UN MODELO ROBUSTO Y EFICIENTE PARA EL ANALISIS SINTACTICO DE LENGUAJES NATURALES MEDIANTE ARBOLES MULTIPLES VIRTUALES

José F. Quesada
Centro Informático Científico de Andalucía (CICA)
Avda. Reina Mercedes, s/n (41014) Sevilla
josefran@cica.es

This paper presents a complete framework aimed at the efficient parsing of natural languages. The theoretical background is based on three main components. First, a bidirectional and bottom-up chart uses events as the control strategy. Second, a strong mathematical model imposes top-down restrictions over the events. And third, a computational model specifically designed for the manipulation of symbolic phenomena involved in natural languages that improves the overall efficiency of the system. The results obtained by the parser in the analysis of complex linguistic phenomena such as recursivity and local and non-local dependencies, show that the parser can achieve a computational complexity in the order of O(n log(n)), where n is the lenght of the input string.

Este artículo aborda el problema de la eficiencia durante el análisis sintáctico de lenguajes naturales. Para ello se propone un marco completo que incluye una estrategia de análisis ascendente bidireccional y dirigida por eventos. Sobre el componente de análisis ascendente actúa un modelo matemático que impone fuertes restricciones descendentes sobre los eventos de análisis. La experiencia en el diseño de otros analizadores nos permite concluir que las características de la implementación misma, es decir, las técnicas propias de ingeniería del software utilizadas, condicionan enormemente la eficiencia del parser. Por ello, junto a los fundamentos lingüísticos, matemáticos y de teoría de lenguajes formales, cobra especial importancia el componente computacional. En esta línea, se propone una nueva estructura, que se denominará árbol múltiple virtual (mvt). El análisis estadístico de los resultados obtenidos por este parser para el análisis de algunos de los fenómenos sintácticamente más complejos que aparecen en los lenguajes naturales (recursividad, dependencia local y dependencia no local) muestran un nivel de complejidad computacional del orden $O(n\ log(n))$, donde n es la longitud de la cadena de entrada.

1.- Chart Bidireccional Ascendente con Predicciones Descendentes Dirigido por Eventos y basado en Arboles Múltiples Virtuales

Este artículo aborda fundamentalmente el problema de la eficiencia durante el análisis sintáctico de lenguajes naturales. El estudio de la eficiencia de un sistema informático se puede realizar desde dos perspectivas: interesa conocer tanto la complejidad algorítmica del sistema como la eficiencia real de la implementación misma.

El modelo de análisis sintáctico que se propone a lo largo del artículo muestra una complejidad algorítmica del orden $O(n\ lg(n))$, donde n es el número de palabras de la cadena analizada. Este resultado se ha obtenido para el análisis de algunos de los fenómenos lingüísticos más representativos de los lenguajes naturales, como la recursividad, la dependencia local y la dependencia no local.

En cuanto a la eficiencia real, una implementación en C ha obtenido resultados que van desde 5.000 hasta 15.000 palabras analizadas por segundo.

Tradicionalmente ha existido una cierta dicotomía, en Inteligencia Artificial en general y en PLN en particular, entre eficiencia y generalidad. Para el caso del análisis sintáctico, la idea es que se garantizan mejores niveles de eficiencia en la medida en que se limiten los fenómenos lingüísticos permitidos. El objetivo del modelo de análisis sintáctico que se propone en este artículo ha sido conjugar simultáneamente la máxima generalidad lingüística necesaria en PLN con niveles de eficiencia altos.

En concreto, este modelo se puede caracterizar por los siguientes parámetros:

- La estrategia de análisis es de tipo chart bidireccional ascendente.
- Existe un componente que incorpora fuertes restricciones descendentes que reduce la sobrecarga de arcos no útiles en el chart ascendente.
- La estructura clásica de arcos de un chart es sustituida por eventos que se relacionan entre sí mediante un conjunto de nodos de enlace que funcionan como colectores-difusores de información.
- El bosque de análisis se construye como un conjunto de relaciones virtuales entre todos los nodos que intervienen. A este modelo se le ha denominado Arbol Múltiple Virtual.

2.- Tablas de Coberturas y Nodos de Enlace en un Arbol Múltiple Virtual

Sea $\mathcal{G}$ una gramática libre de contexto. Indicaremos mediante $\mathcal{P}$ al conjunto de las producciones de la gramática y mediante $\mathcal{V}$ al vocabulario (terminal y no terminal). Para cada símbolo $\alpha \in \mathcal{V}$ definimos su cobertura izquierda, derecha y media como los siguientes conjuntos de producciones:

- $CI(\alpha) = \{(\delta \rightarrow \Delta) \in \mathcal{P} : \exists\, \Omega \in \mathcal{V}^* (\Delta \equiv \alpha\, \Omega)\}$

- $\mathcal{CD}(\alpha) = \{(\delta \rightarrow \Delta) \in {}_{\mathcal{P}} : (\delta \rightarrow \Delta) \notin CI(\alpha) \wedge \exists\, \Omega \in \mathcal{V}^* (\Delta \equiv \Omega\alpha)\}$

- $\mathcal{CM}(\alpha) = \{(\delta \rightarrow \Delta) \in {}_{\mathcal{P}} : (\delta \rightarrow \Delta) \notin CI(\alpha) \wedge (\delta \rightarrow \Delta) \notin \mathcal{CD}(\alpha) \wedge \exists\, \Lambda, \Omega \in \mathcal{V}^* (\Delta \equiv \Lambda\alpha\Omega)\}$

A lo largo del artículo usaremos la gramática propuesta en Tomita 1991 como un ejemplo de aplicación del modelo. En el siguiente listado aparece esta gramática con las producciones numeradas:

```
1: S  -> NP VP
2: S  -> S PP
3: S  -> S and S
4: NP -> n
5: NP -> det n
6: NP -> NP PP
7: NP -> NP and NP
8: VP -> v NP
9: VP -> v S
10:PP -> p NP
```

En la Tabla 1 se muestran las tablas de coberturas para esta gramática.

Símbolo	CI	$\mathcal{CM}$	$\mathcal{CD}$
NP	1,6,7		7,8,10
PP			2,6

Tabla 1: Tabla de Coberturas

Símbolo	*CI*	*CM*	*CD*
S	1,3		3,9
VP			1
and		3,7	
det	5		
n	4		5
p	10		
v	8,9		

Tabla 1: Tabla de Coberturas

Como se puede observar, estas tablas indican qué producciones se dispararán ante la presencia de cad símbolo de la gramática.

Consideremos ahora la misma frase que usa Tomita:

```
I saw Jane and Jack hit the man with a telescope
```

El analizador léxico devolverá la siguiente cadena de categorías sintácticas al parser:

```
n v n and n v det n p det n
```

La primera acción del parser que proponemos incorpora un nodo de enlace (*MvtCaD: Multi Virtu Tree, Collection and Difussion*) al principio y al final de la entrada y uno más entre cada dos símbolo Por su parte, los símbolos que forman la entrada al parser son transformados en estructuras *MvtNo* (*Multi Virtual Tree, Node*). Estas estructuras se comportan como sub-componentes del Arbol Múltipl Virtual.

Así pues, para la frase considerada anteriormente, obtendríamos 12 estructuras *MvtCaD* y 11 *MvtNo* relacionadas entre sí tal y como muestra la Fig 1.

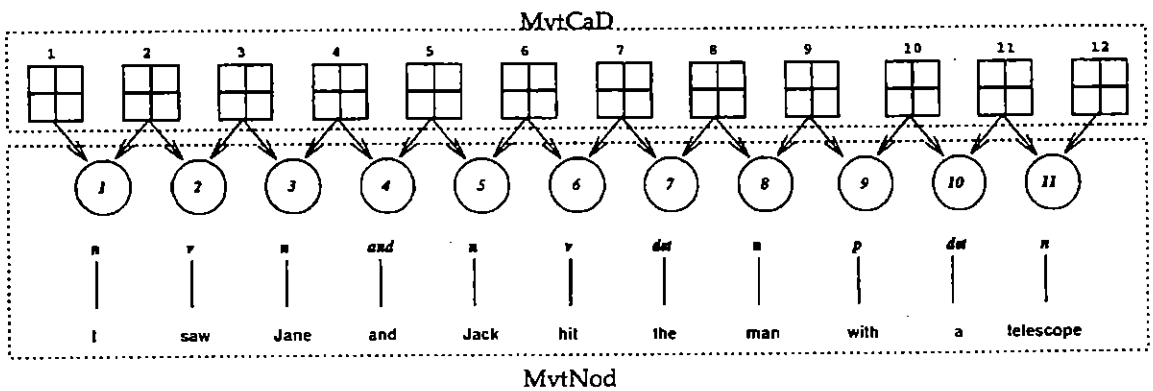

Fig 1: MvtCaD y MvtNod

A continuación, cada elemento *MvtNod* dispara un evento por cada entrada de su tabla de cobertura Así por ejemplo, la estructura *MvtNod* número 1 correspondiente a (*n:I*) dispara dos eventos, uno pa la producción número 4 y otro para la producción número 5 (las dos producciones en las que el sín bolo *n* aparece en el lado derecho y que han sido registradas en las tablas de coberturas de dicho sín bolo). Cada evento define un intervalo de aplicación, que inicialmente se corresponde con la úni estructura *MvtNod* que lo ha generado. Usando la terminología matemática habitual indicaremos q un evento es abierto o cerrado en cada uno de sus extremos en función de que la producción que lo h disparado necesite o no, respectivamente, más símbolos en dicho extremo. Por ejemplo, el even generado por la producción número 4 sobre el *MvtNod* número 1 es cerrado en ambos extremos, mie tras que el generado por la producción número 5 sobre el mismo *MvtNod* es abierto en su extrem

izquierdo (*MvtCaD* número 1) y cerrado en su extremo derecho (*MvtCaD* número 2).

Para diferenciar estas posibles situaciones, cada estructura *MvtCaD* cuenta con cuatro componentes (Fig 2).

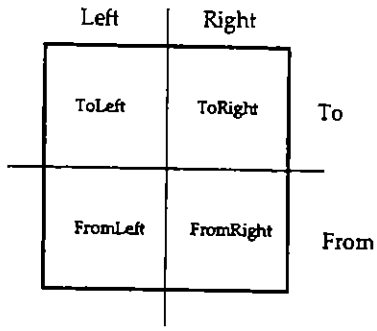

Fig 2: Estructura interna de un MvtCaD

La mitad *Left* registra los eventos generados por los *MvtNod* anclados a la izquierda del *MvtCaD*, mientras que la mitad *Right* registra los eventos generados por los *MvtNod* anclados a su derecha.

La mitad superior *To* se usa para los eventos cuyo extremo en dicho *MvtCaD* es abierto, mientras que la mitad inferior *From* para los eventos cuyo extremo es cerrado.

De esta forma, los dos primeros eventos generados por la estructura *MvtNod* número 1 entre los *MvtCaD* números 1 y 2 habrán generado 2 eventos que quedan registrados según indica la Figura 3.

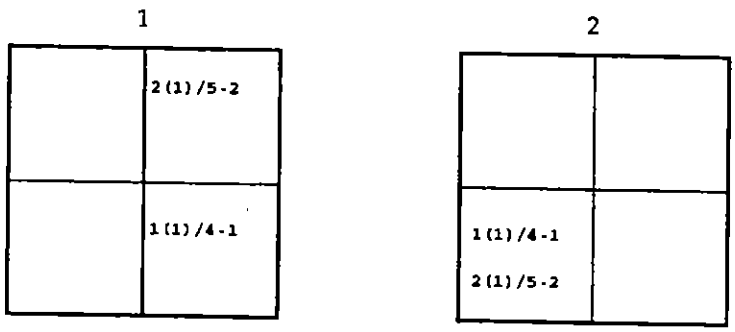

Fig 3: Anclajes de Eventos en estructuras MvtCaD

Cada referencia a un evento indica el número de evento, el componente *MvtNod* que lo ha generado, el número de producción y la posición que ocupa el extremo del evento en el lado derecho de la producción:

```
evento(MvtNod)/producción-posición
```

En la Figura 4 aparecen todos los eventos generados por los *MvtNod* iniciales y sus anclajes con los correspondientes *MvtCaD*.

Estas estructuras representan el estado inicial de un chart ascendente, y usando estas mismas estructuras podríamos proseguir con dicho algoritmo de análisis. Sin embargo, si analizamos con detalle la información que suministra el gráfico anterior y la relacionamos con un estudio detallado de la gramática de partida podemos obtener algunas conclusiones que serán de gran utilidad:

- El evento número 2 está esperando un símbolo (*det*) al principio de la cadena de análisis que no se podrá obtener, ya que dicho símbolo no se puede derivar a partir de una secuencia de producciones nulas. Por lo tanto, este evento podría ser anulado.
- El evento número 6 espera el símbolo *det* en una posición de la cadena donde hay un símbolo *v*. Teniendo en cuenta que a partir de *v* no se podrá obtener un símbolo *det*, sería conveniente prescindir del evento número 6. Algo similar sucede con el evento 10.

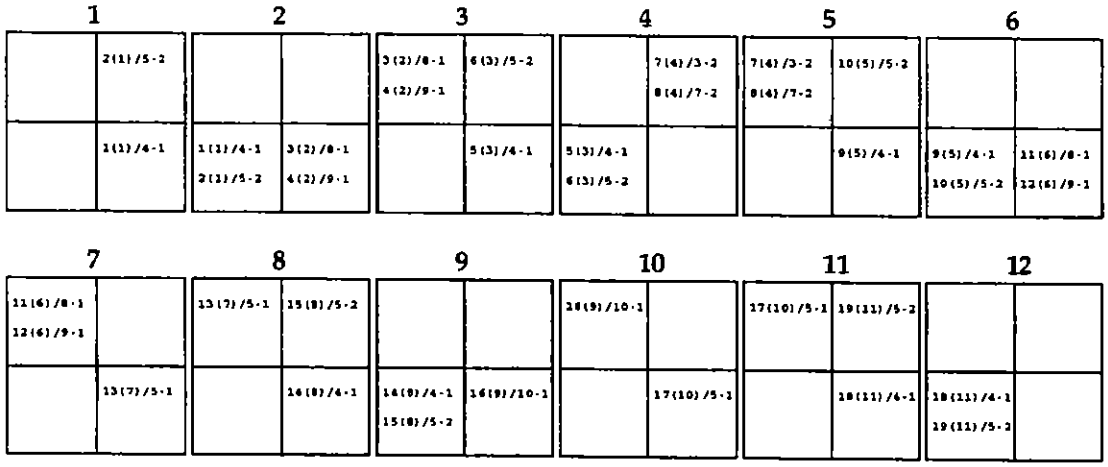

Fig 4: Estadoo Inicial de los MvtCaD

- Por su parte, el evento 14 pretende generar un símbolo *NP*. De ejecutarse este evento, el sím bolo generado no podría combinarse en ningún caso con el símbolo *det* que aparece a s izquierda. Lo mismo sucede con el evento número 18.

Estos ejemplos ilustran a un nivel informal algunas de las situaciones en que se pueden eliminar even tos incluso en las primeras fases del análisis de una entrada. La eliminación de eventos es una acció que aumenta considerablemente la eficiencia del proceso de análisis, ya que estos eventos no útile generan una considerable sobrecarga de acciones que evidentemente nunca serán de utilidad. Si embargo, la eliminación de un evento es una operación arriesgada que sólo se ha de permitir cuand se posea una certeza absoluta de su inutilidad.

Esta certeza se logra mediante dos mecanismos: en primer lugar, la estrategia bidireccional introduc toda la información disponible en las estructuras *MvtCaD*, y en segundo lugar, a partir de la gramátic se extrae un modelo matemático que contiene toda la información necesaria. En la sección siguient se definen detalladamente todos los predicados y relaciones que forman el modelo matemático.

3.- Un Modelo Matemático para la Predicción Descendente: Derivabilidad y Adyacencia

Sea $\mathcal{G}$ una gramática libre de contexto. Usaremos $\mathcal{R}$ para referirnos al conjunto de símbolos raíz de l gramática, $\mathcal{P}$ para el conjunto de las producciones de la gramática, $\mathcal{T}$ para el conjunto de símbolos ter minales, $\mathcal{N}$ para los no terminales y $\mathcal{V}$ para el vocabulario (terminal y no terminal).

Sea $\alpha, \beta \in \mathcal{V}, \Delta \in \mathcal{V}^*$:

Símbolo Raíz (root)

α es un símbolo raíz de $\mathcal{G}$, y lo representaremos mediante $\mathcal{R}(\alpha)$, sii $(\alpha \in_{\mathcal{R}})$

Símbolo Epsilon (epsilon)

α es un símbolo epsilon de $\mathcal{G}$, y lo representaremos mediante $\mathcal{E}(\alpha)$, sii $(\varepsilon \Rightarrow \alpha)$ [1].

1. Donde ε es la cadena vacía y $\Rightarrow$ es la relación de derivación ascendente en $\mathcal{G}$, la cual se define como la clausura reflexiva y transitiva de la relación de derivación directa ascendente ($\rightarrow$), que a su vez se define como:
Sean $\delta \in \mathcal{V}$, $\Delta, \Gamma, \Omega \in \mathcal{V}^*$: $\Gamma\Delta\Omega \rightarrow \Gamma\delta\Omega$ sii $(\delta \rightarrow \Delta) \in_{\mathcal{P}}$

Cadena Epsilon

Δ es una cadena epsilon de $\mathcal{G}$, y lo representaremos mediante $\mathcal{E}(\Delta)$, sii $\forall \delta \in \Delta \ (\mathcal{E}(\delta))$

Derivabilidad Parcial por la Izquierda

β es parcialmente derivable por la izquierda a partir de α, y lo representaremos mediante $\alpha \rightarrow_i \beta$, sii $\exists \Gamma, \Delta, \Omega \in \mathcal{V}^* \ (\Gamma\alpha\Delta \Rightarrow \Gamma\beta\Omega)$

Es decir, si partimos de una cadena de símbolos en la que aparece α, tras un conjunto de derivaciones ascendentes obtenemos una nueva cadena donde todos los símbolos que se encontraban a la izquierda de α en la cadena inicial se mantienen y el símbolo α ha sido substituido por β.

Definimos el conjunto de derivaciones parciales por la izquierda de un símbolo α como sigue:
$\mathcal{DPI}(\alpha) = \{\beta \in \mathcal{V} : \alpha \rightarrow_i \beta\}$

Derivabilidad Parcial por la Derecha

β es parcialmente derivable por la derecha a partir de α, y lo representaremos mediante $\alpha \rightarrow_d \beta$, sii $\exists \Gamma, \Delta, \Omega \in \mathcal{V}^* \ (\Gamma\alpha\Delta \Rightarrow \Omega\beta\Delta)$

Es decir, si partimos de una cadena de símbolos en la que aparece α, tras un conjunto de derivaciones ascendentes obtenemos una nueva cadena donde todos los símbolos que se encontraban a la derecha de α en la cadena inicial se mantienen y el símbolo α ha sido substituido por β.

Definimos el conjunto de derivaciones parciales por la derecha de un símbolo α como sigue:
$\mathcal{DPD}(\alpha) = \{\beta \in \mathcal{V} : \alpha \rightarrow_d \beta\}$

Adyacencia Primaria por la Izquierda

β es un adyacente primario por la izquierda de α, β — α, sii
$\exists \delta \in \mathcal{V}, \exists \Gamma, \Omega, \Delta \in \mathcal{V}^* \ (\delta \rightarrow \Gamma\beta\Delta\alpha\Omega \in_{\mathcal{P}} \wedge \mathcal{E}(\Delta))$

Adyacencia Primaria por la Derecha

β es un adyacente primario por la derecha de α, α — β, sii
$\exists \delta \in \mathcal{V}, \exists \Gamma, \Omega, \Delta \in \mathcal{V}^* \ (\delta \rightarrow \Gamma\alpha\Delta\beta\Omega \in_{\mathcal{P}} \wedge \mathcal{E}(\Delta))$

Adyacencia por la Izquierda

β es un adyacente por la izquierda de α, $\beta \cap \alpha$, sii $\exists \delta \in \{\beta\} \cup \mathcal{DPD}(\beta), \exists \gamma \in \{\alpha\} \cup \mathcal{DPI}(\alpha)$ tales que δ — γ

Definimos el conjunto de símbolos adyacentes por la izquierda de un símbolo α como sigue:
$\mathcal{AI}(\alpha) = \{\beta \in \mathcal{V} : \beta \cap \alpha\}$

Adyacencia por la Derecha

β es un adyacente por la derecha de α, $\alpha \cap \beta$, sii $\exists \delta \in \{\beta\} \cup \mathcal{DPI}(\beta), \exists \gamma \in \{\alpha\} \cup \mathcal{DPD}(\alpha)$ tales que γ — δ

Definimos el conjunto de símbolos adyacentes por la izquierda de un símbolo α como sigue:
$\mathcal{AD}(\alpha) = \{\beta \in \mathcal{V} : \alpha \cap \beta\}$

Símbolo Más a la Izquierda

α es un símbolo más a la izquierda, $\mathcal{SMI}(\alpha)$, sii $\exists \delta \in \mathcal{V} \ (\alpha \rightarrow_i \delta \ \wedge \ \mathcal{R}(\delta))$

Símbolo Más a la Derecha

α es un símbolo más a la derecha, $\mathcal{SMD}(\alpha)$, sii $\exists \delta \in \mathcal{V} \ (\alpha \rightarrow_d \delta \ \wedge \ \mathcal{R}(\delta))$

La Tabla 2 contiene la información que se obtiene tras aplicar este modelo matemático a la gramática presentada anteriormente. Las columnas $\mathcal{R}$(símbolo raíz) y $\mathcal{E}$ (símbolo epsilon) contienen un 0 ó 1 en función de que el símbolo cumpla o no la condición. Las columnas $\mathcal{DPI}$ (derivaciones parciales

por la izquierda), $\mathcal{DPD}$ (derivaciones parciales por la derecha), $\mathcal{AI}$ (adyacencias por la izquierda) y $\mathcal{AD}$ (adyacencia por la derecha) contienen los conjuntos de símbolos correspondientes.

Símbolo	$\mathcal{R}$	$\mathcal{E}$	$\mathcal{DPI}$	$\mathcal{DPD}$	$\mathcal{AI}$	$\mathcal{AD}$
NP	0	0	NP, S	NP,PP,S,VP	and,p,v	PP,VP,and,p,v
PP	0	0		NP,PP,S,VP	NP,PP,S,VP,n	PP,VP,and,p,v
S	1	0	S	S,VP	and,v	PP,and,p
VP	0	0		S,VP	NP,PP,n	PP,and,p
and	0	0			NP,PP,S,VP,n	NP,S,det,n
det	0	0	NP,S		and,p,v	n
n	0	0	NP,S	NP,PP,S,VP	and,det,p,v	PP,VP,and,p,v
p	0	0	PP		NP,PP,S,VP,n	NP,det,n
v	0	0	VP		NP,PP,n	NP,S,det,n

Tabla 2: Tablas de Derivaciones y Adyacencias

4.- Fusiones, Derivaciones y Adyacencias de Eventos: Estructuras EvLink (Enlaces de Eventos)

A partir de las estructuras de datos y el modelo matemático descritos en las secciones anteriores podemos conseguir que el proceso de análisis sintáctico se reduzca a una sencilla operación interna a las estructuras *MvtCaD*.

Cada evento es anclado en sus extremos a sendas estructuras *MvtCaD*, y la utilidad o inutilidad de dicho evento se podrá decidir analizando las relaciones de fusión, derivabilidad y adyacencia que se producen en cada *MvtCaD* (Fig 5).

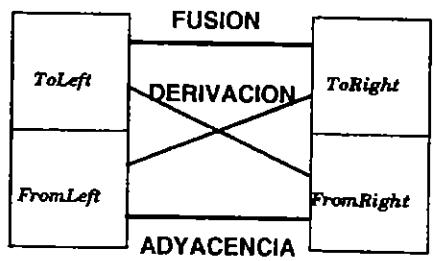

Fig 5: Relaciones Internas en un MvtCaD

Enlaces de Fusión

Se producen entre los componentes *ToLeft* y *ToRight* de un *MvtCaD*. Mediante dichos enlaces se indica que dos eventos pueden fusionarse congruentemente para obtener un evento de mayor amplitud. Esta situación se da en la Figura 4 en el *MvtCaD* número 8 entre los eventos 13 y 15, y en el *MvtCaD* 11 entre los eventos 17 y 19.

Enlaces de Derivación

Se producen entre el extremo abierto de un evento (*ToLeft* o *ToRight*) y el extremo cerrado de otro evento (*FromRight* o *FromLeft*, respectivamente) en un mismo *MvtCaD*. Mediante estos enlaces se indica que el símbolo que espera el primer evento en su extremo abierto podrá ser generado a partir

del segundo evento. Para determinar esta situación se hace uso de las relaciones de derivabilidad parcial definidas en la sección anterior.

Esta situación aparece en la Figura 4 en el *MvtCaD* 10, entre los eventos 16 y 17. El evento 16 ha aplicado la producción 10 sobre el símbolo *p* y espera un símbolo *NP* a su derecha. Por su parte, el evento 17 está aplicando la producción 5 sobre el símbolo *det*, y de tener éxito, obtendrá un símbolo *NP* que podrá usar el evento 16. Esta información esta disponible en la Tabla 2 ($NP \in \mathcal{DPI}(det)$).

Enlaces de Adyacencia

Se producen entre los extremos cerrados de dos eventos consecutivos, es decir, entre los componentes *FromLeft* y *FromRight* de un *MvtCaD*. Mediante este tipo de enlaces se indica que los resultados de la ejecución de ambos eventos o algunas de sus derivaciones parciales posteriores podrán reunirse aplicando alguna de las producciones de la gramática. Matemáticamente se trata de aplicar la relación de adyacencia definida previamente.

Un enlace de adyacencia se produce entre los eventos 1 y 3 en el *MvtCaD* número 2 de la Figura 4. En este caso, el símbolo que generará el evento 1, *NP*, es adyacente por la izquierda con el símbolo que generará el evento número 3, *VP*, pues existe una producción, la número 1, donde aparecen de forma consecutiva ambos símbolos. Esta información aparece asimismo registrada en la tabla 2: $VP \in \mathcal{AD}(NP)$ y $NP \in \mathcal{AI}(VP)$.

Enlaces de Principio y Final de Cadena

Con las definiciones anteriores, los eventos que entran en el componente *FromRight* del primer *MvtCaD* y *FromLeft* del último quedarían necesariamente sin ningún enlace. Para evitar esta situación, en estos casos se usan los criterios de símbolo más a la izquierda y símbolo más a la derecha.

5.- Los Estados Lógicos de los Eventos

Cada vez que un evento se incorpora al proceso de análisis, es decir, se ancla en los *MvtCaD* que delimitan su intervalo de aplicación, se analizan los enlaces posibles de dicho evento. El estado inicial de un evento es de ejecución si su intervalo de aplicación es cerrado en ambos extremos, o bien de derivación si el intervalo es abierto en alguno de los extremos.

Durante la fase de análisis de enlaces del evento se estudian las 4 posibilidades comentadas en la sección anterior: fusión, derivación, adyacencia y principio o final de cadena. Si alguno de los análisis tiene éxito se crea la correspondiente estructura *EvLink* y se modifica consecuentemente el estado lógico de ambos eventos.

En total, se contemplan cuatro estados lógicos posibles para los eventos: Derivación, Ejecución, Borrado y Epsilon.

- Derivación. El estado de derivación es una situación latente en la que el evento prevé tener éxito pero necesita que otros eventos se ejecuten antes de que él pueda completarse.
- Ejecución. Se corresponde con aquellos eventos cerrados en ambos extremos que han pasado con éxito los análisis de enlaces.
- Borrado. En esta situación se encuentran aquellos eventos que no han logrado pasar alguno de los análisis pertinentes en cualquiera de los extremos.
- Epsilon. Si un evento que está en el estado de borrado posee un extremo abierto sin enlaces, pero cuyo símbolo necesario en dicho extremo es de tipo epsilon, entonces el evento pasa al estado epsilon.

6.- El Algoritmo de Análisis

El algoritmo de análisis cuenta con una fase inicial en la que se crean las estructuras *MvtCaD* y *MvtNod* iniciales a partir de la cadena de entrada. Cada uno de los *MvtNod* genera los eventos determinados por sus tablas de coberturas, los ancla en los *MvtCaD* correspondientes y analiza los enlaces oportunos modificando consecuentemente el estado lógico de los eventos y creando las estructuras *EvLink* necesarias.

En el siguiente listado aparece la salida que genera el modo *trace* del parser durante esta fase de inicio para la gramática y la frase presentadas en las secciones anteriores. El listado incluye únicamente las acciones desencadenadas por los cuatro primeros *MvtNod*.

```
@> Input> I saw Jane and Jack hit the man with a telescope
@>
@> Parsing> (n,v,n,and,n,v,det,n,p,det,n)
@>
@> ----- Generating Events on MvtNod:  n:1
@> CreatingEvent> (e=1,p=04/(1-1),s=n) cbi(FrRi-FrLe) (0 Der)
@>          LogEvent> e=1 (0 Der => 1 Run)
@>          LogEvent> e=1 (1 Run => 2 Del)
@> CreatingEvent> (e=2,p=05/(2-2),s=n) cbd(ToLe-FrLe) (0 Der)
@>          LogEvent> e=2 (0 Der => 2 Del)
@>
@> ----- Generating Events on MvtNod:  v:2
@> CreatingEvent> (e=3,p=08/(1-1),s=v) cbi(FrRi-ToRi) (0 Der)
@>    Analysing> FRLEFRRI (1-3)
@>       Success> Adjacency FRLEFRRI (1-3)
@>       CreatingLink> (1-3)
@>          LogEvent> e=1 (2 Del => 1 Run)
@>          LogEvent> e=3 (0 Der => 2 Del)
@>    Analysing> FRLEFRRI (2-3)
@>       Success> Adjacency FRLEFRRI (2-3)
@>       CreatingLink> (2-3)
@> CreatingEvent> (e=4,p=09/(1-1),s=v) cbi(FrRi-ToRi) (0 Der)
@>    Analysing> FRLEFRRI (1-4)
@>       Success> Adjacency FRLEFRRI (1-4)
@>       CreatingLink> (1-4)
@>          LogEvent> e=4 (0 Der => 2 Del)
@>    Analysing> FRLEFRRI (2-4)
@>       Success> Adjacency FRLEFRRI (2-4)
@>       CreatingLink> (2-4)
@>
@> ----- Generating Events on MvtNod:  n:3
@> CreatingEvent> (e=5,p=04/(1-1),s=n) cbi(FrRi-FrLe) (0 Der)
@>          LogEvent> e=5 (0 Der => 1 Run)
@>    Analysing> TORIFRRI (3-5)
@>       Success> Derivation TORIFRRI (3-5)
@>       CreatingLink> (3-5)
@>          LogEvent> e=3 (2 Del => 0 Der)
@>          LogEvent> e=5 (1 Run => 2 Del)
@>    Analysing> TORIFRRI (4-5)
@>       Success> Derivation TORIFRRI (4-5)
@>       CreatingLink> (4-5)
@>          LogEvent> e=4 (2 Del => 0 Der)
@> CreatingEvent> (e=6,p=05/(2-2),s=n) cbd(ToLe-FrLe) (0 Der)
@>    Analysing> TORITOLE (3-6)
@>    Analysing> TORITOLE (4-6)
@>          LogEvent> e=6 (0 Der => 2 Del)
@>
@> ----- Generating Events on MvtNod:  and:4
@> CreatingEvent> (e=7,p=03/(2-2),s=and) cbm(ToLe-ToRi) (0 Der)
@>    Analysing> FRLETOLE (5-7)
@>       Success> Derivation FRLETOLE (5-7)
@>       CreatingLink> (5-7)
@>          LogEvent> e=5 (2 Del => 1 Run)
@>          LogEvent> e=7 (0 Der => 2 Del)
@>    Analysing> FRLETOLE (6-7)
@>       Success> Derivation FRLETOLE (6-7)
@>       CreatingLink> (6-7)
@> CreatingEvent> (e=8,p=07/(2-2),s=and) cbm(ToLe-ToRi) (0 Der)
@>    Analysing> FRLETOLE (5-8)
@>       Success> Derivation FRLETOLE (5-8)
@>       CreatingLink> (5-8)
@>          LogEvent> e=8 (0 Der => 2 Del)
@>    Analysing> FRLETOLE (6-8)
@>       Success> Derivation FRLETOLE (6-8)
@>       CreatingLink> (6-8)
```

Una vez finalizada esta fase inicial se entra en un ciclo de análisis cuyo algoritmo es:

```
ciclo = 1
while (ciclo)
      ciclo = 0
      if (evento = ObtenEventoEpsilon())
            ciclo = 1
            ExpansionEpsilon(evento)
      else if (evento = ObtenEventoEliminacion())
            ciclo = 1
            EliminacionEvento(evento)
      else if  (evento = ObtenEventoEjecucion())
            ciclo = 1
            EjecucionEvento(evento)
      else if (evlink = ObtenEvLinkFusion())
            ciclo = 1
            FusionEvLink(evlink)
```

El objetivo de este algoritmo es establecer un orden jerárquico estricto en la aplicación de las diferentes acciones. Cada una de estas acciones modifica los *MvtCaD* correspondientes. La ejecución de un evento incorpora un nuevo *MvtNod* al proceso de análisis y lanza los eventos asociados con éste. La operación más complicada es la de fusión, en la que se han de tener en cuenta posibles situaciones de fusión con autoderivación, en las que dos eventos no sólo deben fusionarse sino que es necesario mantener uno o los dos anteriores por razones de ambigüedad estructural.

El siguiente listado contiene parte del proceso de análisis para la misma gramática y frase que venimos considerando:

```
@> RemovingEvent> (e=2,p=05/(2-2) Del
@>       RemovingLink> (2-3)
@>       RemovingLink> (2-4)
@> RemovingEvent> (e=6,p=05/(2-2) Del
@>       RemovingLink> (6-7)
@>       RemovingLink> (6-8)
@> RemovingEvent> (e=10,p=05/(2-2) Del
@>       RemovingLink> (10-11)
@>       RemovingLink> (10-12)
@> RemovingEvent> (e=14,p=04/(1-1) Del
@>       RemovingLink> (14-16)
@> RemovingEvent> (e=18,p=04/(1-1) Del
@>
@> RunningEvent> (e=1,p=04/(1-1),s=NP) MvtNod new
@> RemovingEvent> (e=1,p=04/(1-1) Run
@>       RemovingLink> (1-3)
@>          LogEvent> e=3 (0 Der => 2 Del)
@>       RemovingLink> (1-4)
@>          LogEvent> e=4 (0 Der => 2 Del)
@>
@> ----- Generating Events on MvtNod:  NP:12
@> CreatingEvent> (e=20,p=01/(1-1),s=NP) cbi(FrRi-ToRi) (0 Der)
@>          LogEvent> e=20 (0 Der => 2 Del)
@>    Analysing> TORIFRRI (20-3)
@>       Success> Derivation TORIFRRI (20-3)
@>       CreatingLink> (20-3)
@>          LogEvent> e=20 (2 Del => 0 Der)
@>          LogEvent> e=3 (2 Del => 0 Der)
@>    Analysing> TORIFRRI (20-4)
@>       Success> Derivation TORIFRRI (20-4)
@>       CreatingLink> (20-4)
@>          LogEvent> e=4 (2 Del => 0 Der)
@> CreatingEvent> (e=21,p=06/(1-1),s=NP) cbi(FrRi-ToRi) (0 Der)
@>          LogEvent> e=21 (0 Der => 2 Del)
@>    Analysing> TORIFRRI (21-3)
@>    Analysing> TORIFRRI (21-4)
@> CreatingEvent> (e=22,p=07/(1-1),s=NP) cbi(FrRi-ToRi) (0 Der)
@>          LogEvent> e=22 (0 Der => 2 Del)
@>    Analysing> TORIFRRI (22-3)
@>    Analysing> TORIFRRI (22-4)
@> CreatingEvent> (e=23,p=07/(3-3),s=NP) cbd(ToLe-FrLe) (0 Der)
@>          LogEvent> e=23 (0 Der => 2 Del)
@>    Analysing> FRLEFRRI (23-3)
@>       Success> Adjacency FRLEFRRI (23-3)
@>       CreatingLink> (23-3)
@>    Analysing> FRLEFRRI (23-4)
@>       Success> Adjacency FRLEFRRI (23-4)
@>       CreatingLink> (23-4)
@> CreatingEvent> (e=24,p=08/(2-2),s=NP) cbd(ToLe-FrLe) (0 Der)
@>          LogEvent> e=24 (0 Der => 2 Del)
```

```
@>    Analysing> FRLEFRRI (24-3)
@>    Analysing> FRLEFRRI (24-4)
@> CreatingEvent> (e=25,p=10/(2-2),s=NP) cbd(ToLe-FrLe) (0 Der)
@>          LogEvent> e=25 (0 Der => 2 Del)
@>    Analysing> FRLEFRRI (25-3)
@>       Success> Adjacency FRLEFRRI (25-3)
@>       CreatingLink> (25-3)
@>    Analysing> FRLEFRRI (25-4)
@>       Success> Adjacency FRLEFRRI (25-4)
@>       CreatingLink> (25-4)
@>
@> RemovingEvent> (e=21,p=06/(1-1) Del
@> RemovingEvent> (e=22,p=07/(1-1) Del
@> RemovingEvent> (e=23,p=07/(3-3) Del
@>       RemovingLink> (23-3)
@>       RemovingLink> (23-4)
@> RemovingEvent> (e=24,p=08/(2-2) Del
@> RemovingEvent> (e=25,p=10/(2-2) Del
@>       RemovingLink> (25-3)
@>       RemovingLink> (25-4)
```

7.- Arboles Múltiples Virtuales

Como se acaba de indicar, la ejecución de un evento introduce un nuevo *MvtNod*, es decir, un nuevo subcomponente del bosque de análisis. Todo *MvtNod* se caracteriza por dos parámetros básicos: el ámbito o intervalo de aplicación y el símbolo asociado al análisis de dicho intervalo. Ahora bien, determinados *MvtNod* serán localmente ambigüos y cada uno de los posibles análisis se representará mediante una estructura *MvtAna*. Una estructura *MvtAna* se define simplemente como una secuencia de estructuras *MvtNod*.

Esta idea, en principio elemental e inofensiva, consistente en representar el bosque de análisis en base a dos estructuras interdependientes: *MvtNod* que contienen listas de *MvtAna*, y *MvtAna* que son en realidad secuencias de *MvtNod*, tiene consecuencias bastante interesantes.

En lo que respecta al modelo formal, la principal consecuencia de este modelo de datos es una separación representacional entre la estructura o esqueleto del bosque de análisis y el contenido de cada uno de los nodos de éste. Esta separación hace que la estructura de análisis final obtenida se base únicamente en un conjunto de relaciones "virtuales" entre los componentes *MvtNod* y *MvtAna*. A este modelo final se le ha denominado consiguientemente Arbol Múltiple Virtual.

Para la gramática y la oración que venimos usando como ejemplo, la salida del parser es:

```
MvtNod n:1     -> I
MvtNod v:2     -> saw
MvtNod n:3     -> Jane
MvtNod and:4   -> and
MvtNod n:5     -> Jack
MvtNod v:6     -> hit
MvtNod det:7   -> the
MvtNod n:8     -> man
MvtNod p:9     -> with
MvtNod det:10  -> a
MvtNod n:11    -> telescope
MvtNod NP:12   ->  n:1
MvtNod NP:13   ->  n:3
MvtNod NP:14   ->  n:5
MvtNod NP:15   ->  det:7,n:8
MvtNod NP:16   ->  det:10,n:11
MvtNod VP:17   ->  v:2,NP:13
MvtNod NP:18   ->  NP:13,and:4,NP:14
MvtNod VP:19   ->  v:6,NP:15
MvtNod PP:20   ->  p:9,NP:16
MvtNod S:21    ->  NP:12,VP:17
MvtNod S:22    ->  NP:14,VP:19
MvtNod S:23    ->  NP:18,VP:19
MvtNod NP:24   ->  NP:15,PP:20
MvtNod S:25    ->  S:22,PP:20 || NP:14,VP:29
MvtNod S:26    ->  S:21,and:4,S:22 || NP:12,VP:28
MvtNod S:27    ->  S:23,PP:20 || NP:18,VP:29
MvtNod VP:28   ->  v:2,S:23
MvtNod VP:29   ->  v:6,NP:24
MvtNod S:30    ->  S:21,and:4,S:25 || S:26,PP:20 || NP:12,VP:31
MvtNod VP:31   ->  v:2,S:27
MvtRoot S:30
```

Este listado contiene toda la información relativa al análisis gramatical de la oración. Para ello basta considerar el nodo raíz del árbol múltiple virtual (*S:30*) y expandirlo siguiendo las derivaciones de cada uno de los nodos.

8.- Resultados Experimentales

Para analizar el rendimiento de este sistema proponemos un modelo basado en el análisis estadístico del comportamiento del parser ante diversos fenómenos gramaticalmente complejos propios de los lenguajes naturales. Los resultados obtenidos sólo son generalizables para dichos fenómenos y en las situaciones contempladas en el experimento. No obstante la estabilidad del parser ante todos los fenómenos estudiados y la complejidad de estos nos permite ser optimistas a la hora de pensar en una generalización de mayor alcance.

8.1.- Gramáticas para la Evaluación

Con el objetivo de evaluar la eficiencia real del parser se han diseñado tres gramáticas que contienen modelos formales donde se incluyen algunos de los fenómenos lingüísticos de los lenguajes naturales más complejos desde el punto de vista de su análisis sintáctico, y se han medido los tiempos invertidos con una implementación en C del algoritmo propuesto.

Estructuras Recursivas

La siguiente gramática contiene los tres tipos de estructuras recursivas: a la izquierda, a la derecha y recursividad interna:

```
(rg:O)
(O->A BC D)
(A->a A)
(A->)
(BC->b BC c)
(BC->)
(D->D d)
(D->)
```

Las cadenas analizadas tienen el formato:

```
a* (b c)* d*
```

Dependencias Locales

En este caso se ha pretendido diseñar una gramática donde el análisis de un nodo tuviese una fuerte dependencia del o de los análisis de los nodos que tiene a su alrededor. En concreto se ha creado una gramática con 50 símbolos terminales (`s0` a `s49`), y las construcciones permitidas son aquellas de cualquier longitud tales que para dos símbolos consecutivos `sX` y `sY` se cumple `|X-Y| = 1` ó `|X-Y| = 49`. Esta gramática posee 300 producciones. El siguiente listado contiene las 12 primeras:

```
        (rg:O)
% S00
        (0:O -> S00)
        (0:O -> Sx00)
        (0:Sx00 -> S49  S00)
        (0:Sx00 -> Sx49 S00)
        (0:Sx00 -> S01  S00)
        (0:Sx00 -> Sx01 S00)

% S01
        (0:O -> S01)
        (0:O -> Sx01)
        (0:Sx01 -> S00  S01)
        (0:Sx01 -> Sx00 S01)
        (0:Sx01 -> S02  S01)
        (0:Sx01 -> Sx02 S01)
```

Las cadenas de símbolos analizadas tienen la forma:

```
(s00 s01 ... s49)*
```

Dependencias No Locales

La gramática que se lista a continuación contiene un modelo especialmente complejo donde se unen a los fenómenos de recursividad y linealidad un fuerte componente de dependencia no local:

```
(rg:O1 O2 O3)       % Raices de la Gramática
(O1 -> A1 B1 C1)
(O2 -> C2 A2 B2)
(O3 -> B3 C3 A3)
(A1 -> A1 a)
(A1 -> a)
(A2 -> A2 a)
(A2 -> a)
(A3 -> A3 a)
(A3 -> a)
(B1 -> B1 b)
(B1 -> b)
(B2 -> B2 b)
(B2 -> b)
(B3 -> B3 b)
(B3 -> b)
(C1 -> C1 c)
(C1 -> c)
(C2 -> C2 c)
(C2 -> c)
(C3 -> C3 c)
(C3 -> c)
```

Las cadenas analizadas han tenido la forma:

```
a* b* c*
c* a* b*
b* c* a*
```

8.2.- Eficiencia Real

La siguientes tablas muestran los resultados obtenidos por el parser. Cada una de las tablas contiene la siguiente información:

- P: Número de análisis. En todos los casos se han realizado 500 análisis para obtener la media de los tiempos invertidos.
- W: Número total de palabras analizadas.
- W/P: Palabras por análisis, es decir, tamaño de la cadena de entrada al parser. Se han analizado entradas que contienen entre 50 y 1.000 palabras.
- E: Número total de eventos generados para los 500 análisis.
- T: Tiempo total invertido en los 500 análisis.
- T/E: Tiempo medio consumido por cada evento, medido en microsegundos.
- T/W: Tiempo medio consumido en el análisis de cada palabra, medido en microsegundos.
- W/S: Palabras analizadas por segundo.

Estructuras Recursivas

```
|P:500|W:025000|W/P:0050|E:0087500|T:002.067|T/E:024|T/W:083|W/S:12097|
|P:500|W:050000|W/P:0100|E:0175500|T:004.183|T/E:024|T/W:084|W/S:11953|
|P:500|W:075000|W/P:0150|E:0262500|T:006.216|T/E:024|T/W:083|W/S:12065|
|P:500|W:100000|W/P:0200|E:0350500|T:008.866|T/E:025|T/W:089|W/S:11279|
|P:500|W:125000|W/P:0250|E:0437500|T:011.366|T/E:026|T/W:091|W/S:10998|
|P:500|W:150000|W/P:0300|E:0525500|T:013.666|T/E:026|T/W:091|W/S:10976|
|P:500|W:175000|W/P:0350|E:0612500|T:016.883|T/E:028|T/W:096|W/S:10366|
|P:500|W:200000|W/P:0400|E:0700500|T:019.316|T/E:028|T/W:097|W/S:10354|
|P:500|W:225000|W/P:0450|E:0787500|T:022.116|T/E:028|T/W:098|W/S:10174|
|P:500|W:250000|W/P:0500|E:0875500|T:024.932|T/E:028|T/W:100|W/S:10027|
|P:500|W:275000|W/P:0550|E:0962500|T:028.016|T/E:029|T/W:102|W/S:09816|
|P:500|W:300000|W/P:0600|E:1050500|T:030.665|T/E:029|T/W:102|W/S:09783|
|P:500|W:325000|W/P:0650|E:1137500|T:034.015|T/E:030|T/W:105|W/S:09555|
|P:500|W:350000|W/P:0700|E:1225500|T:036.782|T/E:030|T/W:105|W/S:09516|
|P:500|W:375000|W/P:0750|E:1312500|T:039.465|T/E:030|T/W:105|W/S:09502|
|P:500|W:400000|W/P:0800|E:1400500|T:043.015|T/E:031|T/W:108|W/S:09299|
|P:500|W:425000|W/P:0850|E:1487500|T:046.398|T/E:031|T/W:109|W/S:09160|
|P:500|W:450000|W/P:0900|E:1575500|T:049.031|T/E:031|T/W:109|W/S:09178|
|P:500|W:475000|W/P:0950|E:1662500|T:052.165|T/E:031|T/W:110|W/S:09106|
|P:500|W:500000|W/P:1000|E:1750500|T:055.214|T/E:032|T/W:110|W/S:09056|
```

Dependencias Locales

```
|P:500|W:025000|W/P:0050|E:0249000|T:005.366|T/E:022|T/W:215|W/S:04659|
|P:500|W:050000|W/P:0100|E:0499000|T:011.400|T/E:023|T/W:228|W/S:04386|
|P:500|W:075000|W/P:0150|E:0749000|T:018.249|T/E:024|T/W:243|W/S:04110|
|P:500|W:100000|W/P:0200|E:0999000|T:025.899|T/E:026|T/W:259|W/S:03861|
|P:500|W:125000|W/P:0250|E:1249000|T:034.199|T/E:027|T/W:274|W/S:03655|
|P:500|W:150000|W/P:0300|E:1499000|T:043.432|T/E:029|T/W:290|W/S:03454|
|P:500|W:175000|W/P:0350|E:1749000|T:051.248|T/E:029|T/W:293|W/S:03415|
|P:500|W:200000|W/P:0400|E:1999000|T:060.948|T/E:030|T/W:305|W/S:03282|
|P:500|W:225000|W/P:0450|E:2249000|T:072.014|T/E:032|T/W:320|W/S:03124|
|P:500|W:250000|W/P:0500|E:2499000|T:082.547|T/E:033|T/W:330|W/S:03029|
|P:500|W:275000|W/P:0550|E:2749000|T:093.780|T/E:034|T/W:341|W/S:02932|
|P:500|W:300000|W/P:0600|E:2999000|T:106.712|T/E:036|T/W:356|W/S:02811|
|P:500|W:325000|W/P:0650|E:3249000|T:119.545|T/E:037|T/W:368|W/S:02719|
|P:500|W:350000|W/P:0700|E:3499000|T:135.378|T/E:039|T/W:387|W/S:02585|
|P:500|W:375000|W/P:0750|E:3749000|T:147.544|T/E:039|T/W:393|W/S:02542|
|P:500|W:400000|W/P:0800|E:3999000|T:162.394|T/E:041|T/W:406|W/S:02463|
|P:500|W:425000|W/P:0850|E:4249000|T:177.176|T/E:042|T/W:417|W/S:02399|
|P:500|W:450000|W/P:0900|E:4499000|T:195.376|T/E:043|T/W:434|W/S:02303|
|P:500|W:475000|W/P:0950|E:4749000|T:210.542|T/E:044|T/W:443|W/S:02256|
|P:500|W:500000|W/P:1000|E:4999000|T:229.407|T/E:046|T/W:459|W/S:02180|
```

Dependencias No Locales

```
|P:500|W:025000|W/P:0050|E:0201000|T:004.933|T/E:025|T/W:197|W/S:05068|
|P:500|W:050000|W/P:0100|E:0401000|T:009.916|T/E:025|T/W:198|W/S:05042|
|P:500|W:075000|W/P:0150|E:0601000|T:015.783|T/E:026|T/W:210|W/S:04752|
|P:500|W:100000|W/P:0200|E:0801000|T:020.916|T/E:026|T/W:209|W/S:04781|
|P:500|W:125000|W/P:0250|E:1001000|T:026.566|T/E:027|T/W:213|W/S:04705|
|P:500|W:150000|W/P:0300|E:1201000|T:033.132|T/E:028|T/W:221|W/S:04527|
|P:500|W:175000|W/P:0350|E:1401000|T:039.432|T/E:028|T/W:225|W/S:04438|
|P:500|W:200000|W/P:0400|E:1601000|T:046.281|T/E:029|T/W:231|W/S:04321|
|P:500|W:225000|W/P:0450|E:1801000|T:053.031|T/E:029|T/W:236|W/S:04243|
|P:500|W:250000|W/P:0500|E:2001000|T:060.031|T/E:030|T/W:240|W/S:04165|
|P:500|W:275000|W/P:0550|E:2201000|T:067.447|T/E:031|T/W:245|W/S:04077|
|P:500|W:300000|W/P:0600|E:2401000|T:074.797|T/E:031|T/W:249|W/S:04011|
|P:500|W:325000|W/P:0650|E:2601000|T:082.347|T/E:032|T/W:253|W/S:03947|
|P:500|W:350000|W/P:0700|E:2801000|T:090.696|T/E:032|T/W:259|W/S:03859|
|P:500|W:375000|W/P:0750|E:3001000|T:098.746|T/E:033|T/W:263|W/S:03798|
|P:500|W:400000|W/P:0800|E:3201000|T:107.529|T/E:034|T/W:269|W/S:03720|
|P:500|W:425000|W/P:0850|E:3401000|T:116.162|T/E:034|T/W:273|W/S:03659|
|P:500|W:450000|W/P:0900|E:3601000|T:124.928|T/E:035|T/W:278|W/S:03602|
|P:500|W:475000|W/P:0950|E:3801000|T:134.111|T/E:035|T/W:282|W/S:03542|
|P:500|W:500000|W/P:1000|E:4001000|T:143.544|T/E:036|T/W:287|W/S:03483|
```

La Figura 6 contiene los gráficos para estas tres gramáticas. En cada uno de los gráficos se muestran las relaciones entre el tiempo total invertido (T) en el análisis de las 500 entradas (medido en segundos) y el tamaño de la cadena (W/P) de cada una de las entradas que se analiza (entre 50 y 1000 palabras)

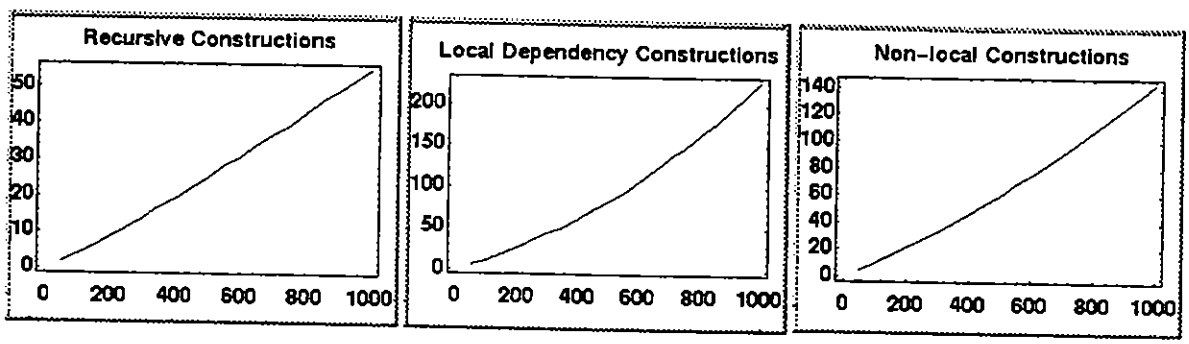

Fig 6: Gráficos Tiempo-Palabras

8.3.- Complejidad Algorítmica

Finalmente mostramos los modelos que se obtienen mediante un test de Regresión Lineal Simple. En el primer caso se ha usado la variable *T* como respuesta y *W*lg(W)* como factor. El segundo análisis analiza el tiempo por palabra (*T/W*) en función de *W*lg(W)*. Asimismo se incluyen los coeficientes de correlación de Pearson (CCP) para las dos situaciones:

Estructuras Recursivas

$$T = -5.183 + 219E\text{-}7 * (W \lg(W))$$

$$CCP(T,W \lg(W)) = 0.999$$

$$T/W = 0.0001 + 46E\text{-}13 * (W \lg(W))$$

$$CCP(T/W,W \lg(W)) = 0.974$$

Dependencias Locales

$$T = -17.82 + 352E\text{-}7 * (W \lg(W))$$

$$CCP(T,W \lg(W)) = 0.993$$

$$T/W = 0.0002 + 38E\text{-}12 * (W \lg(W))$$

$$CCP(T/W,W \lg(W)) = 0.998$$

Dependencias No Locales

$$T = -1.031 + 851\text{-}8 * (W \lg(W))$$

$$CCP(T,W \lg(W)) = 0.998$$

$$T/W = 0.0002 + 14E\text{-}12 * (W \lg(W))$$

$$CCP(T/W,W \lg(W)) = 0.998$$

Conclusión

La mayoría de las técnicas de análisis para lenguajes libres de contexto son mecanismos híbridos que incorporan en distintos grados componentes de análisis ascendente junto con otros de predicción descendente. En esta misma línea, se presenta un modelo para el análisis bidireccional y dirigido por eventos.

La dirección por eventos permite una manipulación bastante flexible de fenómenos tales como la ambigüedad léxica o ε-producciones. Puesto que cada evento se corresponde con la aplicación potencial de una producción sobre la superficie de análisis activa en un momento determinado, el objetivo del algoritmo será la eliminación del mayor número posible de eventos para disminuir el número de cómputos necesarios para el análisis. Para lograr un modelo consistente y robusto de eliminación de eventos se definen formalmente una serie de relaciones entre los símbolos y las producciones de una gramática, sobre los que se implementan los mecanismos de control.

Una de las claves de la eficiencia del algoritmo es el modelo de representación de información simbólica sobre un sistema dirigido fundamentalmente al cómputo numérico (como es un ordenador). El modelo adecúa la representación de una gramática a las necesidades del parser, eliminando las operaciones de búsqueda y reduciendo considerablemente las de comparación.

Finalmente se presentan los resultados experimentales para gramáticas que contienen fenómenos tales como recursividad, dependencias locales (o lineales) y no locales, obteniéndose para todos ellos un nivel $O(n \lg(n))$ de complejidad real.

Bibliografía

- Aho, A.V.; Sethi, R.; Ullman, J.D. 1985. *Compilers: Principles, Techniques and Tools.* Addison-Wesley.
- Aho, A.V.; Ullman, J.D. 1972. *The Theory of Parsing, Translation and Compiling. Volume I: Parsing.* Englewood Cliffs, N.J.: Prentice-Hall.
- Alblas, H.; den Akker, R.; Luttighuis, P.O.; Sikkel, K. 1994. "A bibliography on parallel parsing". *SIGPLAN*

Notices, **29(1)**, 54-65.

Andrews, N.A.; Brown, J.C. 1993. "A high-speed natural language parser". *AISB Quarterly*, **86**, 12-19.

Bolc, L. ed. 1987. *Natural Language Parsing Systems*. Heidelberg: Springer-Verlag.

Chapman, N. P. 1987. *LR Parsing. Theory and Practice*. Cambridge : Cambridge University Press.

Chung, M.; Moldovan, D. 1994. "Applying Parallel Processing to Natural-Language Processing". *IEEE Expert*, Febrero 1994, 36-44.

Carroll, J. 1994. "Relating Complexity to Practical Performance in Parsing with Wide-Coverage Unification Grammars". *CMP-LG e-print archive cmp-lg/9405033*.

Dowding, J.; Moore, R.; Andry, F.; Moran, D. 1994. "Interleaving Sintax and Semantics in an Efficient Bottom-Up Parser". *Proceedings of the 32nd Annual Meeting of the ACL*.

Earley, J. 1970. "An efficient context-free parsing algorithm" *Communications of the ACM*, **14**: 453-460. (También en Grosz, Sparck Jones y Webber 1986: 25-33)

Grosz, B; K. Sparck Jones y B. L. Webber. 1986. *Readings in Natural Language Processing*. Los Altos: Morgan Kaufmann.

Harrison, M. 1978. *Introduction to Formal Language Theory*. Addison-Wesley.

Hendler, J. 1994. "Beyond the Fifth Generation: Parallel AI Research in Japan". *IEEE Expert*, Febrero 1994, 2-7.

Kay, M. 1980. "Algorithm Schemata and Data Structures in Syntactic Processing". En Grosz, Sparck Jones y Webber 1986: 35-70.

Martin, W.A.; Church, K.W.; Patil, R.S. 1987. "Preliminary Analysis of a Breadth-First Parsing Algorithm: Theoretical and Experimental Results". En Bolc 1987, 267-328.

Nijholt, A. 1991. "Overview of Parallel Parsing Strategies". En Tomita 1991: 207-229.

Nurkkala, T.; Kumar, V. 1994. "The performance of a highly unstructured parallel algorithm on the KSR1". *Proceedings of the Scalable High-Performance Computing Conference*, 215-220.

Partee, B. H.; Meulen, A. ter; Wall. R.E. 1990. *Mathematical Methods in Linguistics*. Kluwer Academic Publishers.

Shann, P. 1991. "Experiments with GLR and Chart Parsing". En Tomita 1991, 17-34.

Small, S.L. 1987. "A Distributed Word-Based Approach to Parsing". En Bolc 1987: 161-201.

Tomita, M. 1987. "An Efficient Augmented Context-Free Parsing Algorithm", *Computational Linguistics* **13 (1-2)**, 31-46.

Tomita, M. ed. 1991. *Generalized LR Parsing*. London: Kluwer Academic Press.

Wah, B.W. ed. 1994. "Report on Workshop on High Performance Computing and Communications for Grand Challenge Applications: Computer Vision, Speech and Natural Language Processing, and Artificial Intelligence" *IEEE Transactions on Knowledge and Data Engineering*, **5(1)**.