# RELACIONES DE COBERTURA, DERIVABILIDAD Y ADYACENCIA COMO MODELO FORMAL PARA EL ANALISIS SINTACTICO EFICIENTE DE LENGUAJES NATURALES

*José F. Quesada*

Automatic treatment of natural languages (Natural Language Processing) includes syntactic analysis (parsing) as one of its main techniques: that is, to decide whether a string of words belongs or not to a given grammar, and if ownership exists (grammaticality) to obtain the syntactic structure or derivation tree. The computational complexity of the problem along with the characteristics of the implementation of the parsing algorithms complicate the practical application of these algorithms to natural languages. This paper presents a formal model aimed at the compilation of the context-free kernel of a grammar (following the unification-based paradigm). This model is based on the relations of derivability, adjacency and coverage. Also, there will be presented a computational model for the representation of this information. Both components improve the computational complexity and the efficiency of the parsing algorithm, allowing its application for natural languages in real-time environments.

## *1.- Análisis Ascendente con Predicciones Descendentes*

Una de las estrategias más frecuentes en el diseño de analizadores sintácticos para lenguajes libres de contexto es el análisis ascendente enriquecido con predicciones descendentes. Los modelos ascendentes han sido defendidos por su eficiencia [Pereira 1985; Tomita 1987; Carter 1990; Shan 1991], aunque esta se mejora sustancialmente si se incorporan predicciones descendentes [Andrews y Brown 1993; Dowding et al 1994; Carroll 1994]. Desde el punto de vista de la estrategia de análisis, se trataría de analizadores controlados simultáneamente por los datos (la cadena de entrada que se va a analizar) y la meta (la gramática).

Este doble control como núcleo de la estrategia de análisis está presente en el algoritmo de Earley [1970], donde el operador *predictor* asume la función de predicción descendente, introduciendo en un estado las producciones aplicables según la configuración del

I.S.S.N. 1132-0265

analizador en ese momento, mientras que el operador *scanner* junto con el componente *look-ahead* incorporan la estrategia ascendente. Un estudio detallado del algoritmo nos ha pensar en una estrategia de análisis fundamentalmente descendente que es enriquecida c un componente de control ascendente.

Para los algoritmos basados en la técnica LR [Aho y Ullman 1972; Aho et al 198 Chapman 1987; Tomita 1991], las tablas *action* y *goto* hacen de la estrategia ascendente componente básico del modelo. Sin embargo, no se trata de analizadores exclusivamen ascendentes, ya que la pila (o grafo) de estados almacena el contexto a la izquierda d símbolo que se está analizando, y esta información funciona realmente como un componen predictivo que se puede usar para resolver conflictos (*shift-reduce* y *reduce-reduce*) tal como proponen Pereira [1985] y Shieber [1983].

La combinación de las estrategias ascendente y descendente forma parte asimismo de l analizadores basados en *charts* [Kay 1980]. De acuerdo con Martin et al [1987]:

> *"All efficient parsers combine both top-down and bottom-up information in some way (usi the chart, for example), so that it is useless to classify these parsers as one type or the othe [Martin et al 1987, 272]*

La fusión de componentes ascendentes y descendentes como estrategia de análisis muy común también en las implementaciones de parsers sobre sistemas distribuidos masivamente paralelos [Alblas et al 1994; Chung y Moldovan 1994; Nurkkala y Kum 1994].

En esta misma línea, en las secciones siguientes se presentará un parser que combir ambas estrategias (ascedente y descendente) con el objetivo de aumentar la robustez eficiencia. Las características de un entorno de Inteligencia Artificial en Tiempo Real [Wa 1994; Hendler 1994; Alexandersson 1995] imponen fuertes restricciones de eficiencia, que justifica abordar los problemas relacionados con los modelos de memoria y l estructuras de datos.

El resto del artículo se divide de la siguiente forma. En la sección 2 se describe el model formal o matemático sobre el que descansa el proceso de compilación de una gramática; resultado de esta compilación es la obtención de una serie de tablas: cobertura derivaciones y adyacencias. La sección 3 se dedica al estudio del mecanismo de control d parser; se puede considerar que el núcleo de este módulo es la idea de dirección por eventos

## 2.- *Compilación de la Gramática*

### *2.1.- Derivación Ascendente*

El objetivo de este apartado es obtener una definición formal de la relación de derivació ascendente para una gramática libre de contexto [Harrison 1978; Partee et al 1990].

#### *2.1.1.- Producción libre de contexto*

Dado un conjunto $\mathcal{X}$, no vacío, diremos que $p$ es una producción libre de contexto sobr

$\mathcal{X}$, con $p = <\delta,\Delta>$, y lo notaremos mediante $\delta \rightarrow \Delta$, sii $\delta \in \mathcal{X}$ y $\Delta \in \mathcal{X}^*$

Dado un conjunto $\mathcal{X}$, no vacío, diremos que $\mathcal{P}$ es un conjunto de producciones libres de contexto sobre $\mathcal{X}$, sii $\mathcal{P} \subseteq \mathcal{X} \times \mathcal{X}^*$

Un conjunto de producciones, $\mathcal{P}$, sobre un conjunto $\mathcal{X}$, define tres subconjuntos de $\mathcal{X}$, que denominaremos:

- Conjunto de no terminales: $\mathcal{P}_{\mathcal{N}}$

$$\mathcal{P}_{\mathcal{N}} = \{ \delta \in \mathcal{X} : \exists \Delta \in \mathcal{X}^* (\delta \rightarrow \Delta \in \mathcal{P}) \}$$

- Conjunto de terminales: $\mathcal{P}_{\mathcal{T}}$

$$\mathcal{P}_{\mathcal{T}} = \{ \delta \in \mathcal{X} : \delta \notin \mathcal{P}_{\mathcal{N}} \wedge \exists \lambda \in \mathcal{X}, \exists \Gamma, \Omega \in \mathcal{X}^* (\lambda \rightarrow \Gamma \delta \Omega \in \mathcal{P}) \}$$

- Conjunto vocabulario: $\mathcal{P}_{\mathcal{V}}$

$$\mathcal{P}_{\mathcal{V}} = \mathcal{P}_{\mathcal{N}} \cup \mathcal{P}_{\mathcal{T}}$$

### *2.1.2.- Gramática libre de contexto*

Dado un conjunto de producciones, $\mathcal{P}$, y un conjunto $\mathcal{R}$, tal que $\mathcal{R} \subseteq \mathcal{P}_{\mathcal{N}}$, se define una gramática libre de contexto, $\mathcal{G}$, como $\mathcal{G} = <\mathcal{P}, \mathcal{R}>$.

A partir de una gramática quedan definidos automáticamente los siguientes conjuntos:

- Producciones de $\mathcal{G}$: $\mathcal{G}_{\mathcal{P}}$
- Raíces de $\mathcal{G}$: $\mathcal{G}_{\mathcal{R}}$
- Símbolos terminales de $\mathcal{G}$: $\mathcal{T}$ ( = $\mathcal{P}_{\mathcal{T}}$)
- Símbolos no terminales de $\mathcal{G}$: $\mathcal{N}$ ( = $\mathcal{P}_{\mathcal{N}}$)
- Vocabulario de $\mathcal{G}$: $\mathcal{V}$ ( = $\mathcal{P}_{\mathcal{V}}$)

### *2.1.3.- Derivación Directa Ascendente en $\mathcal{G}$*

Sean $\delta \in \mathcal{V}$, $\Delta, \Gamma, \Omega \in \mathcal{V}^*$

Definimos la relación de derivación directa ascendente en $\mathcal{G}$, $\Rightarrow_G$, de la siguiente forma:

$$\Gamma \Delta \Omega \Rightarrow_G \Gamma \delta \Omega \quad \text{sii } \delta \rightarrow \Delta \in \mathcal{G}_{\mathcal{P}}$$

### *2.1.4.- Derivación Ascendente en $\mathcal{G}$*

Sean $\Gamma, \Omega \in \mathcal{V}^*$.

Definimos la relación de derivación en $\mathcal{G}$, $\Rightarrow\!\!\!\!\Rightarrow_G$, como sigue:

$$\Gamma \Rightarrow\!\!\!\!\Rightarrow_G \Omega \quad \text{sii } \exists \Delta_1, \ldots, \Delta_n \in \mathcal{V}^* \text{ tales que } \forall_i \ (1 \leq i < n)$$
$$\Delta_i \Rightarrow_G \Delta_{i+1}, \text{ donde } \Delta_1 \equiv \Gamma, \Delta_n \equiv \Omega$$

Matemáticamente la derivación ascendente es la clausura reflexiva y transitiva de la derivación directa.

### 2.1.5.- Sentencias en $\mathcal{G}$

Sea $\Delta \in \mathcal{T}^*$.

Diremos que $\Delta$ es una sentencia de $\mathcal{G}$, y lo representaremos mediante $\mathcal{S}_{\mathcal{G}}(\Delta)$,

sii $\exists\, \delta \in \mathcal{G}_{\mathcal{R}}(\Delta \Rightarrow\!\!\!\!\!\!\Rightarrow_G \delta)$

### 2.1.6.- Lenguaje definido por una Gramática $\mathcal{G}$

Finalmente definimos el concepto de lenguaje según la forma habitual:

$$\mathcal{L}_{\mathcal{G}} = \{\Delta \in \mathcal{T}^* : \mathcal{S}_{\mathcal{G}}(\Delta)\}$$

## 2.2.- Relaciones de Derivación y Adyacencia

En este apartado se definen varias relaciones entre los símbolos del vocabulario de u gramática, a partir de las cuales se consigue el modelo (compilado) de ésta, que perm dirigir al parser de una forma eficiente.

Sean $\alpha, \beta \in \mathcal{V}$:

### 2.2.1.- Derivabilidad por la Izquierda

$\beta$ es derivable por la izquierda a partir de $\alpha$, $\alpha \rightarrow_i \beta$, sii

$$\exists\ \Gamma, \Delta, \Omega \in \mathcal{V}^* (\Gamma\alpha\Delta \Rightarrow\!\!\!\!\!\!\Rightarrow_G \Gamma\beta\Omega)$$

### 2.2.2.- Derivabilidad por la Derecha

$\beta$ es derivable por la derecha a partir de $\alpha$, $\alpha \rightarrow_d \beta$, sii

$$\exists\, \Gamma, \Delta, \Omega \in \mathcal{V}^* (\Gamma\alpha\Delta \Rightarrow\!\!\!\!\!\!\Rightarrow_G \Omega\beta\Delta)$$

### 2.2.3.- Adyacencia Primaria por la Izquierda

$\beta$ es un adyacente primario por la izquierda de $\alpha$, $\alpha \Uparrow \beta$, sii

$$\exists\, \delta \in \mathcal{V}, \exists\, \Gamma, \Omega \in \mathcal{V}^* (\delta \rightarrow \Gamma\alpha\beta\Omega \in \mathcal{G}_{\mathcal{P}})$$

## 2.3.- Modelo Formal de una Gramática $\mathcal{G}$

El modelo final que define una gramática, con vistas a su utilización por el parser, asoc con cada símbolo $\alpha$ del vocabulario de la gramática, cinco conjuntos: Cobertura por Izquierda de $\alpha$: CobI($\alpha$), Cobertura por la Derecha: CobD($\alpha$), Derivaciones por Izquierda: DerI($\alpha$), Derivaciones por la Derecha: DerD($\alpha$) y Adyacencias Primari AdyP($\alpha$).

Formalmente:

$\forall\, \alpha \in \mathcal{V}$:

$$\text{CobI}(\alpha) = \{\, \delta \rightarrow \Delta \in \mathcal{G}_{\mathcal{P}} : \exists\, \Omega \in \mathcal{V}^* (\Delta \equiv \alpha\, \Omega)\}$$

$$\text{CobD}(\alpha) = \{\delta \rightarrow \Delta \in \mathcal{G}_{\mathcal{P}} : \exists\, \Omega \in \mathcal{V}^+ (\Delta \equiv \Omega\, \alpha)\}$$

$$\text{DerI}(\alpha) = \{\beta \in \mathcal{V}: \alpha \rightarrow_i \beta\}$$
$$\text{DerD}(\alpha) = \{\beta \in \mathcal{V}: \alpha \rightarrow_d \beta\}$$
$$\text{AdyP}(\alpha) = \{\beta \in \mathcal{V}: \alpha \Uparrow \beta\}$$

## *2.4.- Un Ejemplo*

Se muestra a continuación el resultado obtenido por este modelo formal para la gramática presentada en [Shieber 1983]. El lenguaje de especificación permitido por el parser permite escribir las gramáticas según el siguiente formalismo:

```
$g % Definición de la Gramática utilizada en Shieber 1983
    (rg:S)  % Raíces de la Gramática
    ( 1: S -> NP  VP)
    ( 2: S -> SS  VP)
    ( 3: NP -> DET  NOM)
    ( 4: NP -> NOM)
    ( 5: NP -> PNOUN)
    ( 6: NP -> NP  SSNP)
    ( 7: NP -> NP  PARTP)
    ( 8: NP -> NP  PP)
    ( 9: DET -> NP  s)
    (10: NOM -> N)
    (11: NOM -> ADJ  NOM)
    (12: VP -> AUX  VP)
    (13: VP -> V0)
    (14: VP -> V1  NP)
    (15: VP -> V2  NP  PP)
    (16: VP -> V3  INF)
    (17: VP -> V4  ADJ)
    (18: VP -> V5  PP)
    (19: SS -> that  S)
    (20: INF -> to  VP)
    (21: PP -> P  NP)
    (22: PARTP -> VPART  PP)
    (23: SSNP -> that  SNP)
    (24: SNP -> VP)
    (25: SNP -> NP  VPNP)
    (26: VPNP -> V1)
    (27: VPNP -> V2  PP)
    (28: VPNP -> V3  INFNP)
    (29: VPNP -> AUX  VPNP)
    (30: INFNP -> to  VPNP)
$fg
```

Las tablas de coberturas, derivaciones y adyacencias generadas para esta gramática aparecen listadas a continuación:

Tabla de coberturas:

| Simb | CobI | CobD |
|---|---|---|
| ADJ | (11) | (17) |
| AUX | (12,29) | () |
| DET | (3) | () |
| INF | () | (16) |
| INFNP | () | (28) |
| N | (10) | (10) |
| NOM | (4) | (3,4,11) |
| NP | (1,6,7,8,9,25) | (14,21) |
| P | (21) | () |
| PARTP | () | (7) |
| PNOUN | (5) | (5) |
| PP | () | (8,15,18,22,27) |
| S | () | (19) |
| SNP | () | (23) |
| SS | (2) | () |
| SSNP | () | (6) |
| V0 | (13) | (13) |
| V1 | (14,26) | (26) |
| V2 | (15,27) | () |
| V3 | (16,28) | () |
| V4 | (17) | () |
| V5 | (18) | () |
| VP | (24) | (1,2,12,20,24) |
| VPART | (22) | () |
| VPNP | () | (25,29,30) |
| s | () | (9) |
| that | (19,23) | () |
| to | (20,30) | () |

Tabla de derivaciones:

| Simb | DerI | DerD | AdyP |
|---|---|---|---|
| ADJ | <NOM,NP,S,DET,SNP> | <VP,S,INF,SNP,SS,SSNP, NP,PP,PARTP,VPNP,INFNP> | <NOM> |
| AUX | <VP,VPNP,SNP> | <> | <VP,VPNP> |
| DET | <NP,S,SNP> | <> | <NOM> |
| INF | <> | <VP,S,SNP,SS,SSNP,NP,PP, PARTP,VPNP,INFNP> | <> |
| INFNP | <> | <VPNP,SNP,SSNP,NP,VP,PP,S, INF,PARTP,SS> | <> |
| N | <NOM,NP,S,DET,SNP> | <NOM,NP,VP,PP,S,INF,SNP, PARTP,VPNP,SS,SSNP,INFNP> | <> |
| NOM | <NP,S,DET,SNP> | <NP,VP,PP,S,INF,SNP,PARTP, VPNP,SS,SSNP,INFNP> | <> |

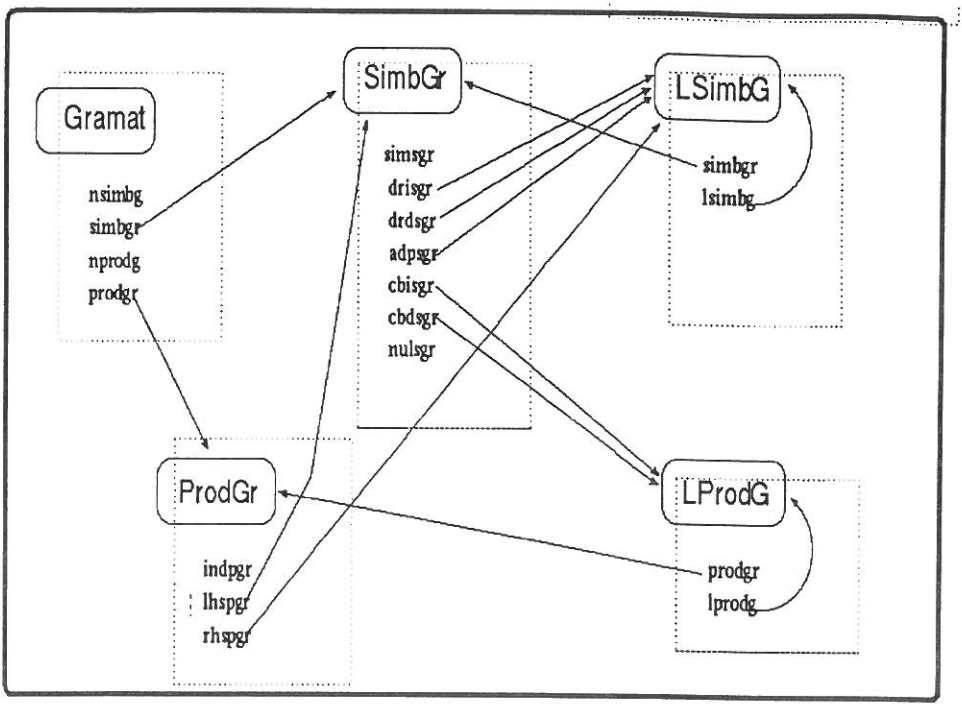
| | | | |
|---|---|---|---|
| NP | <S,DET,SNP> | <VP,PP,S,INF,SNP,PARTP, VPNP,SS,SSNP,INFNP> | <VP,S,NP, PARTP,PP, S,PP,VPNP> |
| P | <PP> | <> | <NP> |
| PARTP | <> | <NP,VP,PP,S,INF,SNP,VPNP, SS,SSNP,INFNP> | <> |
| PNOUN | <NP,S,DET,SNP> | <NP,VP,PP,S,INF,SNP,PARTP, VPNP,SS,SSNP,INFNP> | <> |
| PP | <> | <NP,VP,PARTP,VPNP,S,INF, SNP,SS,SSNP,INFNP> | <> |
| S | <> | <SS> | <> |
| SNP | <> | <SSNP,NP,VP,PP,S,INF,PARTP, VPNP,SS,INFNP> | <> |
| SS | <S> | <> | <VP> |
| SSNP | <> | <NP,VP,PP,S,INF,SNP,PARTP, VPNP,SS,INFNP> | <> |
| V0 | <VP,SNP> | <VP,S,INF,SNP,SS,SSNP,NP, PP,PARTP,VPNP,INFNP> | <> |
| V1 | <VP,VPNP,SNP> | <VPNP,SNP,INFNP,SSNP,NP, VP,PP,S,INF,PARTP,SS> | <NP> |
| V2 | <VP,VPNP,SNP> | <> | <NP,PP> |
| V3 | <VP,VPNP,SNP> | <> | <INF,INFNP> |
| V4 | <VP,SNP> | <> | <ADJ> |
| V5 | <VP,SNP> | <> | <PP> |
| VP | <SNP> | <S,INF,SNP,SS,SSNP,NP,PP, PARTP,VPNP,INFNP> | <> |
| VPART | <PARTP> | <> | <PP> |
| VPNP | <> | <SNP,INFNP,SSNP,NP,VP,PP,S, INF,PARTP,SS> | <> |
| s | <> | <DET> | <> |
| that | <SS,SSNP,S> | <> | <S,SNP> |
| to | <INF,INFNP> | <> | <VP,VPNP> |

## *2.5.- Memoria Simbólica*

Una de las claves para lograr un parser eficiente es la optimización del modelo de representación de la información. En concreto, las aplicaciones que requieren un nivel de manipulación fundamentalmente simbólico hacen que la mayor parte del tiempo de cómputo se dedique a la manipulación de cadenas de caracteres (comparación, copia, etc.). Para evitar esta carga computacional se ha diseñado un modelo de representación, al que se ha denominado *memoria simbólica*, que reduce considerablemente todas las operaciones de comparación y copia de cadenas permitiendo una manipulación muy eficiente del modelo gramatical presentado anteriormente.

La idea básica es que durante la compilación de la gramática se obtiene una estructura donde cada símbolo aparece una sóla vez. Las dos consecuencias fundamentales son que se reducen los costosos y lentos procesos de comparación de cadenas de caracteres a una mera

comparación de dos números (dos cadenas son la misma si efectivamente se encuentran e la misma posición de memoria), y en segundo lugar, se eliminan completamente la operaciones de búsqueda (todo objeto, símbolo o producción) conoce las direcciones d todos los objetos que debe usar.




En la figura anterior se representan gráficamente las estructuras de datos usadas y las relaciones entre ellas: Una gramática (estructura *Gramat*) está formada por un conjunto de símbolos; con el fin de lograr una mayor eficiencia, éstos se almacenan mediante un puntero a una matriz de punteros a estructuras *SimbGr* previamente ordenadas. Asimismo una gramática contiene *nprodg* producciones, que se almacenan como un puntero (*prodgr*) a una matriz de punteros a estructuras *ProdGr*. Cada producción se identifica por un índice (*indpgr*) y apunta a un símbolo como "left-hand side" (*lhspgr*) y a una lista de símbolos (estructura *LSimbG*) como "right-hand side" (*rhspgr*). Cada símbolo (estructura *SimbGr*) contiene el identificador del símbolo (*simsgr*) junto con la información relativa a las tablas de coberturas (*cbisgr* y *cbdsgr*), de derivaciones (*drisgr* y *drdsgr*) y de adyacencias (*adpsgr*). Asimismo para cada símbolo el parámetro *nulsgr* indica si el símbolo es anulable, es decir,

se puede obtener mediante la aplicación de una o más ε-producciones (producciones cuyo lado derecho es nulo). Finalmente las estructuras *LSimbG* y *LProdG* son listas recursivas de estructuras *SimbGr* y *ProdGr* respectivamente.

## *3.- Parsing Dirigido por Eventos*

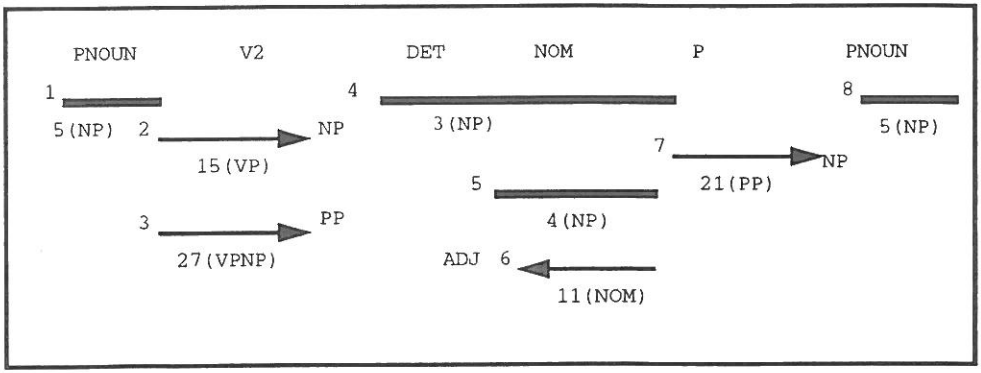
### *3.1.- Eventos y Coberturas*

La idea consistente en utilizar eventos o agentes de análisis como primitivas que dirigen la estrategia del parser no es nueva (véase por ejemplo [Small 1987]), y en general es común en algoritmos paralelos [Nijholt 1991].

Consideremos la gramática definida en el apartado 2.4. Supongamos que pretendemos analizar la entrada *Joe bought the book for Susan*. La entrada al parser estará constituida por la lista de categorías sintácticas asociadas con cada uno de los términos de la entrada. De acuerdo con la especificación de Shieber [1983], la entrada anterior se convierte para el parser en *PNOUN V2 DET NOM P PNOUN*. El parser permite manipular correctamente la ambigüedad léxica, para ello espera que el componente léxico devuelva las entradas de la forma *Cat1//Cat2//...//CatN* especificando que una palabra posee N posibles categorías sintácticas. El parser lanzará eventos para cada una de las posibilidades de la disyunción. Asimismo, el componente léxico implementado es capaz de tratar entradas compuestas (un bloque de varias palabras compone una única categoría sintáctica), haciendo transparente para el parser este tipo de fenómenos.

Para cada nodo (formado por una categoría sintáctica o una disyunción de categorías) de la cadena de análisis, el parser lanza todos los análisis posibles usando la información disponible en las tablas de coberturas de la gramática.

En la siguiente figura se muestra el estado que se obtiene tras la aplicación de las coberturas sobre los símbolos que forman la cadena que entra al parser.




Las flechas representan producciones que aún no han completado su lado derecho, mientras que las barras representan producciones completas. Cada flecha o barra es en

realidad un evento o agente de análisis; en la figura los números que aparecen a la izquier de cada barra o flecha indica el orden en el que se han ido generando los eventos. Ca evento va asociado con una producción que se indica justo debajo de la flecha o bar añadiendo al número de producción correspondiente el lado izquierdo de ésta, es decir, símbolo que se obtendría si se aplicase la producción. Como se puede deducir a partir de l estructuras de datos presentadas en el apartado 2.5, toda esta información está disponible una forma inmediata sin que sea necesario ningún proceso de búsqueda a través de gramática.

Para el caso de las producciones no completas (flechas), junto al final de la flecha se l indicado el símbolo que el evento espera obtener en esa posición. Así por ejemplo, el even número 2 está intentando aplicar la producción número 15 (cuyo lado izquierdo es VP), evento se ha lanzado desde la cobertura izquierda del símbolo V2 de la cadena de entrada, actualmente se encuentra esperando un símbolo NP donde realmente existe un símbo DET. De esta forma, para una producción no completa, se pueden definir dos símbolos: símbolo esperado (exigido por la producción para continuar expandiéndose) y el símbo real (que es el que efectivamente aparece en la cadena de entrada).

### *3.2.- Eliminación de Eventos*

A partir del estado de análisis mostrado en la figura anterior se obtiene un cierto nivel d ambigüedad (consecuencia necesaria de la ambigüedad estructural que debe permit cualquier gramática para un lenguaje natural), y el objetivo del parser consistirá en reduc este nivel de ambigüedad, lo que se traducirá en un aumento de la eficiencia. Es decir, objetivo del parser será la eliminación de eventos. Con este objetivo, el parser incluye un serie de filtros que aprovechan las tablas de derivaciones y adyacencias para la eliminació de eventos en una fase temprana del análisis.

Como se verá más adelante, los análisis de derivaciones, de enlaces a partir d adyacencias y la propagación de restricciones son mecanismos con una gran capacidad d eliminación de eventos. Los eventos eliminados son todos aquellos que de acuerdo con e estado de análisis disponible no tendrán éxito, por tanto, es sumamente importante conta con toda la información acerca de los posibles análisis para una entrada dada. Esta es l justificación para el modelo bidireccional implementado.

A continuación se describen los filtros de eliminación de eventos:

#### *3.2.1.- Análisis de Duplicidad*

El análisis de duplicidad es una consecuencia del modelo bidireccional de análisis. S pueden distinguir dos comportamientos para este mecanismo. En el primero de ellos no s trata de un filtro para la eliminación de eventos sino de una precondición para su generación Así, para el ejemplo considerado en el apartado 3.1, de acuerdo con la tabla de cobertura del símbolo NOM, se debería haber creado un nuevo evento que aplicaría la producción 3 d derecha a izquierda. Sin embargo, como ya existe un evento (el número 4) que aplica l misma producción sobre el mismo intervalo de símbolos de la cadena de análisis, este nuev

evento no se llega ni siquiera a lanzar. En otras ocasiones, el modelo bidireccional sí que puede lanzar dos eventos que se corresponden al mismo análisis. Sin embargo, para mantener la consistencia del sistema hay que permitir los dos eventos, y fusionarlos (mediante el análisis de duplicidad) en fases posteriores.

### *3.2.2.- Análisis de Derivaciones*

Estos análisis se aplican sobre los extremos de los eventos no completos. Intuitivamente se trata de comprobar si el símbolo esperado es derivable a partir del símbolo real. Podemos distinguir entre un Control de Derivaciones por la Izquierda (evento de izquierda a derecha) o un Control de Derivaciones por la Derecha (evento de derecha a izquierda).

- Control de Derivaciones por la Izquierda

Supongamos que un evento está aplicando la producción $\delta \rightarrow \delta 1 \ldots \delta_n$, sobre la superficie $\Gamma \ \delta_1 \ldots \delta_i \ \omega \ \Omega$, con $1 \leq i < n$. Segun la definición de evento, éste estaría realizando una predicción del símbolo $\delta_{i+1}$ (necesario) sobre el símbolo (real) $\omega$. En este caso se comprueba si $\delta_{i+1} \in \text{DerI}(\omega)$.

- Control de Derivaciones por la Derecha

Supongamos que un evento está aplicando la producción $\delta \rightarrow \delta 1 \ldots \delta_n$ sobre la superficie $\Gamma \ \varphi \ \delta_i \ldots \delta_n \ \Omega$ con $1 < i \leq n$. El evento realiza una predicción del símbolo (necesario) $\delta_{i-1}$ sobre el símbolo (real) $\varphi$. En este caso se comprueba si $\delta_{i-1} \in \text{DerD}(\varphi)$.

Si el control de derivaciones falla, se puede eliminar el evento. Ahora bien, si el símbolo esperado se puede obtener tras la aplicación de una o más $\varepsilon$-producciones (*nulsgr*), y el control de derivaciones falla, el algoritmo espande el evento sobre el siguiente símbolo de la producción.

Este filtro permite eliminar para la frase ejemplo del apartado 3.1 los eventos número 3 (no es posible derivar por la izquierda PP a partir de DET) y número 6 (no es derivable por la derecha ADJ a partir de DET).

### *3.2.3.- Análisis de Adyacencias*

Sobre los límites cerrados de un evento (límites en los que el evento no realiza predicciones) se comprueba la posibilidad de adyacencia entre el símbolo que generaría la aplicación del evento y los símbolos adyacentes de la superficie de análisis. Se pueden distinguir dos situaciones según si el análisis se realiza en el límite izquierdo o en el derecho del evento.

Supongamos que un evento completo está aplicando la producción $\varphi \rightarrow \varphi_1 \ldots \varphi_n$ sobre la superficie de análisis $\Delta\Gamma\Omega$, de forma que $\Gamma \equiv \varphi_1 \ldots \varphi_n$

- Control de Adyacencias por la Izquierda:

Si $\Delta$ es una cadena no vacía ($\Delta \equiv \delta_1 \ldots \delta_x$), se comprueba si existe algún símbolo $\delta$ en el conjunto $\{\delta_x\} \cup \text{DerD}(\delta_x)$, tal que $\varphi \in \text{AdyP}(\delta)$.

• Control de Adyacencia por la Derecha:

Si $\Omega$ es una cadena no vacía ($\Omega \equiv \omega 1 \ldots \omega_y$), se comprueba si existe algún símbolo en el conjunto $\{\omega 1\} \cup \text{DerI}(\omega 1)$, tal que $\omega \in \text{AdyP}(\varphi)$.

Obviamente si el evento es incompleto, sólo se comprueban las adyacencias bien por izquierda o bien por la derecha según el sentido del evento.

Si alguno de los controles de adyacencia falla, el evento se elimina.

Para el ejemplo considerado, este control elimina el evento número 5, ya que no so adyacentes los símbolos DET (o alguna de sus derivaciones por la derecha) y NP (resultad de aplicar el evento).

### *3.2.4.- Propagación de Restricciones*

Los filtros para análisis de derivaciones y adyacencias estudiados se aplican de un forma local, es decir, en los mismos límites de los eventos. No obstante, existen mucha situaciones en las que el éxito de un evento depende de que otro evento no adyacente co este se aplique.

El mecanismo de propagación de restricciones lanza los controles de derivaciones adyacencias, pero ahora no de una forma local, sino a lo largo de la cadena de análisis permitiendo detectar este tipo de dependencias.

### *3.3.- Ejecución de Eventos*

El módulo de ejecución de eventos utiliza la información disponible en el mapa d análisis, y de acuerdo con un sencillo algoritmo que prioriza el análisis de los componente deterministas lanza la ejecución de los eventos.

Siguiendo con el ejemplo, tras la aplicación de los filtros anteriores, el mapa de análisi para la oración *Joe bought the book for Susan* queda así:


A partir de aquí se pueden ejecutar los eventos número 1, 4 y 8 de una forma completamente determinista, y continuar con el proceso de análisis.

No obstante, existen gramáticas estructuralmente ambigüas para las que aparecerár

análisis no deterministas, lo que gráficamente es equivalente a disponer de más de un evento completo sobre un mismo nodo de la superficie de análisis. Para este caso el parser utiliza las ideas ya clásicas de compactación de subestructuras generando un bosque (*parse forest*) con todos los análisis posibles.

## *4.- 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 denominado *memoria simbólica* adecúa la representación de una gramática a las necesidades del parser, eliminando las operaciones de búsqueda y reduciendo considerablemente las de comparación.

# BIBLIOGRAFIA

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.

Alexandersson, J.; Maier, E.; Reithinger, N. 1995. "A Robust and Efficient Three-Layered Dialogue Component for a Speech-to-Speech Translation System". *Proceedings of the EACL*.

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.

Bresnan, J. ed. 1982. *The Mental Representation of Grammatical Relations*. Cambridge, Mass.: MIT Press.

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-Coverag Unification Grammars". *CMP-LG e-print archive cmp-lg/9405033.*

Carter, D. 1990. "Efficient Disjunctive Unification for Bottom-Up Parsing". *COLING-90,* 70-75.

Dowding, J.; Moore, R.; Andry, F.; Moran, D. 1994. "Interleaving Sintax and Semantics in an Efficie 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:** 45 460. (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.* Lo 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,* Febre 1994, 2-7.

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

Martin, W.A.; Church, K.W.; Patil, R.S. 1987. "Preliminary Analysis of a Breadth-First Parsin 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 th 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.* Kluwe Academic Publishers.

Pereira, F.C.N. 1985. "A New Characterization of Attachment Preferences". En Dowty, D.R. Karttunen, L.; Zwicky, A.M. 1985. *Natural Language Parsing.* Cambridge: Cambridge Universit Press, 307-319.

Quesada, J.~F. 1993. Un algoritmo para el análisis de gramáticas libres de contexto orientado a procesamiento del lenguaje natural. En Martín Vide, C. ed. 1993. *Lenguajes Naturales y Lenguaje Naturales IX.* Barcelona: PPU. 399--406.

Quesada, J.~F. 1994. O2RTED: La orientación a objetos y la dirección por eventos aplicadas al análisis sintáctico en tiempo real. En Martín Vide, C. ed. 1994. Lenguajes Naturales y Lenguajes Naturales X. Barcelona: PPU. 557--564.

Quesada, J.~F. 1996. Un Modelo Robusto y Eficiente para el Análisis Sintáctico de Lenguajes Naturales mediante Arboles Múltiples Virtuales. *Procesamiento del Lenguaje Natural,* **19,** 14--29.

Quesada, J.~F. 1997. *El algoritmo SCP de análisis sintáctico mediante propagación de restricciones.* Tesis Doctoral. Julio 1997. Universidad de Sevilla.

Quesada, J.~F. 1997. A General, Sound and Efficient Natural Language Parsing Algorithm based on Syntactic Constraints Propagation. *Proceedings of the VII Conference AEPIA'97,* 775--786.

Quesada, J.~F. & J.~G. Amores. 1997. Linguistic and Computational Advantages of Bidirectiona Bottom--Up Parsing with Top--Down Predictions. *Procesamiento del Lenguaje Natural*, **21**, 137-- 146.

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

Shieber, S. M. 1983. "Sentence disambiguation by a shift-reduce parsing technique". *IJCAI-83*, 699-703.

Shieber, S.M. 1986. *An Introduction to Unification-Based Approaches to Grammar.* Stanford, California: Center for the Study of Language and Information, CSLI Lecture Notes Series.

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)**.