanálisis, manteniendo así el fenómeno de división de estados dentro de unos límites razonables. Esto nos asegura una buena compartición de los cálculos y estructuras del análisis, mejorando la eficiencia. Respecto a la estrategia de Tomita *et al.*, nuestro dominio determinista es más amplio y, consecuentemente, la complejidad temporal para el analizador es lineal sobre una clase de gramáticas también más amplia.

En lo que respecta al formalismo operacional, trabajamos sobre un entorno dinámico S^1, lo cual significa que nuestros items representan únicamente la cima de la pila. Esto implica una diferencia con respecto a la propuesta de Lang, e implícitamente respecto a la de Tomita, que usaron S^2. Desde un punto de vista práctico, el empleo de S^1 se traduce en una mejor compartición de ambos, cálculos y estructuras sintácticas, y un mejor rendimiento. Respecto a la representación del análisis considerada, empleamos la estrategia de Lang, basada en grafos AND-OR.

4. Resultados experimentales

Para ilustrar esta discusión, compararemos nuestra propuesta en base a ICE (Vilares, 1992), con los algoritmos de Lang (Lang, 1988) y Tomita *et al.* (Tomita y Saito, 1988). Como formalismo gramatical, emplearemos las siguientes reglas:

$$\begin{array}{lll} \text{S} \rightarrow \text{NP VP} & \text{NP} \rightarrow \text{det noun} & \text{VP} \rightarrow \text{verb NP} \\ \text{S} \rightarrow \text{S PP} & \text{NP} \rightarrow \text{NP PP} & \text{PP} \rightarrow \text{prep NP} \end{array}$$

para generar el lenguaje, de hecho una versión simplificada de las frases preposicionales en inglés. Las pruebas se han hecho sobre cadenas de entrada de dos tipos:

$$\text{det ? verb det noun prep det noun } \{*\,\text{noun}\}^i\ \{\text{prep det noun}\}^{8-i}\ \text{prep det noun} \quad (1)$$

$$\text{det ? verb det noun prep } \{*\text{prep}\}^i\ \{\text{det noun prep}\}^{8-i}\ \text{det noun} \quad (2)$$

donde i representa el número de palabras "$*$", es decir, el número de sentencias desconocidas en la cadena de entrada. Esto se correspondería, por ejemplo, a cadenas del tipo:

$$\text{The ? gives the cake to the friend } \{*\ \text{friend}\}^i\ \{\text{of the friend}\}^{8-i}\ \text{of the boy} \quad (3)$$

$$\text{The ? gives the cake to } \{*\ \text{of}\}^i\ \{\text{the friend of}\}^{8-i}\ \text{the boy} \quad (4)$$

respectivamente. Puesto que nuestra gramática contiene reglas "NP $\rightarrow$ NP PP" y "PP $\rightarrow$ prep NP", el número de análisis cíclicos en estas sentencias incompletas crece exponencialmente con i. Dicho número es:

$$C_0 = C_1 = 1 \quad \text{y} \quad C_i = \left(\begin{array}{c} 2i \\ i \end{array} \right) \frac{1}{i+1}, \text{ si } i > 1$$

En efecto, el analizador debe simular el análisis de un número arbitrario de palabras y, en consecuencia, ya no verse limitado por la cadena de entrada. Llegados a este punto, el analizador debería aplicar repetidamente las mismas reducciones sobre las mismas reglas gramaticales. De este modo, incluso cuando la gramática no es cíclica, la situación generada es de todo punto comparable a dicho marco de trabajo. En concreto, al tratar con sentencias incompletas, podemos derivar un no terminal de sí mismo sin acciones de lectura extra sobre la cadena de entrada. Esto nos permite evaluar nuestra propuesta en un contexto con ciclos y ampliamente ambiguo, a pesar de la simplicidad de la gramática.

Los resultados experimentales esenciales se muestran en Fig. 1 (resp. Fig. 3) en

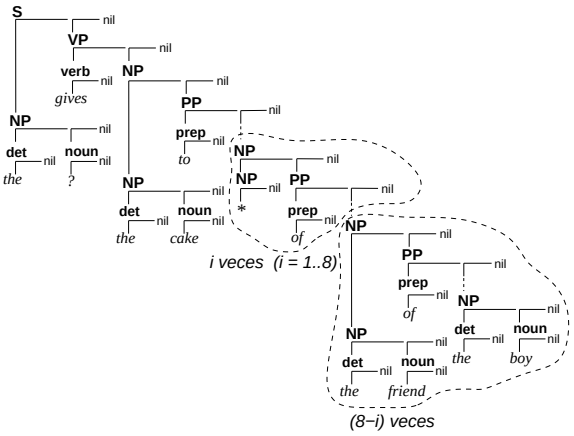

Figura 4: Bosque compartido en modo incompleto para el ejemplo de *preposiciones*.

base al ejemplo 1 (resp. ejemplo 2), para los que el bosque compartido resultante se muestra en Fig. 2 (resp. Fig. 4). Dado que para tales bosques el número de combinaciones posibles de sus árboles es exponencial, estas figuras se centran sólo en ejemplos particulares. En todos los casos hemos tomado como referencia para medir la eficiencia, el número de items generado por el sistema durante el proceso de análisis, en lugar de criterio temporal alguno, al ser éstos más dependientes de la implementación. Los bosques compartidos representados muestran la existencia de un comportamiento cíclico y de ambigüedades en el análisis.

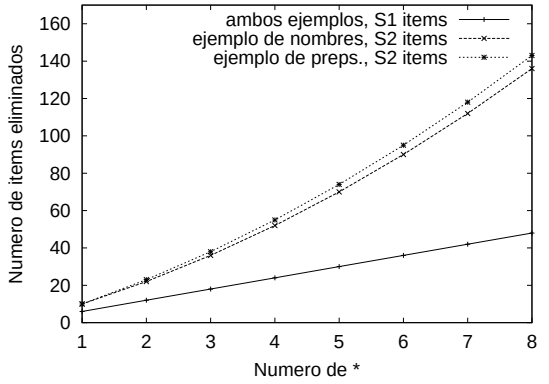

Figura 5: Items a eliminar en la fase de simplificación.

En este punto, se van a comparar tres entornos dinámicos. El clásico, S^T, es comparable a aquellos métodos de análisis basados en retroceso, que incluyan algún tipo de mecanismo para la detección de ciclos. En este caso, no es posible ningún tipo de compartición de cálculos o estructuras, por lo que sólo tiene un interés teórico. Los otros dos entornos dinámicos, S^1 and S^2, sí tienen un interés práctico real. El primero es considerado por ICE, mientras que el caso de S^2 puede identificarse, para estas pruebas, con los resultados de Lang y Tomita *et al.* Con objeto de permitir una comparación objetiva entre las diferentes propuestas consideradas, hemos de emplear un esquema de experimentación uniforme. De este modo, aunque el algoritmo de Lang puede ser aplicado a cualquier estrategia de análisis, el algoritmo de Tomita *et al.* fue pensado originalmente para APs LR(0). Hemos adaptado, pues, ambos al caso del esquema LALR(1), el mismo empleado por ICE. Sea cual sea el caso, estos resultados ilustran el mejor rendimiento de nuestra propuesta, ICE, en relación a estrategias previas. Esto es debido a que:

- Es innecesaria una fase suplementaria de simplificación, *a posteriori*, para eliminar de la estructura de salida los nodos que deriven secuencias de cadenas desconocidas, "*".
- La consideración del entorno dinámico S^1 en lugar del entorno S^2, permite una compartición mucho más eficiente tanto de estructuras como de cálculos. Consecuentemente, el número de items generados es menor.

Para poner en evidencia el coste de la fase de simplificación empleada por Lang y Tomita *et al.*, la Fig. 5 muestra el número de items a eliminar en dicho proceso para ambos ejemplos, el de nombres y el de preposiciones. Esta estimación se incluye tanto para el caso S^2, el entorno dinámico considerado originalmente, como para el caso S^1, para lo que hemos debido adaptar previamente los métodos originales de Lang y Tomita *et al.*

5. Conclusiones

Los sistemas de diálogo deberían tener una capacidad de comprensión total de la entrada. Sin embargo, en la práctica, esto no es siempre posible con la tecnología actual, incluso restringiéndose al tratamiento de un dominio restringido. En consecuencia, la robustez es crucial a la hora de encontrar una interpretación adecuada para la entrada, y nos vemos forzados a calcular hipótesis para garantizar la interactividad en este tipo de

ámbitos. Por lo tanto, el análisis de sentencias incompletas es una tarea clave en una gran variedad de interfaces hombre-máquina, y como parte de un campo más amplio y complejo, el del análisis robusto. Este es el caso de los sistemas conversacionales, donde el lenguaje a menudo contiene ruido, tanto de origen humano como tos o tartamudeo; como mecánico, atribuible a errores en el proceso de reconocimiento.

En este contexto, nuestra propuesta permite un tratamiento mejorado del cálculo, eliminando la fase extra de simplificación de soluciones previas, beneficiándose del concepto de entorno dinámico. En particular, esto permite la compartición de cálculos y estructuras, reduciendo tanto la cantidad de datos a procesar como los costes para dicho tratamiento.

Bibliografía

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

Lang, B. 1988. Parsing incomplete sentences. En D. Vargha (ed.), editor, *COLING'88*, páginas 365–371, Budapest, Hungary. vol. 1.

Sikkel, K. 1993. *Parsing Schemata.* Ph.D. tesis, Univ. of Twente, The Netherlands.

Stainton, Robert S. 2000. The meaning of 'sentences'. *Noûs*, 34(3):441–454.

Stolcke, Andreas. 1998. Linguistic knowledge and empirical methods in speech recognition. *The AI Magazine*, 18(4):25–31.

Tomita, M. y H. Saito. 1988. Parsing noisy sentences. En *COLING'88*, páginas 561–566, Budapest, Hungary.

Vilares, Manuel 1992. *Efficient Incremental Parsing for Context-Free Languages.* Ph.D. tesis, University of Nice. ISBN 2-7261-0768-0, France.