Capítulo 3

Algoritmos de análisis sintáctico para TAG

Se describen varios de los algoritmos de análisis sintáctico que han sido propuestos para las gramáticas de adjunción de árboles desde que éstas fueron por descritas por primera vez en [93]. La mayor parte de estos algoritmos están basados en algoritmos de análisis sintáctico independientes del contexto extendidos para el tratamiento de la adjunción. Asimismo es de destacar que prácticamente todos estos algoritmos utilizan técnicas de programación dinámica, obteniendo una complejidad polinómica de orden $\mathcal{O}(n^6)$, donde n es la longitud de la cadena de entrada. La aportación de este capítulo es múltiple: se proporciona un camino evolutivo que relaciona los algoritmos de análisis de TAG más populares; se describen por vez primera algunos algoritmos de análisis de TAG, como por ejemplo, las diferentes versiones de los algoritmos de tipo Earley ascendente y las versiones propuestas de los algoritmos de tipo Earley sin la propiedad del prefijo válido; por último, se realiza una descripción conjunta de la mayor parte de los algoritmos de análisis existentes para TAG. Este capítulo está basado en [8, 9].

3.1. Introducción

A la hora de describir los algoritmos de análisis para TAG, tenemos que elegir una representación adecuada para indicar el reconocimiento parcial de los árboles elementales. En las gramáticas independientes del contexto se utilizan habitualmente producciones con punto para separar la parte de la producción que ya ha sido procesada de la que no lo ha sido aún. En los algoritmos que proceden unidireccionalmente, un solo punto es suficiente, mientras que en aquellos que proceden bidireccionalmente se necesitan dos puntos [189]. En el caso de TAG, al no ser las estructuras elementales producciones sino árboles, deberemos representar árboles con punto. Existen varias alternativas:

1. Al igual que en las gramáticas independientes del contexto se escriben producciones completas con punto, en TAG se podría escribir cada árbol elemental completo con su punto correspondiente. Un ejemplo de utilización de este tipo de notación puede encontrarse en [176].

2. Si identificamos unívocamente todo elemento en una producción independiente del contexto, para lo cual podemos utilizar subíndices correspondientes al número de la producción y a la posición que cada elemento ocupa en al producción de tal modo que la producción $k : N \rightarrow MPQ$ se representaría como $N_{k,0} \rightarrow N_{k,1}N_{k,2}N_{k,3}$, podemos representar una producción con punto simplemente indicando un elemento de la producción contiguo al

punto e indicando si este se encuentra situado a derecha o izquierda de dicho elemento. Del mismo modo, podemos indicar la posición del punto en un árbol elemental de una TAG indicando simplemente un nodo del árbol adyacente al punto y la posición relativa del punto respecto a dicho nodo: arriba-izquierda, abajo-izquierda, abajo-derecha o arriba-derecha. Este es el tipo de notación utilizada en [168].

3. Como un árbol elemental γ puede considerarse constituido por un conjunto de producciones independientes del contexto $\mathcal{P}(\gamma)$, podemos indicar la posición del punto en el árbol simplemente indicando la posición en la producción correspondiente. Este tipo de representación recibe el nombre de *multicapa* en [64].

4. Aplicar un aplanamiento a los árboles, esto es, una conversión de los mismos a cadenas de caracteres parentizadas en las cuales cada nuevo nivel de paréntesis indica un nivel adicional de profundidad en el árbol elemental. Este tipo de representación, que recibe el nombre de *plana* en [64], implica que cada árbol inicial se representa únicamente por una cadena de caracteres mientras que cada árbol auxiliar se representa por dos cadenas correspondientes a las partes situadas a la izquierda y a la derecha de la espina. Dichas cadenas pueden ser utilizadas como partes derechas de las producciones de una gramática independiente del contexto que permitiría representar de modo conciso los árboles elementales de una gramática de adjunción [64].

La primera alternativa presenta el inconveniente de que las representaciones lineales de árboles suelen dificultar su comprensión. Alternativamente podrían utilizarse representaciones pictóricas de los árboles, que si bien son incómodas a lo hora de describir los pasos deductivos del algoritmo pueden ser útiles para representar gráficamente el comportamiento del algoritmo. La segunda alternativa, aunque muy manejable y concisa tiene el inconveniente de que es necesario recurrir a las descripciones de los árboles elementales para poder ver el contexto en el que se está aplicando cada paso del algoritmo. La tercera alternativa resuelve este problema al proporcionar un contexto formado por los hermanos del nodo considerado, que si bien es limitado, proporciona al lector la información suficiente la mayor parte de las veces. La desventaja es que normalmente deberemos añadir pasos deductivos extra que realizan el recorrido del árbol en un orden determinado. La cuarta alternativa, aunque en principio supone una reducción del número de producciones independientes al contexto, en la práctica su utilización no implica generalmente una reducción del número de pasos deductivos utilizados para describir el procesamiento realizado por los algoritmos de análisis sintáctico[1] y contribuye a oscurecer la notación.

En nuestro caso hemos preferido utilizar la representación multicapa, por lo que indicaremos la posición del punto en un árbol mediante una producción $N \rightarrow \delta \bullet \nu$, donde $\delta\nu$ son los hijos de N. En el caso de los elementos del lado derecho de las producciones cuyas etiquetas pertenecen a $V_T \cup \epsilon$, puesto que no pueden tener descendientes ni son susceptibles de ser nodos de adjunción, consideraremos directamente su etiqueta a la hora de describir las producciones, en lugar del nodo propiamente dicho[2]. Adicionalmente y con el fin de simplificar los esquemas de análisis de los algoritmos más complejos, seguiremos el enfoque de [125] de considerar la producción adicional $\top \rightarrow \mathbf{R}^\alpha$ para cada árbol inicial α y las dos producciones adicionales siguientes para cada árbol auxiliar β: $\top \rightarrow \mathbf{R}^\beta$ y $\mathbf{F}^\beta \rightarrow \bot$, donde $\mathbf{R}^\beta$ y $\mathbf{F}^\beta$ se refieren a los nodos raíz y pie de β, respectivamente. Con el fin de no modificar la capacidad generativa de las gramáticas,

[1] Como ejemplo, podemos comparar el algoritmo descrito en [64] con el representado por el esquema 3.7.

[2] Con esta convención, que se adopta porque simplifica considerablemente la notación, la única forma de distinguir diferentes hojas de un árbol elemental con la misma etiqueta es referenciando los respectivos padres o la producción independiente del contexto de la que forman parte.

los nuevos nodos $\top$ y $\bot$ no pueden ser nodos de adjunción, por lo cual la nueva gramática se asemeja a una TAG *limpia*[3].

Con respecto a las restricciones de adjunción, $\beta \in \text{adj}(N^\gamma)$ si $\beta \in \boldsymbol{A}$ puede ser adjuntado en N^γ. Si $\beta = \textbf{nil}$ entonces la adjunción en el nodo N^γ es opcional. Si **nil** es el único valor que retorna la función para un nodo N^γ, entonces no es posible realizar ninguna adjunción en dicho nodo.

Ejemplo 3.1 El árbol inicial α de la figura 3.1 se representa en notación multicapa mediante el siguiente conjunto de producciones independientes del contexto:

$$\top \rightarrow \langle \alpha, 0 \rangle$$

$$\langle \alpha, 0 \rangle \rightarrow a \ \langle \alpha, 2 \rangle$$

$$\langle \alpha, 2 \rangle \rightarrow c$$

El árbol auxiliar β de la misma figura se representa mediante el siguiente conjunto de producciones:

$$\top \rightarrow \langle \beta, 0 \rangle$$

$$\langle \beta, 0 \rangle \rightarrow a \ \langle \beta, 2 \rangle$$

$$\langle \beta, 2 \rangle \rightarrow \langle \beta, 2{,}1 \rangle \ c$$

$$\langle \beta, 2{,}1 \rangle \rightarrow \bot$$

En ambos casos $\langle \alpha, g \rangle$ denota el nodo de α que ocupa la posición g según el direccionamiento de Gorn[4]. ¶

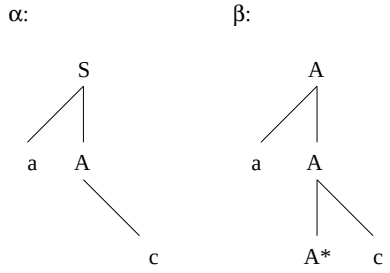

Figura 3.1: Ejemplos de árbol inicial y auxiliar

La relación $\Rightarrow$ de derivación sobre $\mathcal{P}(\gamma)$ se define como $\delta \Rightarrow \nu$ si existen $\delta', \delta'', M^\gamma, \upsilon$ tal que $\delta = \delta' M^\gamma \delta''$, $\nu = \delta' \upsilon \delta''$ y existe una producción $M^\gamma \rightarrow \upsilon \in \mathcal{P}(\gamma)$. Mediante $\overset{*}{\Rightarrow}$ denotaremos el cierre reflexivo y transitivo de $\Rightarrow$.

Sea $\mathcal{P}(\mathcal{T}) = \bigcup_{\gamma \in \boldsymbol{I} \cup \boldsymbol{A}} \mathcal{P}(\gamma)$ el conjunto de todas las producciones independientes del contexto presentes en los árboles de una gramática de adjunción de árboles $\mathcal{T}$. Con el fin de ser capaces de representar derivaciones que incluyan adjunciones extendemos la relación $\Rightarrow$ a $\mathcal{P}(\mathcal{T})$, de tal modo que $\delta \overset{*}{\Rightarrow} \nu$ si existen $\delta', \delta'', M^\gamma, \upsilon$ tal que $\delta = \delta' M^\gamma \delta''$, $\mathbf{R}^\beta \overset{*}{\Rightarrow} \upsilon_1 \mathbf{F}^\beta \upsilon_3$, $\beta \in \text{adj}(M^\gamma)$, $M^\gamma \rightarrow \upsilon_2$ y $\nu = \delta' \upsilon_1 \upsilon_2 \upsilon_3 \delta''$.

[3]Poller y Becker [149] denominan TAG *limpias* a las gramáticas de adjunción en las que los nodos raíz de los árboles elementales y los nodos pie de los árboles auxiliares tienen restricciones de adjunción nulas.

[4]En el direccionamiento de Gorn se utiliza 0 para referirse al nodo raíz, k para referirse al k-ésimo hijo del nodo raíz y $p.q$ para referirse al q-ésimo hijo del nodo con dirección p.

La mayoría de los algoritmos de análisis sintáctico diseñados para TAG se corresponden con adaptaciones de algoritmos para el análisis de gramáticas independientes del contexto y por lo tanto recibirán el mismo nombre en ambos casos. Solamente en aquellos casos en los que pueda existir confusión diferenciaremos explícitamente entre unos y otros[5]. Para describir los diferentes algoritmos de análisis sintáctico haremos uso de *esquemas de análisis*, una estructura para realizar descripciones de alto nivel de algoritmos de análisis desarrollada por Sikkel en [189] y que se describe en el apéndice A.

3.2. Algoritmo de tipo CYK

A continuación mostramos una extensión para TAG del algoritmo CYK de análisis sintáctico (ver sección B.1) basado en el algoritmo descrito en [209, 206] pero con algunas correcciones. Asumiremos que todo nodo de un árbol elemental de la gramática tiene a lo sumo dos descendientes. Este condicionante puede verse como una trasposición de la forma normal de Chomsky [85] al caso de las gramáticas de adjunción.

El reconocimiento de los nodos de un árbol elemental se realiza aplicando casi literalmente el algoritmo para el caso independiente del contexto. Sin embargo ahora es preciso definir cómo reconocer la operación de adjunción de forma totalmente ascendente. Para ello se definen ítems de la forma

$$\left\{ [N^\gamma, i, j \mid p, q \mid adj] \mid \begin{array}{ll} N^\gamma \overset{*}{\Rightarrow} a_{i+1} \dots a_p \, \mathbf{F}^\gamma \, a_{q+1} \dots a_j \overset{*}{\Rightarrow} a_{i+1} \dots a_j & \text{sii } (p,q) \neq (-,-) \\ N^\gamma \overset{*}{\Rightarrow} a_{i+1} \dots a_j & \text{sii } (p,q) = (-,-) \end{array} \right\}$$

que contienen un nuevo componente *adj* con respecto a los ítems definidos en [209, 206], que toma su valor del conjunto $\{\text{true}, \text{false}\}$ y que indica

- si su valor es *true* significa que el ítem es el resultado de una operación de adjunción ya totalmente realizada;
- si valor es *false* significa que el ítem no es el resultado de una adjunción.

Con ello podemos limitar a una sola la cantidad de adjunciones que se pueden realizar sobre cada nodo de un árbol elemental y podemos también asegurar el cumplimiento de las restricciones de adjunción obligatoria, ajustándonos de este modo a lo establecido en el formalismo de las gramáticas de adjunción [175].

Esquema de análisis sintáctico 3.1 El sistema de análisis $\mathbb{P}_{\text{CYK}}$ que se corresponde con el algoritmo CYK para una gramática de adjunción de árboles $\mathcal{T}$ y una cadena de entrada $a_1 \dots a_n$ se define como sigue:

$$\mathcal{I}_{\text{CYK}} = \left\{ [N^\gamma, i, j \mid p, q \mid adj] \mid \begin{array}{l} \text{etiqueta}(N^\gamma) \in V_N, \ \gamma \in \boldsymbol{I} \cup \boldsymbol{A}, \ \ 0 \leq i \leq j, \\ (p,q) \leq (i,j), \ adj \in \{\text{true}, \text{false}\} \end{array} \right\}$$

$$\mathcal{H}_{\text{CYK}} = \left\{ \ [a, i-1, i] \mid a = a_i, 1 \leq i \leq n \ \right\}$$

$$\mathcal{D}^{\text{Scan}}_{\text{CYK}} = \frac{[a, i, i+1]}{[N^\gamma, i, i+1 \mid -, - \mid \text{false}]} \ \ N^\gamma \to a \in \mathcal{P}(\gamma)$$

[5] En el apéndice B se describen sucintamente los algoritmos de análisis sintántico para gramáticas independientes del contexto que son relevantes en este capítulo

$$\mathcal{D}^{\epsilon}_{\text{CYK}} = \frac{}{[N^\gamma, i, i \mid -,- \mid \text{false}]} \quad N^\gamma \to \epsilon \in \mathcal{P}(\gamma)$$

$$\mathcal{D}^{\text{Foot}}_{\text{CYK}} = \frac{}{[\mathbf{F}^\gamma, i, j \mid i, j \mid \text{false}]}$$

$$\mathcal{D}^{\text{LeftDom}}_{\text{CYK}} = \frac{[M^\gamma, i, k \mid p, q \mid adj1],\ [P^\gamma, k, j \mid -,- \mid adj2]}{[N^\gamma, i, j \mid p, q \mid \text{false}]} \quad \begin{array}{l} N^\gamma \to M^\gamma P^\gamma \in \mathcal{P}(\gamma), \\ M^\gamma \in \text{espina}(\gamma), \\ adj1 = \text{false sii } \mathbf{nil} \in \text{adj}(M^\gamma), \\ adj2 = \text{false sii } \mathbf{nil} \in \text{adj}(P^\gamma) \end{array}$$

$$\mathcal{D}^{\text{RightDom}}_{\text{CYK}} = \frac{[M^\gamma, i, k \mid -,- \mid adj1],\ [P^\gamma, k, j \mid p, q \mid adj2]}{[N^\gamma, i, j \mid p, q \mid \text{false}]} \quad \begin{array}{l} N^\gamma \to M^\gamma P^\gamma \in \mathcal{P}(\gamma), \\ P^\gamma \in \text{espina}(\gamma), \\ adj1 = \text{false sii } \mathbf{nil} \in \text{adj}(M^\gamma), \\ adj2 = \text{false sii } \mathbf{nil} \in \text{adj}(P^\gamma) \end{array}$$

$$\mathcal{D}^{\text{NoDom}}_{\text{CYK}} = \frac{[M^\gamma, i, k \mid -,- \mid adj1],\ [P^\gamma, k, j \mid -,- \mid adj2]}{[N^\gamma, i, j \mid -,- \mid \text{false}]} \quad \begin{array}{l} N^\gamma \to M^\gamma P^\gamma \in \mathcal{P}(\gamma), \\ N^\gamma \notin \text{espina}(\gamma), \\ adj1 = \text{false sii } \mathbf{nil} \in \text{adj}(M^\gamma), \\ adj2 = \text{false sii } \mathbf{nil} \in \text{adj}(P^\gamma) \end{array}$$

$$\mathcal{D}^{\text{Unary}}_{\text{CYK}} = \frac{[M^\gamma, i, j \mid p, q \mid adj]}{[N^\gamma, i, j \mid p, q] \mid \text{false}]} \quad \begin{array}{l} N^\gamma \to M^\gamma \in \mathcal{P}(\gamma), \\ adj = \text{false sii } \mathbf{nil} \in \text{adj}(M^\gamma) \end{array}$$

$$\mathcal{D}^{\text{Adj}}_{\text{CYK}} = \frac{[\mathbf{R}^\beta, i', j' \mid i, j \mid adj],\ [N^\gamma, i, j \mid p, q \mid \text{false}]}{[N^\gamma, i', j' \mid p, q \mid \text{true}]} \quad \beta \in \boldsymbol{A},\ \beta \in \text{adj}(N^\gamma)$$

$$\mathcal{D}_{\text{CYK}} = \mathcal{D}^{\text{Scan}}_{\text{CYK}} \cup \mathcal{D}^{\epsilon}_{\text{CYK}} \cup \mathcal{D}^{\text{Foot}}_{\text{CYK}} \cup \mathcal{D}^{\text{LeftDom}}_{\text{CYK}} \cup \mathcal{D}^{\text{RightDom}}_{\text{CYK}} \cup \mathcal{D}^{\text{NoDom}}_{\text{CYK}} \cup \mathcal{D}^{\text{Unary}}_{\text{CYK}} \cup \mathcal{D}^{\text{Adj}}_{\text{CYK}}$$

$$\mathcal{F}_{\text{CYK}} = \left\{ \ [\mathbf{R}^\alpha, 0, n \mid -,- \mid adj] \mid \alpha \in \boldsymbol{I},\ adj = \text{false sii } \mathbf{nil} \in \text{adj}(\mathbf{R}^\alpha) \ \right\}$$

donde $(p, q) \leq (i, j)$ se cumple si $i \leq p \leq q \leq j$ en el caso de que $(p, q) \neq (-,-)$ y si $i \leq j$ en el caso de que $(p, q) = (-,-)$. §

La definición de las hipótesis realizada en este sistema de análisis sintáctico se corresponde con la estándar y es la misma que se utilizará en los restantes sistemas de análisis del capítulo. Por consiguiente, no nos volveremos a referir explícitamente a ellas.

Los pasos clave en el sistema de análisis $\mathbb{P}_{\text{CYK}}$ son $\mathcal{D}^{\text{Foot}}_{\text{CYK}}$ y $\mathcal{D}^{\text{Adj}}_{\text{CYK}}$, puesto que son los que definen el mecanismo de adjunción. Los demás pasos se encargan de recorrer de forma ascendente los árboles elementales, propagando la información relativa a la porción de la cadena de entrada cubierta por el pie en el caso de los árboles auxiliares. El paso deductivo $\mathcal{D}^{\text{Scan}}_{\text{CYK}}$ permite resolver la limitación planteada en [209, 206] con respecto a nodos frontera etiquetados con ϵ.

El paso deductivo $\mathcal{D}^{\text{Foot}}_{\text{CYK}}$ hace posible el análisis ascendente de los árboles auxiliares, puesto que predice todas las posibles porciones de la cadena de entrada que puede cubrir el pie. Si

no fuese así, sería necesario esperar al reconocimiento total del pie antes de poder continuar el reconocimiento del árbol auxiliar, con lo cual el algoritmo de análisis no sería totalmente ascendente.

Una de las consecuencias de la utilización del paso $\mathcal{D}^{\text{Foot}}_{\text{CYK}}$ es que existirán múltiples análisis diferentes para cada árbol auxiliar, diferentes únicamente en las posición de la cadena de entrada que se *supone* en cada caso que cubre el pie. Pero finalmente sólo uno o unos pocos de esos árboles podrán formar parte del árbol de derivación, aquellos que hayan predicho correctamente el pie. La realización de esta comprobación le corresponde al paso $\mathcal{D}^{\text{Adj}}_{\text{CYK}}$. Efectivamente, cuando se ha alcanzado la raíz de un árbol auxiliar, se comprueba si existe un subárbol de un árbol elemental cuya raíz pueda ser un nodo de adjunción para dicho árbol auxiliar y que además cubra la porción de la cadena de entrada que espera ser cubierta por el pie. En caso de que así sea, se eliminará la posibilidad de realizar otras adjunciones en ese nodo y el análisis continuará ascendiendo por el nuevo árbol elemental. En la figura 3.2 se muestra gráficamente la aplicación de este paso, donde los dos árboles de la izquierda se corresponden con las descripciones de los ítems antecedentes y el árbol de la derecha con la descripción del ítem consecuente. Se utilizan líneas punteadas para indicar partes de árboles elementales que no han sido aún analizadas, mientras que las espinas se resaltan mediante líneas gruesas.

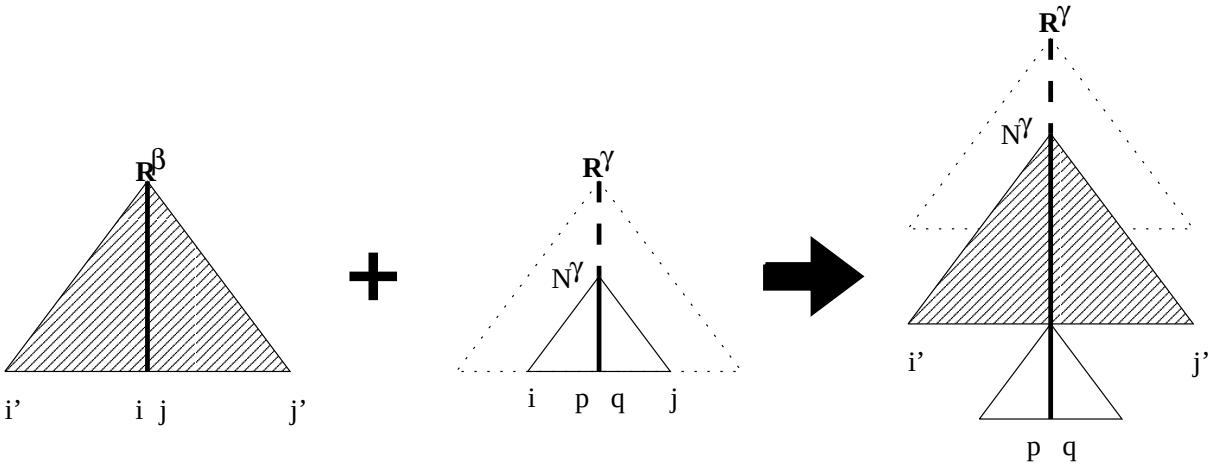

Figura 3.2: Descripción gráfica de un paso $\mathcal{D}^{\text{Adj}}_{\text{CYK}}$

La complejidad temporal del algoritmo con respecto a la longitud n de la cadena de entrada es $\mathcal{O}(n^6)$ y viene dada por el paso deductivo encargado de la adjunción, que debe combinar seis posiciones de la cadena de entrada. La complejidad espacial con respecto a la cadena de entrada es $\mathcal{O}(n^4)$, puesto que cada ítem almacena cuatro posiciones de la cadena de entrada.

3.3. Algoritmos de tipo Earley ascendente

El algoritmo CYK presenta un limitación muy importante: sólo es aplicable a gramáticas en las cuales un nodo no tiene más de dos descendentes. Nuestro objetivo ahora es extender el esquema **CYK** a la clase general de TAG, obteniendo lo que podríamos llamar un esquema Earley ascendente (ver sección B.2) extendido a gramáticas de adjunción de árboles. Debemos reseñar que no conocemos ninguna adaptación anterior para TAG de este algoritmo.

Como primer paso para la realización de un algoritmo de tipo Earley ascendente para gramáticas de adjunción de árboles precisamos introducir puntos en las producciones que repre-

sentan los árboles elementales, por lo que los ítems que utilizaremos tendrán la forma

$$\left\{ \begin{array}{ll} [N^\gamma \to \delta \bullet \nu, i, j \mid p, q \mid adj] \mid & \delta \stackrel{*}{\Rightarrow} a_{i+1} \dots a_p \, \mathbf{F}^\gamma \, a_{q+1} \dots a_j \stackrel{*}{\Rightarrow} a_{i+1} \dots a_j \quad \text{sii } (p,q) \neq (-,-) \\ & \delta \stackrel{*}{\Rightarrow} a_{i+1} \dots a_j \quad \text{sii } (p,q) = (-,-) \end{array} \right\}$$

Los ítems del nuevo esquema de análisis sintáctico, que denominaremos **buE$_1$**, son por tanto un refinamiento de los ítems de **CYK**. Sobre los pasos deductivos aplicaremos también un *refinamiento* puesto que los pasos de tipo LeftDom, RightDom y NoDom serán separados en diferentes pasos de tipo Init y Comp, mientras que los pasos de tipo Unary y ϵ ya no serán necesarios puesto que todas las producciones serán tratadas uniformemente con independencia de su longitud. Finalmente, se realizará una extensión del dominio de las producciones permitiendo árboles con un grado arbitrario de ramificación.

Esquema de análisis sintáctico 3.2 El sistema de análisis $\mathbb{P}_{\text{buE}_1}$ que se corresponde con una primera extensión del algoritmo de Earley ascendente para TAG, dada una gramática de adjunción de árboles $\mathcal{T}$ y una cadena de entrada $a_1 \dots a_n$ se define como sigue:

$$\mathcal{I}_{\text{buE}_1} = \left\{ \begin{array}{ll} [N^\gamma \to \delta \bullet \nu, i, j \mid p, q \mid adj] \mid & N^\gamma \to \delta\nu \in \mathcal{P}(\gamma), \ \gamma \in \boldsymbol{I} \cup \boldsymbol{A}, \\ & 0 \leq i \leq j, \ (p,q) \leq (i,j), \ adj \in \{\text{true}, \text{false}\} \end{array} \right\}$$

$$\mathcal{D}^{\text{Init}}_{\text{buE}_1} = \frac{}{[N^\gamma \to \bullet\delta, i, i \mid -, - \mid \text{false}]}$$

$$\mathcal{D}^{\text{Foot}}_{\text{buE}_1} = \frac{}{[\mathbf{F}^\beta \to \bot\bullet, i, j \mid i, j \mid \text{false}]}$$

$$\mathcal{D}^{\text{Scan}}_{\text{buE}_1} = \frac{[N^\gamma \to \delta \bullet a\nu, i, j \mid p, q \mid \text{false}], \ [a, j, j+1]}{[N^\gamma \to \delta a \bullet \nu, i, j+1 \mid p, q \mid \text{false}]}$$

$$\mathcal{D}^{\text{Comp}}_{\text{buE}_1} = \frac{\begin{array}{l}[N^\gamma \to \delta \bullet M^\gamma\nu, i, k \mid p', q' \mid \text{false}], \\ [M^\gamma \to \upsilon\bullet, i, j \mid p, q \mid adj]\end{array}}{[N^\gamma \to \delta M^\gamma \bullet \nu, i, j \mid p' \cup p, q' \cup q \mid \text{false}]} \ \ adj = \text{false sii } \mathbf{nil} \in \text{adj}(M^\gamma)$$

$$\mathcal{D}^{\text{Adj}}_{\text{buE}_1} = \frac{[\top \to \mathbf{R}^\beta\bullet, k, j \mid k', j' \mid \text{false}], \ [M^\gamma \to \upsilon\bullet, k', j' \mid p, q \mid \text{false}]}{[M^\gamma \to \upsilon\bullet, k, j \mid p, q \mid \text{true}]} \ \ \beta \in \boldsymbol{A}, \ \beta \in \text{adj}(\gamma)$$

$$\mathcal{D}_{\text{buE}_1} = \mathcal{D}^{\text{Init}}_{\text{buE}_1} \cup \mathcal{D}^{\text{Foot}}_{\text{buE}_1} \cup \mathcal{D}^{\text{Scan}}_{\text{buE}_1} \cup \mathcal{D}^{\text{Comp}}_{\text{buE}_1} \cup \mathcal{D}^{\text{Adj}}_{\text{buE}_1}$$

$$\mathcal{F}_{\text{buE}_1} = \left\{ \ [\top \to \mathbf{R}^\alpha\bullet, 0, n \mid -, - \mid \mathit{false}] \mid \alpha \in \boldsymbol{I} \ \right\}$$

y donde $p \cup q$ se refiere a la operación de unión de índices, función parcial de enteros a enteros definida como

$$p \cup q = \left\{ \begin{array}{ll} p & \text{si } q = - \\ q & \text{si } p = - \\ - & \text{si } p = q = - \end{array} \right.$$

donde $-$ se utiliza para indicar que el valor de uno de los parámetros está indefinido. §

Los pasos $\mathcal{D}^{\text{Init}}_{\text{buE}_1}$ son los encargados de lanzar el análisis ascendente. Los ítems generados por estos pasos son siempre válidos, puesto que no expanden ninguna porción la cadena de entrada. Tan sólo expresan la expectativa de que una determinada producción pueda ser aplicada para reconocer una porción de la cadena que comienza en una determinada posición.

Los pasos $\mathcal{D}^{\text{Foot}}_{\text{buE}_1}$ son utilizados, al igual que en el caso CYK, para predecir la porción de la cadena de entrada cubierta por el pie de un árbol auxiliar.

El algoritmo procede a reconocer ascendentemente los árboles auxiliares mediante la aplicación de pasos $\mathcal{D}^{\text{Comp}}_{\text{buE}_1}$, propagando también la información correspondiente a la porción de la cadena de entrada cubierta por el pie en el caso de los árboles auxiliares.

El paso deductivo $\mathcal{D}^{\text{Adj}}_{\text{buE}_1}$ se comporta de modo idéntico al paso homónimo del algoritmo CYK, comprobando que las predicciones respecto al pie se corresponden con las de una adjunción que se ha realizado realmente.

Proposición 3.1 $\mathbf{CYK} \overset{\mathbf{ir}}{\Longrightarrow} \mathbf{CYK'} \overset{\mathbf{sr}}{\Longrightarrow} \mathbf{ECYK} \overset{\mathbf{ext}}{\Longrightarrow} \mathbf{buE_1}$.

Demostración:

Como primer paso definiremos el sistema de análisis $\mathbb{P}_{\text{CYK}'}$ para una gramática de adjunción $\mathcal{T}$ y una cadena de entrada $a_1 \dots a_n$.

$$\mathcal{I}_{\text{CYK}'} = \left\{ \begin{array}{ll} [N^\gamma \to \delta \bullet \nu, i, j \mid p, q \mid adj] \mid & N^\gamma \to \delta\nu \in \mathcal{P}(\gamma),\ \gamma \in \boldsymbol{I} \cup \boldsymbol{A} \\ & 0 \le i \le j,\ (p,q) \le (k,j),\ adj \in \{\text{true}, \text{false}\} \end{array} \right\}$$

$$\mathcal{D}^{\text{Scan}}_{\text{CYK}'} = \frac{[a, i, i+1]}{[N^\gamma \to a\bullet, i, i+1 \mid -, - \mid \text{false}]}$$

$$\mathcal{D}^{\epsilon}_{\text{CYK}'} = \frac{}{[N^\gamma \to \bullet, i, i \mid -, - \mid \text{false}]} \quad N^\gamma \to \epsilon$$

$$\mathcal{D}^{\text{Foot}}_{\text{CYK}'} = \frac{}{[\mathbf{F}^\gamma \to \bot\bullet, i, j \mid i, j \mid \text{false}]}$$

$$\mathcal{D}^{\text{LeftDom}}_{\text{CYK}'} = \frac{\begin{array}{l}[M^\gamma \to \delta\bullet, i, k \mid p, q \mid adj1], \\ [P^\gamma \to \nu\bullet, k, j \mid -, - \mid adj2]\end{array}}{[N^\gamma \to M^\gamma P^\gamma\bullet, i, j \mid p, q \mid \text{false}]} \begin{array}{l} N^\gamma \to M^\gamma P^\gamma \in \mathcal{P}(\gamma), \\ M^\gamma \in \text{espina}(\gamma), \\ adj1 = \text{false sii } \mathbf{nil} \in \text{adj}(M^\gamma), \\ adj2 = \text{false sii } \mathbf{nil} \in \text{adj}(P^\gamma) \end{array}$$

$$\mathcal{D}^{\text{RightDom}}_{\text{CYK}'} = \frac{\begin{array}{l}[M^\gamma \to \delta\bullet, i, k \mid -, - \mid adj1], \\ [P^\gamma \to \nu\bullet, k, j \mid p, q \mid adj2]\end{array}}{[N^\gamma \to M^\gamma P^\gamma\bullet, i, j \mid p, q \mid \text{false}]} \begin{array}{l} N^\gamma \to M^\gamma P^\gamma \in \mathcal{P}(\gamma), \\ P^\gamma \in \text{espina}(\gamma), \\ adj1 = \text{false sii } \mathbf{nil} \in \text{adj}(M^\gamma), \\ adj2 = \text{false sii } \mathbf{nil} \in \text{adj}(P^\gamma) \end{array}$$

$$\mathcal{D}^{\text{NoDom}}_{\text{CYK}'} = \frac{\begin{array}{l}[M^\gamma \to \delta\bullet, i, k \mid -, - \mid adj1], \\ [P^\gamma \to \nu\bullet, k, j \mid -, - \mid adj2]\end{array}}{[N^\gamma \to M^\gamma P^\gamma\bullet, i, j \mid -, - \mid \text{false}]} \begin{array}{l} N^\gamma \to M^\gamma P^\gamma \in \mathcal{P}(\gamma), \\ N^\gamma \notin \text{espina}(\gamma), \\ adj1 = \text{false sii } \mathbf{nil} \in \text{adj}(M^\gamma), \\ adj2 = \text{false sii } \mathbf{nil} \in \text{adj}(P^\gamma) \end{array}$$

$$\mathcal{D}^{\text{Unary}}_{\text{CYK}'} = \frac{[M^\gamma \to \delta\bullet, i, j \mid p, q \mid adj]}{[N^\gamma \to M^\gamma\bullet, i, j \mid p, q] \mid \text{false}]} \begin{array}{l} N^\gamma \to M^\gamma \in \mathcal{P}(\gamma), \\ adj = \text{false sii } \mathbf{nil} \in \text{adj}(M^\gamma) \end{array}$$

$$\mathcal{D}^{\text{Adj}}_{\text{CYK}'} = \frac{[\top \rightarrow \mathbf{R}^{\beta}\bullet, i', j' \mid i, j \mid adj],\ [N^{\gamma} \rightarrow \delta\bullet, i, j \mid p, q \mid \text{false}]}{[N^{\gamma} \rightarrow \delta\bullet, i', j' \mid p, q \mid \text{true}]} \quad \beta \in \boldsymbol{A},\ \beta \in \text{adj}(N^{\gamma})$$

$$\mathcal{D}_{\text{CYK}'} = \mathcal{D}^{\text{Scan}}_{\text{CYK}'} \cup \mathcal{D}^{\epsilon}_{\text{CYK}'} \cup \mathcal{D}^{\text{Foot}}_{\text{CYK}'} \cup \mathcal{D}^{\text{LeftDom}}_{\text{CYK}'} \cup \mathcal{D}^{\text{RightDom}}_{\text{CYK}'} \cup \mathcal{D}^{\text{NoDom}}_{\text{CYK}'} \cup \mathcal{D}^{\text{Unary}}_{\text{CYK}'} \cup \mathcal{D}^{\text{Adj}}_{\text{CYK}'}$$

$$\mathcal{F}_{\text{CYK}'} = \left\{\ [\top \rightarrow \mathbf{R}^{\alpha}\bullet, 0, n \mid -, - \mid adj] \mid \alpha \in \boldsymbol{I},\ adj = \text{false sii } \mathbf{nil} \in \text{adj}(\mathbf{R}^{\alpha})\ \right\}$$

Para demostrar que **CYK** $\overset{\textbf{ir}}{\Longrightarrow}$ **CYK′**, definiremos la siguiente función

$$f([N^{\gamma} \rightarrow \delta\bullet, i, j \mid p, q \mid adj]) = [N^{\gamma}, i, j \mid p, q \mid adj]$$

de la cual se obtiene directamente que $\mathcal{I}_{\text{CYK}} = f(\mathcal{I}_{\text{CYK}'})$ y que $\Delta_{\text{CYK}} = f(\Delta_{\text{CYK}'})$ por inducción en la longitud de las secuencias de derivación. En consecuencia, $\mathbb{P}_{\text{CYK}} \xrightarrow{\textbf{ir}} \mathbb{P}_{\text{CYK}'}$, con lo que hemos probado lo que pretendíamos.

Definiremos ahora el sistema de análisis sintáctico $\mathbb{P}_{\text{ECYK}}$ para una gramática de adjunción de árboles $\mathcal{T}$ en la que ningún nodo puede tener más de dos descendientes y una cadena de entrada $a_1 \ldots a_n$ dada:

$$\mathcal{I}_{\text{ECYK}} = \mathcal{I}_{\text{CYK}'} = \mathcal{I}_{\text{buE}_1}$$

$$\mathcal{D}^{\text{Init}}_{\text{ECYK}} = \mathcal{D}^{\text{Init}}_{\text{buE}_1}$$

$$\mathcal{D}^{\text{Foot}}_{\text{ECYK}} = \mathcal{D}^{\text{Foot}}_{\text{buE}_1}$$

$$\mathcal{D}^{\text{Scan}}_{\text{ECYK}} = \mathcal{D}^{\text{Scan}}_{\text{buE}_1}$$

$$\mathcal{D}^{\text{Comp}}_{\text{ECYK}} = \mathcal{D}^{\text{Comp}}_{\text{buE}_1}$$

$$\mathcal{D}^{\text{Adj}}_{\text{ECYK}} = \mathcal{D}^{\text{Adj}}_{\text{buE}_1}$$

$$\mathcal{D}_{\text{ECYK}} = \mathcal{D}^{\text{Init}}_{\text{ECYK}} \cup \mathcal{D}^{\text{Foot}}_{\text{ECYK}} \cup \mathcal{D}^{\text{Scan}}_{\text{ECYK}} \cup \mathcal{D}^{\text{Comp}}_{\text{ECYK}} \cup \mathcal{D}^{\text{Adj}}_{\text{ECYK}}$$

$$\mathcal{F}_{\text{ECYK}} = \mathcal{F}_{\text{buE}_1}$$

Para demostrar que **CYK′** $\overset{\textbf{sr}}{\Longrightarrow}$ **ECYK**, deberemos demostrar que para todo sistema de análisis $\mathbb{P}_{\text{CYK}'}$ y $\mathbb{P}_{\text{ECYK}}$ se cumple que $\mathcal{I}_{\text{CYK}'} \subseteq \mathcal{I}_{\text{ECYK}}$ y que $\overset{*}{\vdash}_{\text{CYK}'} \subseteq \overset{*}{\vdash}_{\text{ECYK}}$. Lo primero es cierto por definición, puesto que $\mathcal{I}_{\text{CYK}'} = \mathcal{I}_{\text{ECYK}}$. Para lo segundo debemos mostrar que $\mathcal{D}_{\text{CYK}'} \subseteq \overset{*}{\vdash}_{\text{ECYK}}$. Consideremos caso por caso:

- un paso deductivo $\mathcal{D}^{\text{Scan}}_{\text{CYK}'}$ es equivalente a la secuencia de pasos deductivos constituida por la aplicación de un paso $\mathcal{D}^{\text{Init}}_{\text{ECYK}}$ y un paso $\mathcal{D}^{\text{Scan}}_{\text{ECYK}}$:

$$\frac{}{[N^{\gamma} \rightarrow \bullet a, i, i, \mid -, - \mid \text{false}]}$$

$$\frac{[N^{\gamma} \rightarrow \bullet a, i, i, \mid -, - \mid \text{false}], [a, i, i+1]}{[N^{\gamma} \rightarrow a\bullet, i, i+1, \mid -, - \mid \text{false}]}$$

- un paso deductivo $\mathcal{D}^{\epsilon}_{\text{CYK}'}$ es equivalente a un paso $\mathcal{D}^{\text{Init}}_{\text{ECYK}}$:

$$\frac{}{[N^{\gamma} \rightarrow \bullet, i, i, \mid -, - \mid \text{false}]}$$

- $\mathcal{D}^{\text{Foot}}_{\text{ECYK}} = \mathcal{D}^{\text{Foot}}_{\text{CYK}'}$.

- un paso deductivo $\mathcal{D}_{\text{CYK}'}^{\text{LeftDom}}$ es equivalente a la secuencia de pasos deductivos constituida por un paso $\mathcal{D}_{\text{ECYK}}^{\text{Init}}$ y dos pasos $\mathcal{D}_{\text{ECYK}}^{\text{Comp}}$:

$$\frac{}{[N^\gamma \rightarrow \bullet M^\gamma P^\gamma, i, i, \mid -, - \mid \text{false}]}$$

$$\frac{[N^\gamma \rightarrow \bullet M^\gamma P^\gamma, i, i, \mid -, - \mid \text{false}],\ [M^\gamma \rightarrow \delta \bullet, i, k \mid p, q \mid adj]}{[N^\gamma \rightarrow M^\gamma \bullet P^\gamma, i, k, \mid p, q \mid \text{false}]}$$

$$\frac{[N^\gamma \rightarrow M^\gamma \bullet P^\gamma, i, k, \mid p, q \mid \text{false}], [P^\gamma \rightarrow \nu \bullet, k, j \mid -, - \mid adj]}{[N^\gamma \rightarrow M^\gamma P^\gamma \bullet, i, j, \mid p, q \mid \text{false}]}$$

- un paso $\mathcal{D}_{\text{CYK}'}^{\text{RightDom}}$ es equivalente a la secuencia formada por un paso $\mathcal{D}_{\text{ECYK}}^{\text{Init}}$ y dos pasos $\mathcal{D}_{\text{ECYK}}^{\text{Comp}}$:

$$\frac{}{[N^\gamma \rightarrow \bullet M^\gamma P^\gamma, i, i, \mid -, - \mid \text{false}]}$$

$$\frac{[N^\gamma \rightarrow \bullet M^\gamma P^\gamma, i, i, \mid -, - \mid \text{false}],\ [M^\gamma \rightarrow \delta \bullet, i, k \mid -, - \mid adj]}{[N^\gamma \rightarrow M^\gamma \bullet P^\gamma, i, k, \mid -, - \mid \text{false}]}$$

$$\frac{[N^\gamma \rightarrow M^\gamma \bullet P^\gamma, i, k, \mid -, - \mid \text{false}],\ [P^\gamma \rightarrow \nu \bullet, k, j \mid p, q \mid adj]}{[N^\gamma \rightarrow M^\gamma P^\gamma \bullet, i, j, \mid p, q \mid \text{false}]}$$

- un paso $\mathcal{D}_{\text{CYK}'}^{\text{NoDom}}$ es equivalente a la secuencia formada por un paso $\mathcal{D}_{\text{ECYK}}^{\text{Init}}$ y dos pasos $\mathcal{D}_{\text{ECYK}}^{\text{Comp}}$:

$$\frac{}{[N^\gamma \rightarrow \bullet M^\gamma P^\gamma, i, i, \mid -, - \mid \text{false}]}$$

$$\frac{[N^\gamma \rightarrow \bullet M^\gamma P^\gamma, i, i, \mid -, - \mid \text{false}],\ [M^\gamma \rightarrow \delta \bullet, i, k \mid -, - \mid adj]}{[N^\gamma \rightarrow M^\gamma \bullet P^\gamma, i, k, \mid -, - \mid \text{false}]}$$

$$\frac{[N^\gamma \rightarrow M^\gamma \bullet P^\gamma, i, k, \mid -, - \mid \text{false}],\ [P^\gamma \rightarrow \nu \bullet, k, j \mid -, - \mid adj]}{[N^\gamma \rightarrow M^\gamma P^\gamma \bullet, i, j, \mid -, - \mid \text{false}]}$$

- un paso $\mathcal{D}_{\text{CYK}'}^{\text{Unary}}$ es equivalente a la secuencia formada por un paso $\mathcal{D}_{\text{ECYK}}^{\text{Init}}$ y un paso $\mathcal{D}_{\text{ECYK}}^{\text{Comp}}$:

$$\frac{}{[N^\gamma \rightarrow \bullet M^\gamma, i, i, \mid -, - \mid \text{false}]}$$

$$\frac{[N^\gamma \rightarrow \bullet M^\gamma, i, i, \mid -, - \mid \text{false}],\ [M^\gamma \rightarrow \delta \bullet, i, j \mid -, - \mid adj]}{[N^\gamma \rightarrow M^\gamma \bullet, i, j, \mid -, - \mid \text{false}]}$$

- $\mathcal{D}_{\text{ECYK}}^{\text{Adj}} = \mathcal{D}_{\text{CYK}'}^{\text{adj}}$.

El esquema de análisis sintáctico **ECYK** está definido para gramáticas de adjunción de árboles en las cuales ningún nodo puede tener más de dos descendientes mientras que el esquema de análisis $\mathbf{buE_1}$ está definido para cualquier TAG. Es fácil mostrar que $\mathbf{ECYK} \overset{\mathbf{ext}}{\Longrightarrow} \mathbf{buE_1}$ puesto que $\mathbf{ECYK}(\mathcal{T}) = \mathbf{buE_1}(\mathcal{T})$ es cierto para toda gramática de adjunción de árboles, ya que por definición $\mathbb{P}_{\text{ECYK}} = \mathbb{P}_{\text{buE}_1}$. □

Se puede eliminar el elemento de los ítems que indica si se ha realizado una adjunción en el nodo situado en el lado izquierdo de la producción contenida en dicho ítem, quedando definido el conjunto de ítems

$$\left\{ [N^\gamma \rightarrow \delta \bullet \nu, i, j \mid p, q] \mid \begin{array}{ll} \delta \overset{*}{\Rightarrow} a_{i+1} \ldots a_p \, \mathbf{F}^\gamma \, a_{q+1} \ldots a_j \overset{*}{\Rightarrow} a_{i+1} \ldots a_j & \text{sii } (p,q) \neq (-,-) \\ \delta \overset{*}{\Rightarrow} a_{i+1} \ldots a_j & \text{sii } (p,q) = (-,-) \end{array} \right\}$$

Para ello es necesario aplicar un filtro al esquema de análisis sintáctico $\mathbf{buE_1}$, consistente en la contracción de pasos deductivos: puesto que el ítem generado por un paso deductivo de tipo

Adj sólo podrá ser utilizado en un paso de tipo Comp para avanzar el punto en la producción que predijo el no terminal del lado izquierdo de su producción, podemos crear un nuevo tipo AdjComp de paso deductivo y eliminar los pasos de tipo Adj. Obtendremos así el esquema de análisis **buE** cuyo sistema de análisis $\mathbb{P}_{\text{buE}}$ mostramos a continuación.

Esquema de análisis sintáctico 3.3 El sistema de análisis $\mathbb{P}_{\text{buE}}$ que se corresponde con una nueva versión de la extensión del algoritmo de Earley ascendente para TAG, dada una gramática de adjunción de árboles $\mathcal{T}$ y una cadena de entrada $a_1 \dots a_n$ se define como sigue:

$$\mathcal{I}_{\text{buE}} = \left\{ \begin{array}{ll} [N^\gamma \rightarrow \delta \bullet \nu, i, j \mid p, q] \mid & N^\gamma \rightarrow \delta\nu \in \mathcal{P}(\gamma),\ \gamma \in \boldsymbol{I} \cup \boldsymbol{A}, \\ & 0 \leq i \leq j,\ (p,q) \leq (i,j) \end{array} \right\}$$

$$\mathcal{D}_{\text{buE}}^{\text{Init}} = \frac{}{[N^\gamma \rightarrow \bullet\delta, i, i \mid -, -]}$$

$$\mathcal{D}_{\text{buE}}^{\text{Foot}} = \frac{}{[\mathbf{F}^\beta \rightarrow \bot\bullet, i, j \mid i, j]}$$

$$\mathcal{D}_{\text{buE}}^{\text{Scan}} = \frac{[N^\gamma \rightarrow \delta \bullet a\nu, i, j \mid p, q],\ [a, j, j+1]}{[N^\gamma \rightarrow \delta a \bullet \nu, i, j+1 \mid p, q]}$$

$$\mathcal{D}_{\text{buE}}^{\text{Comp}} = \frac{[N^\gamma \rightarrow \delta \bullet M^\gamma\nu, i, k \mid p, q],\ [M^\gamma \rightarrow \upsilon\bullet, k, j \mid p', q']}{[N^\gamma \rightarrow \delta M^\gamma \bullet \nu, i, j \mid p \cup p', q \cup q']}\ \ \mathbf{nil} \in \text{adj}(M^\gamma)$$

$$\mathcal{D}_{\text{buE}}^{\text{AdjComp}} = \frac{\begin{array}{l}[\top \rightarrow \mathbf{R}^\beta\bullet, k, j \mid l, m], \\ [M^\gamma \rightarrow \upsilon\bullet, l, m \mid p', q'], \\ [N^\gamma \rightarrow \delta \bullet M^\gamma\nu, i, k \mid p, q],\end{array}}{[N^\gamma \rightarrow \delta M^\gamma \bullet \nu, i, j \mid p \cup p', q \cup q']}\ \ \beta \in \boldsymbol{A},\ \beta \in \text{adj}(M^\gamma)$$

$$\mathcal{D}_{\text{buE}} = \mathcal{D}_{\text{buE}}^{\text{Init}} \cup \mathcal{D}_{\text{buE}}^{\text{Foot}} \cup \mathcal{D}_{\text{buE}}^{\text{Scan}} \cup \mathcal{D}_{\text{buE}}^{\text{Comp}} \cup \mathcal{D}_{\text{buE}}^{\text{AdjComp}}$$

$$\mathcal{F}_{\text{buE}} = \left\{\ [\top \rightarrow \mathbf{R}^\alpha\bullet, 0, n \mid -, -] \mid \alpha \in \boldsymbol{I}\ \right\}$$

§

Como se puede observar, el paso $\mathcal{D}_{\text{buE}}^{\text{AdjComp}}$ solamente permite que un árbol auxiliar sea adjuntado en un nodo de un árbol elemental, pues una vez realizada la adjunción, el punto de la producción correspondiente al padre del nodo de adjunción se mueve a la derecha mientras que la producción del nodo de adjunción permanece sin cambios: si posteriormente otro árbol auxiliar es adjuntado en dicho nodo, representará una ambigüedad en el análisis sintáctico de la cadena de entrada, no la adjunción simultánea de dos árboles auxiliares en un mismo nodo [175]. En la figura 3.3 se muestra una representación gráfica de la aplicación de este paso deductivo para su caso más complejo, aquel en el que γ es un árbol auxiliar y el nodo de adjunción pertenece a su espina.

La complejidad temporal del esquema de análisis **buE** con respecto a la cadena de entrada es aparentemente $\mathcal{O}(n^7)$ puesto que son 7 los índices involucrados en el paso deductivo $\mathcal{D}_{\text{buE}}^{\text{AdjComp}}$: i, j, k, l, m y bien p y q o bien p' y q'. Sin embargo, podemos observar que los índices l y m sólo son necesarios para relacionar los dos primeros antecedentes y son irrelevantes para el resto de los ítems involucrados en el paso deductivo. Por tanto, mediante la aplicación parcial o *currificación* del paso deductivo $\mathcal{D}_{\text{buE}}^{\text{AdjComp}}$ la complejidad del mismo se reduce a $\mathcal{O}(n^6)$. De forma equivalente, también se podría incluir dicha aplicación parcial en el propio esquema de análisis, sustituyendo el paso $\mathcal{D}_{\text{buE}}^{\text{AdjComp}}$ por otros dos, uno que combine los dos primeros ítems antecedentes y genere un ítem intermedio y otro que combine dicho ítem con el tercer antecedente del paso original para obtener el ítem resultado. Sin embargo, con ello oscureceríamos la comprensión del algoritmo descrito en el esquema y estaríamos vulnerando la filosofía de los esquemas de análisis, ya que la aplicación parcial es un detalle de implementación del que nos debemos abstraer.

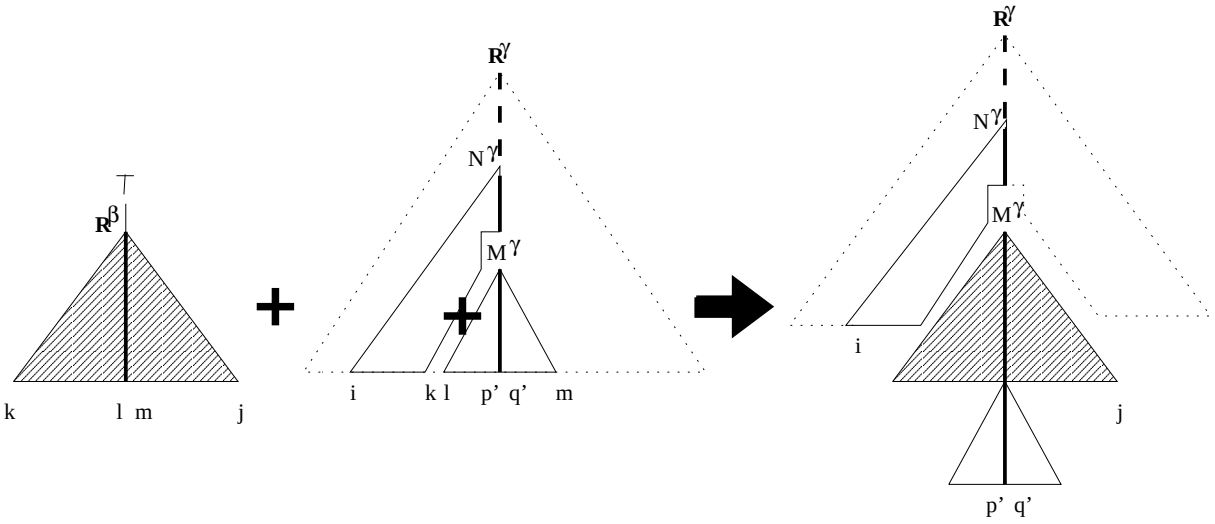

Figura 3.3: Descripción gráfica de un paso $\mathcal{D}_{\text{buE}}^{\text{AdjComp}}$

Proposición 3.2 $\mathbf{buE_1} \overset{\mathbf{df}}{\Longrightarrow} \mathbf{buE'_1} \overset{\mathbf{sc}}{\Longrightarrow} \mathbf{buE_2} \overset{\mathbf{ic}}{\Longrightarrow} \mathbf{buE}$.

Demostración:

Como primer paso definiremos el sistema de análisis $\mathbb{P}_{\text{buE}'_1}$ para una gramática de adjunción $\mathcal{T}$ y una cadena de entrada $a_1 \dots a_n$, en el cual el paso Comp ha sido desdoblado en Comp^1 y Comp^2, el primero de los cuales se encarga de avanzar el punto en aquellos casos en que el nodo sobre el que se avanza no ha sido utilizado como nodo de adjunción, mientras que el segundo avanza el punto sobre un nodo de adjunción.

$$\mathcal{I}_{\text{buE}'_1} = \mathcal{I}_{\text{buE}_1}$$

$$\mathcal{D}^{\text{Init}}_{\text{buE}'_1} = \mathcal{D}^{\text{Init}}_{\text{buE}_1}$$

$$\mathcal{D}^{\text{Foot}}_{\text{buE}'_1} = \mathcal{D}^{\text{Foot}}_{\text{buE}_1}$$

$$\mathcal{D}^{\text{Scan}}_{\text{buE}'_1} = \mathcal{D}^{\text{Scan}}_{\text{buE}_1}$$

$$\mathcal{D}^{\text{Comp}^1}_{\text{buE}'_1} = \frac{[N^\gamma \to \delta \bullet M^\gamma \nu, i, k \mid p', q' \mid \text{false}],\ [M^\gamma \to \upsilon \bullet, k, j \mid p, q \mid \text{false}]}{[N^\gamma \to \delta M^\gamma \bullet \nu, i, j \mid p' \cup p, q' \cup q \mid \text{false}]}\ \ \mathbf{nil} \in \text{adj}(M^\gamma)$$

$$\mathcal{D}^{\text{Comp}^2}_{\text{buE}'_1} = \frac{[N^\gamma \to \delta \bullet M^\gamma \nu, i, k \mid p', q' \mid \text{false}],\ [M^\gamma \to \upsilon \bullet, k, j \mid p, q \mid \text{true}]}{[N^\gamma \to \delta M^\gamma \bullet \nu, i, j \mid p' \cup p, q' \cup q \mid \text{false}]}\ \ \exists \beta \in \boldsymbol{A},\ \ \beta \in \text{adj}(M^\gamma)$$

$$\mathcal{D}^{\text{Adj}}_{\text{buE}_1} = \frac{[\top \rightarrow \mathbf{R}^{\beta}\bullet, k, j \mid k', j' \mid \text{false}],\ [M^{\gamma} \rightarrow \upsilon\bullet, k', j' \mid p, q \mid \text{false}],}{[M^{\gamma} \rightarrow \upsilon\bullet, k, j \mid p, q \mid \text{true}]}\ \ \beta \in \boldsymbol{A},\ \ \beta \in \text{adj}(\gamma)$$

$$\mathcal{D}_{\text{buE}'_1} = \mathcal{D}^{\text{Init}}_{\text{buE}'_1} \cup \mathcal{D}^{\text{Foot}}_{\text{buE}'_1} \cup \mathcal{D}^{\text{Scan}}_{\text{buE}'_1} \cup \mathcal{D}^{\text{Comp}^1}_{\text{buE}'_1} \cup \mathcal{D}^{\text{Comp}^2}_{\text{buE}'_1} \cup \mathcal{D}^{\text{Adj}}_{\text{buE}'_1}$$

$$\mathcal{F}_{\text{buE}'_1} = \mathcal{F}_{\text{buE}_1}$$

Los pasos deductivos $\mathcal{D}^{\text{Comp}^1}_{\text{buE}'_1}$ y $\mathcal{D}^{\text{Comp}^2}_{\text{buE}'_1}$ son el resultado de aplicar un filtro dinámico al paso $\mathcal{D}^{\text{Comp}}_{\text{buE}_1}$. El el caso de $\mathcal{D}^{\text{Comp}^1}_{\text{buE}'_1}$ dicho filtro consiste en comprobar si el último elemento de los ítems contiene el valor false, mientras que en el caso de $\mathcal{D}^{\text{Comp}^2}_{\text{buE}'_1}$ consiste en comprobar que su valor es *true*.

Para demostrar que $\mathbf{buE_1} \overset{\mathbf{df}}{\Longrightarrow} \mathbf{buE'_1}$, deberemos demostrar que para todo esquema de análisis $\mathbb{P}_{\text{buE}_1}$ y $\mathbb{P}_{\text{buE}'_1}$ se cumple que $\mathcal{I}_{\text{buE}_1} \supseteq \mathcal{I}_{\text{buE}'_1}$ y $\vdash_{\text{buE}_1} \supseteq \vdash_{\text{buE}'_1}$. Lo primero se cumple puesto que por definición $\mathcal{I}_{\text{buE}_1} = \mathcal{I}_{\text{buE}'_1}$. Para lo segundo debemos mostrar que $\vdash_{\text{buE}_1} \supseteq \mathcal{D}_{\text{buE}'_1}$. Para ello sólo necesitamos considerar aquellos pasos de $\mathbb{P}_{\text{buE}'_1}$ a los que se les ha aplicado el filtro dinámico, pues los restantes pasos permanecen inalterados:

- un paso $\mathcal{D}^{\text{Comp}^1}_{\text{buE}'_1}$ es equivalente a la aplicación del paso $\mathcal{D}^{\text{Comp}}_{\text{buE}_1}$:

$$\frac{[N^{\gamma} \rightarrow \delta \bullet M^{\gamma}\nu, i, k \mid p', q' \mid \text{false}],\ [M^{\gamma} \rightarrow \upsilon\bullet, k, j \mid p, q \mid \text{false}]}{[N^{\gamma} \rightarrow \delta M^{\gamma} \bullet \nu, i, j \mid p' \cup p, q' \cup q \mid \text{false}]}$$

- un paso $\mathcal{D}^{\text{Comp}^2}_{\text{buE}'_1}$ es equivalente a la aplicación del paso $\mathcal{D}^{\text{Comp}}_{\text{buE}_1}$:

$$\frac{[N^{\gamma} \rightarrow \delta \bullet M^{\gamma}\nu, i, k \mid p', q' \mid \text{false}],\ [M^{\gamma} \rightarrow \upsilon\bullet, k, j \mid p, q \mid \text{true}]}{[N^{\gamma} \rightarrow \delta M^{\gamma} \bullet \nu, i, j \mid p' \cup p, q' \cup q \mid \text{false}]}$$

Sólo se pueden generar ítems que tengan el último componente con valor true como consecuencia de la aplicación de un paso $\mathcal{D}^{\text{Adj}}_{\text{buE}'_1}$ y a su vez estos ítems sólo puede ser utilizados como antecedentes en un paso $\mathcal{D}^{\text{Comp}^2}_{\text{buE}'_1}$. Por tanto podemos juntar ambos pasos pasos en uno sólo. A continuación definimos el sistema de análisis $\mathbb{P}_{\text{buE}_2}$ para una gramática de adjunción $\mathcal{T}$ y una cadena de entrada $a_1 \ldots a_n$ que incorpora esta transformación.

$$\mathcal{I}_{\text{buE}_2} = \mathcal{I}_{\text{buE}'_1}$$

$$\mathcal{D}^{\text{Init}}_{\text{buE}_2} = \mathcal{D}^{\text{Init}}_{\text{buE}'_1}$$

$$\mathcal{D}^{\text{Foot}}_{\text{buE}_2} = \mathcal{D}^{\text{Foot}}_{\text{buE}'_1}$$

$$\mathcal{D}^{\text{Scan}}_{\text{buE}_2} = \mathcal{D}^{\text{Scan}}_{\text{buE}'_1}$$

$$\mathcal{D}^{\text{Comp}}_{\text{buE}_2} = \mathcal{D}^{\text{Comp}^1}_{\text{buE}'_1}$$

$$\mathcal{D}^{\text{Adj}}_{\text{buE}_2} = \frac{\begin{array}{c}[\top \rightarrow \mathbf{R}^{\beta}\bullet, k, j \mid k', j' \mid \text{false}],\\ [M^{\gamma} \rightarrow \upsilon\bullet, k', j' \mid p, q \mid \text{false}],\\ [N^{\gamma} \rightarrow \delta \bullet M^{\gamma}\nu, i, k \mid p', q' \mid \text{false}]\end{array}}{[N^{\gamma} \rightarrow \delta M^{\gamma} \bullet \nu, i, j \mid p' \cup p, q' \cup q \mid \text{false}]}\ \ \beta \in \boldsymbol{A},\ \ \beta \in \text{adj}(\gamma)$$

$$\mathcal{D}_{\text{buE}_2} = \mathcal{D}^{\text{Init}}_{\text{buE}_2} \cup \mathcal{D}^{\text{Foot}}_{\text{buE}_2} \cup \mathcal{D}^{\text{Scan}}_{\text{buE}_2} \cup \mathcal{D}^{\text{Comp}}_{\text{buE}_2} \cup \mathcal{D}^{\text{Adj}}_{\text{buE}_2}$$

$$\mathcal{F}_{\text{buE}_2} = \mathcal{F}_{\text{buE}_1}$$

Para demostrar que el sistema de análisis sintáctico $\mathbb{P}_{\text{buE}_2}$ es el resultado de aplicar una contracción de pasos al sistema de análisis $\mathbb{P}_{\text{buE}'_1}$ tenemos que demostrar que $\mathcal{I}_{\text{buE}'_1} \supseteq \mathcal{I}_{\text{buE}_2}$ y que $\overset{*}{\vdash}_{\text{buE}'_1} \supseteq \overset{*}{\vdash}_{\text{buE}_2}$. Lo primero es cierto por definición, puesto que $\mathcal{I}_{\text{buE}'_1} = \mathcal{I}_{\text{buE}_2}$. Para lo

segundo es suficiente con demostrar que $\overset{*}{\vdash}_{\text{buE}'_1} \supseteq \mathcal{D}_{\text{buE}_2}$. Puesto que los demás pasos deductivos son idénticos en ambos esquemas, debemos mostrar únicamente que $\overset{*}{\vdash}_{\text{buE}'_1} \supseteq \mathcal{D}^{\text{Adj}}_{\text{buE}_2}$, tarea que no presenta complicación alguna puesto que como se ha mencionado anteriormente, un paso $\mathcal{D}^{\text{Adj}}_{\text{buE}_2}$ es equivalente a la aplicación consecutiva de un paso $\mathcal{D}^{\text{Adj}}_{\text{buE}'_1}$ y un paso $\mathcal{D}^{\text{Comp}^2}_{\text{buE}'_1}$:

$$\frac{[\top \rightarrow \mathbf{R}^{\beta}\bullet, k, j \mid k', j' \mid \text{false}],\ [M^{\gamma} \rightarrow \upsilon\bullet, k', j' \mid p, q \mid \text{false}]}{[M^{\gamma} \rightarrow \upsilon\bullet, k, j \mid p, q \mid \text{true}]}$$

$$\frac{[N^{\gamma} \rightarrow \delta \bullet M^{\gamma}\nu, i, k \mid p', q' \mid \text{false}],\ [M^{\gamma} \rightarrow \upsilon\bullet, k, j \mid p, q \mid \text{true}]}{[N^{\gamma} \rightarrow \delta M^{\gamma} \bullet \nu, i, j \mid p' \cup p, q' \cup q \mid \text{false}]}$$

Finalmente observamos que todos los ítems del sistema de análisis $\mathbb{P}_{\text{buE}_2}$ tienen la forma $[N^{\gamma} \rightarrow \delta \bullet \nu, i, j \mid p, q \mid \text{false}]$. Puesto que el último elemento tiene un valor constante, su eliminación no producirá ningún efecto con respecto al conjunto de ítems que puedan ser generados. El sistema de análisis $\mathbb{P}_{\text{buE}}$ se obtiene precisamente al aplicar esta transformación al sistema de análisis $\mathbb{P}_{\text{buE}_2}$. Dicha transformación constituye un caso peculiar de contracción de ítems puesto que no se rompe un ítem en varios, sino que se establece una relación biyectiva entre los conjuntos $\mathcal{I}_{\text{buE}_2}$ y $\mathcal{I}_{\text{buE}}$. Vemos que la relación de contracción de ítems se mantiene entre los dos esquemas de análisis puesto que definiendo la función

$$f([N^{\gamma} \rightarrow \delta \bullet \nu, i, j \mid p, q \mid \text{false}]) = [N^{\gamma} \rightarrow \delta \bullet \nu, i, j \mid p, q]$$

obtenemos que $f(\mathcal{I}_{\text{buE}_2}) = \mathcal{I}_{\text{buE}}$ y $f(\Delta_{\text{buE}_2}) = \Delta_{\text{buE}}$ por inducción en la longitud de las secuencias de derivación. □

3.4. La propiedad del prefijo válido en los algoritmos de análisis

Los analizadores sintácticos que satisfacen la *propiedad del prefijo válido* (VPP) garantizan que, en tanto que leen la cadena de entrada de izquierda a derecha, las subcadenas leídas son prefijos válidos del lenguaje definido por la gramática. A la propiedad del prefijo válido también se la denomina a veces propiedad de detección de errores porque implica que los errores son detectados tan pronto como es posible en una lectura de la cadena de entrada de izquierda a derecha. La carencia de la propiedad del prefijo válido no significa que los errores no sean detectados, sino simplemente que estos lo serán más tarde. En contra de la idea mantenida en ciertos momentos [168], la propiedad del prefijo válido es independiente de la propiedad de análisis *en línea* [169].

Más formalmente, un analizador sintáctico satisface la propiedad del prefijo válido si al leer la subcadena $a_1 \ldots a_k$ de la cadena de entrada $a_1 \ldots a_k a_{k+1} \ldots a_n$ garantiza que hay una cadena $b_1 \ldots b_m$, donde b_i no tiene porque formar parte de la cadena de entrada, tal que $a_1 \ldots a_k b_1 \ldots b_m$ es una cadena válida del lenguaje.

El mantenimiento de la propiedad del prefijo válido exige ir reconociendo los posibles árboles derivados de forma prefija. Este recorrido consta de dos fases, una descendente que dado un nodo expande sus nodos hijos y una ascendente que agrupa los nodos hijos para indicar el reconocimiento del nodo padre. Cuando se desea mantener la propiedad del prefijo válido estas dos fases deben actuar coordinadamente. La fase descendente debe ser además restringida, para evitar expansiones que lleven al reconocimiento de prefijos no válidos [169].

En el caso de gramáticas independientes del contexto, existen numerosos algoritmos de análisis que preservan la propiedad del prefijo válido (por ejemplo, Earley) y que muestran una complejidad en el peor caso igual a aquellos algoritmos que no la preservan (por ejemplo, CYK).

Esto se debe a que la operación de sustitución puede ser aplicada de forma totalmente independiente del contexto, lo que conlleva que el conjunto de caminos de una gramática independiente del contexto sea un lenguaje regular. Como consecuencia, el mantenimiento de la propiedad del prefijo válido se puede asegurar sin tener que aplicar restricciones complejas sobre la fase descendente de los algoritmos.

En el caso de las gramáticas de adjunción, la operación de adjunción no es completamente independiente del contexto en el cual se aplica. Durante el reconocimiento de un árbol derivado en forma prefija, la expansión de un nodo puede depender de operaciones de adjunción previamente realizadas en la parte recorrida del árbol. Esta dependencia del contexto conlleva que el conjunto de caminos ya no sea un lenguaje regular sino un lenguaje independiente del contexto [230, 217]. Un algoritmo básicamente ascendente (p.ej.: de tipo CYK, algunas variantes de tipo Earley) puede simplemente hacer uso de una pila para ir almacenando las dependencias indicadas por el lenguaje que define el conjunto de caminos. Con ello se conseguiría un algoritmo de análisis correcto pero sin la propiedad del prefijo válido. Para preservar esta propiedad es necesario disponer de una fase descendente, que también tendría que disponer de una pila para satisfacer las restricciones impuestas por el lenguaje que define el conjunto de caminos. Schabes [169] argumentaba que entonces, al tener que coordinar las pilas de la fase ascendente y descendente, la complejidad del algoritmo de análisis sintáctico resultante aumentaría. Sin embargo, el algoritmo descrito por Nederhof en [125] mantiene la propiedad del prefijo válido con una complejidad $\mathcal{O}(n^6)$, igual a la de aquellos algoritmos que no la mantienen.

3.5. Algoritmos de tipo Earley sin la propiedad del prefijo válido

Mediante la aplicación de un filtrado dinámico a los esquemas de análisis sintácticos anteriores es posible obtener un esquema de análisis sintáctico de un algoritmo al estilo del de Earley pero extendido al caso de gramáticas de adjunción de árboles. Concretamente, el filtrado que se realiza es el siguiente:

- El paso deductivo $\mathcal{D}_{\text{buE}}^{\text{Init}}$ sólo contendrá producciones cuyo lado izquierdo corresponda con la raíz de un árbol inicial.
- Un conjunto de pasos deductivos predictivos controlan la generación de nuevos ítems tratando de limitarla únicamente a aquellos que puedan resultar útiles en el proceso de análisis.

Los algoritmos descritos en esta sección no preservan la propiedad del prefijo válido puesto que la fase predictiva no es lo suficientemente restrictiva como para evitar que las predicciones realizadas durante el análisis del pie de un árbol no conlleven el análisis de subárboles no válidos.

Una primera aproximación a un esquema de análisis sintáctico para un algoritmo de tipo Earley para gramáticas de adjunción consiste en transformar el esquema de análisis sintáctico **buE**. Al nuevo esquema de análisis, que presenta ciertas semejanzas con los algoritmos descritos por Schabes en [168, 169, 172], lo denominaremos **E** y su correspondiente sistema de análisis $\mathbb{P}_{\text{E}}$ se define a continuación.

Esquema de análisis sintáctico 3.4 El sistema de análisis $\mathbb{P}_{\text{E}}$ que se corresponde con una versión del algoritmo de Earley para TAG sin la propiedad del prefijo válido, dada una gramática de adjunción de árboles $\mathcal{T}$ y una cadena de entrada $a_1 \dots a_n$ se define como sigue:

$$\mathcal{I}_{\text{E}} = \mathcal{I}_{\text{buE}}$$

$$\mathcal{D}_{\mathrm{E}}^{\mathrm{Init}} = \frac{}{[\top \to \bullet \mathbf{R}^{\alpha}, 0, 0 \mid -, -]} \quad \alpha \in \boldsymbol{I}$$

$$\mathcal{D}_{\mathrm{E}}^{\mathrm{Scan}} = \mathcal{D}_{\mathrm{buE}}^{\mathrm{Scan}}$$

$$\mathcal{D}_{\mathrm{E}}^{\mathrm{Pred}} = \frac{[N^{\gamma} \to \delta \bullet M^{\gamma}\nu, i, j \mid p, q]}{[M^{\gamma} \to \bullet \upsilon, j, j \mid -, -]} \quad \mathbf{nil} \in \mathrm{adj}(M^{\gamma})$$

$$\mathcal{D}_{\mathrm{E}}^{\mathrm{Comp}} = \mathcal{D}_{\mathrm{buE}}^{\mathrm{Comp}}$$

$$\mathcal{D}_{\mathrm{E}}^{\mathrm{AdjPred}} = \frac{[N^{\gamma} \to \delta \bullet M^{\gamma}\nu, i, j \mid p, q]}{[\top \to \bullet \mathbf{R}^{\beta}, j, j \mid -, -]} \quad \beta \in \mathrm{adj}(M^{\gamma})$$

$$\mathcal{D}_{\mathrm{E}}^{\mathrm{FootPred}} = \frac{[\mathbf{F}^{\beta} \to \bullet \bot, k, k \mid -, -]}{[M^{\gamma} \to \bullet \delta, k, k \mid -, -]} \quad \beta \in \mathrm{adj}(M^{\gamma})$$

$$\mathcal{D}_{\mathrm{E}}^{\mathrm{FootComp}} = \frac{[M^{\gamma} \to \delta \bullet, k, l \mid p, q],\ [\mathbf{F}^{\beta} \to \bullet \bot, k, k \mid -, -]}{[\mathbf{F}^{\beta} \to \bot \bullet, k, l \mid k, l]} \quad \beta \in \mathrm{adj}(M^{\gamma})$$

$$\mathcal{D}_{\mathrm{E}}^{\mathrm{AdjComp}} = \mathcal{D}_{\mathrm{buE}}^{\mathrm{AdjComp}} = \frac{\begin{array}{c}[\top \to \mathbf{R}^{\beta} \bullet, j, m \mid k, l], \\ [M^{\gamma} \to \upsilon \bullet, k, l \mid p, q], \\ [N^{\gamma} \to \delta \bullet M^{\gamma}\nu, i, j \mid p', q']\end{array}}{[N^{\gamma} \to \delta M^{\gamma} \bullet \nu, i, m \mid p \cup p', q \cup q']} \quad \beta \in \mathrm{adj}(M^{\gamma})$$

$$\mathcal{D}_{\mathrm{E}} = \mathcal{D}_{\mathrm{E}}^{\mathrm{Init}} \cup \mathcal{D}_{\mathrm{E}}^{\mathrm{Scan}} \cup \mathcal{D}_{\mathrm{E}}^{\mathrm{Pred}} \cup \mathcal{D}_{\mathrm{E}}^{\mathrm{Comp}} \cup \mathcal{D}_{\mathrm{E}}^{\mathrm{AdjPred}} \cup \mathcal{D}_{\mathrm{E}}^{\mathrm{FootPred}} \cup \mathcal{D}_{\mathrm{E}}^{\mathrm{FootComp}} \cup \mathcal{D}_{\mathrm{E}}^{\mathrm{AdjComp}}$$

$$\mathcal{F}_{\mathrm{E}} = \mathcal{F}_{\mathrm{buE}}$$

§

El paso $\mathcal{D}_{\mathrm{E}}^{\mathrm{AdjComp}}$, aunque es idéntico al del sistema de análisis $\mathbb{P}_{\mathrm{buE}_1}$ se repite aquí para facilitar la comprensión del algoritmo. Como se ha dicho anteriormente, el esquema de análisis **E** presenta ciertas similitudes con los algoritmos descritos por Schabes en [168, 169, 172]. Podemos establecer la siguiente relación entre los pasos deductivos de **E** y los procesos definidos en dichos algoritmos, tal y como se muestra en la tabla 3.1. La razón del cambio de nombre en las pasos encargados de la adjunción se debe a que creemos que los nombres utilizados por Schabes pueden llevar a confusión, pues se tiende a pensar que cada *Predictor* está asociado con el *Completor* correspondiente a su mismo lado, cuando no es así. Es por ello que hemos decidido emparejar los pasos deductivos por las funciones que realizan: predicción-compleción de adjunción y predicción-compleción de pie.

Respecto al comportamiento del algoritmo, diremos que el análisis comienza creando un ítem correspondiente a una producción del nodo raíz de un árbol inicial con el punto situado

Pasos deductivos de E	Algoritmo de Schabes
$\mathcal{D}_{\mathrm{E}}^{\mathrm{Init}}$	*Initial item*
$\mathcal{D}_{\mathrm{E}}^{\mathrm{Scan}}$	Scanner
$\mathcal{D}_{\mathrm{E}}^{\mathrm{Pred}}$	Move Dot Down
$\mathcal{D}_{\mathrm{E}}^{\mathrm{Comp}}$	Move Dot Up
$\mathcal{D}_{\mathrm{E}}^{\mathrm{AdjPred}}$	Left Predictor
$\mathcal{D}_{\mathrm{E}}^{\mathrm{FootPred}}$	Left Completor
$\mathcal{D}_{\mathrm{E}}^{\mathrm{FootComp}}$	Right Predictor
$\mathcal{D}_{\mathrm{E}}^{\mathrm{AdjComp}}$	Right Completor

Tabla 3.1: Relación entre $\mathcal{D}_{\mathrm{E}}$ y el algoritmo tipo Earley sin VPP de Schabes

en el extremo izquierdo. Posteriormente, los pasos $\mathcal{D}_{\mathrm{E}}^{\mathrm{Pred}}$ y $\mathcal{D}_{\mathrm{E}}^{\mathrm{Comp}}$ se encargan de ir recorriendo el árbol de modo descendente y ascendente, respectivamente, de modo similar a como actúa el algoritmo de Earley para gramáticas independientes del contexto. Se puede predecir la adjunción de un árbol β en un nodo de un árbol elemental γ mediante la aplicación de un paso $\mathcal{D}_{\mathrm{E}}^{\mathrm{AdjPred}}$, con lo que se comienza el análisis del árbol β. Una vez alcanzado el pie de dicho árbol auxiliar, deberemos retomar el análisis de γ, concretamente del subárbol que pende del nodo de adjunción. El problema es que al no conocer en qué nodo de qué árbol elemental se ha producido la adjunción, deberemos predecir todos los posibles nodos donde esté permitida la adjunción de β, predicción realizada por un paso deductivo $\mathcal{D}_{\mathrm{E}}^{\mathrm{FootPred}}$. Es la predicción que se realiza en los pasos $\mathcal{D}_{\mathrm{E}}^{\mathrm{FootPred}}$ lo que provoca que el algoritmo no posea la propiedad del prefijo válido, puesto que se puede comenzar a analizar una parte de la cadena de entrada que no es gramatical, al predecir un subárbol que no se corresponde con el árbol desde el que se realizó la adjunción.

Una vez terminado de analizar todo el subárbol predicho por un paso $\mathcal{D}_{\mathrm{E}}^{\mathrm{FootPred}}$, deberemos retomar el análisis del árbol auxiliar β a partir del pie, tarea encomendada a los pasos deductivos $\mathcal{D}_{\mathrm{E}}^{\mathrm{FootComp}}$. Una vez terminado de analizar completamente el árbol auxiliar β, deberemos concluir la operación de adjunción aplicando un paso $\mathcal{D}_{\mathrm{E}}^{\mathrm{AdjComp}}$. Es en estos pasos en los que se verifica que el subárbol escindido del nodo de adjunción y el árbol auxiliar han sido correctamente reconstruidos. Las adjunciones simultáneas sobre un mismo nodo [175] son evitadas por los pasos $\mathcal{D}_{\mathrm{E}}^{\mathrm{AdjComp}}$ puesto que cuando se ha terminado de recorrer completamente el árbol auxiliar, se verifica que se ha analizado la parte correspondiente al subárbol del nodo de adjunción y se avanza el punto de la producción que contiene a este, sin cambiar el ítem correspondiente al nodo de adjunción. Si posteriormente otro árbol auxiliar es adjuntado en dicho nodo, representará una ambigüedad en el análisis sintáctico de la cadena de entrada pero no la adjunción simultánea de dos árboles auxiliares en un mismo nodo.

Proposición 3.3 $\mathbf{buE} \stackrel{\mathbf{df}}{\Longrightarrow} \mathbf{E}$.

Demostración:

Para demostrar que el esquema de análisis sintáctico **E** es el resultado de aplicar un filtrado dinámico al esquema de análisis **buE**, debemos demostrar que $\mathcal{I}_{\mathrm{buE}} \supseteq \mathcal{I}_{\mathrm{E}}$ y que $\vdash_{\mathrm{buE}} \supseteq \vdash_{\mathrm{E}}$ para los sistemas de análisis sintáctico $\mathbb{P}_{\mathrm{buE}}$ y $\mathbb{P}_{\mathrm{E}}$. Lo primero es cierto por definición puesto que $\mathcal{I}_{\mathrm{buE}} = \mathcal{I}_{\mathrm{E}}$. Respecto a lo segundo, es suficiente con mostrar que $\vdash_{\mathrm{buE}} \supseteq \mathcal{D}_{\mathrm{E}}$.

Los pasos $\mathcal{D}_{\mathrm{E}}^{\mathrm{Scan}}$, $\mathcal{D}_{\mathrm{E}}^{\mathrm{Comp}}$ y $\mathcal{D}_{\mathrm{E}}^{\mathrm{Adjcomp}}$ son idénticos a sus homónimos del sistema $\mathbb{P}_{\mathrm{buE}}$ y los pasos $\mathcal{D}_{\mathrm{E}}^{\mathrm{Init}}$ generan un subconjunto de los ítems generados por $\mathcal{D}_{\mathrm{buE}}^{\mathrm{Init}}$. Respecto a los otros pasos:

- Dado un paso $\frac{[N^\gamma \rightarrow \delta \bullet M^\gamma \nu, i, j | p, q]}{[M^\gamma \rightarrow \bullet \upsilon, j, j | -, -]} \in \mathcal{D}_{\mathrm{E}}^{\mathrm{Pred}}$ existe un paso $\frac{}{[M^\gamma \rightarrow \bullet \upsilon, j, j | -, -]} \in \mathcal{D}_{\mathrm{buE}}^{\mathrm{Init}}$ y por tanto existe la inferencia

$$[N^\gamma \rightarrow \delta \bullet M^\gamma \nu, i, j \mid p, q] \vdash_{\mathrm{buE}} [M^\gamma \rightarrow \bullet \upsilon, j, j \mid -, -]$$

- Dado un paso $\frac{[N^\gamma \rightarrow \delta \bullet M^\gamma \nu, i, j | p, q]}{[\top \rightarrow \bullet \mathbf{R}^\beta, j, j | -, -]} \in \mathcal{D}_{\mathrm{E}}^{\mathrm{AdjPred}}$ existe un paso $\frac{}{[\top \rightarrow \bullet \mathbf{R}^\beta, j, j | -, -]} \in \mathcal{D}_{\mathrm{buE}}^{\mathrm{Init}}$ y por tanto existe la inferencia

$$[N^\gamma \rightarrow \delta \bullet M^\gamma \nu, i, j \mid p, q] \vdash_{\mathrm{buE}} [\top \rightarrow \bullet \mathbf{R}^\beta, j, j \mid -, -]$$

- Dado un paso $\frac{[\mathbf{F}^\beta \rightarrow \bullet \bot, k, k | -, -]}{[M^\gamma \rightarrow \bullet \delta, k, k | -, -]} \in \mathcal{D}_{\mathrm{E}}^{\mathrm{FootPred}}$ existe un paso $\frac{}{[M^\gamma \rightarrow \bullet \delta, k, k | -, -]} \in \mathcal{D}_{\mathrm{buE}}^{\mathrm{Init}}$ y por tanto existe la inferencia

$$[\mathbf{F}^\beta \rightarrow \bullet \bot, k, k \mid -, -] \vdash_{\mathrm{buE}} [M^\gamma \rightarrow \bullet \delta, k, k \mid -, -]$$

- Dado un paso $\frac{[M^\gamma \rightarrow \delta \bullet, k, l | p, q],\ [\mathbf{F}^\beta \rightarrow \bullet \bot, k, k | -, -]}{[\mathbf{F}^\beta \rightarrow \bot \bullet, k, l | k, l]} \in \mathcal{D}_{\mathrm{E}}^{\mathrm{FootComp}}$ existe un paso $\frac{}{[\mathbf{F}^\beta \rightarrow \bot \bullet, k, l | k, l]} \in \mathcal{D}_{\mathrm{buE}}^{\mathrm{Foot}}$ y por tanto existe la inferencia

$$[M^\gamma \rightarrow \delta \bullet, k, l \mid p, q],\ [\mathbf{F}^\beta \rightarrow \bullet \bot, k, k \mid -, -] \vdash_{\mathrm{buE}} [\mathbf{F}^\beta \rightarrow \bot \bullet, k, l \mid k, l]$$

□

La complejidad temporal del esquema de análisis sintáctico **E** con respecto a la cadena de entrada es $\mathcal{O}(n^6)$ puesto que la aparente complejidad $\mathcal{O}(n^7)$ del paso deductivo $\mathcal{D}_{\mathrm{E}}^{\mathrm{AdjComp}}$ puede reducirse a $\mathcal{O}(n^6)$ mediante la aplicación parcial o *currificación* de dicho paso, ya que los índices l y m sólo involucran a los dos primeros ítems antecedentes.

Con el fin de definir un esquema de análisis sintáctico que se corresponda con un algoritmo más cercano al espíritu del algoritmo de Earley, debemos fortalecer la fase predictiva de los esquemas de análisis anteriores, puesto que estos no utilizan toda la información que tienen a su disposición. En concreto:

- Los pasos deductivos $\mathcal{D}_{\mathrm{E}}^{\mathrm{FootPred}}$ en los que se realiza la predicción del pie no comprueban que previamente se haya iniciado la adjunción del árbol β en el nodo M^γ.
- Idem para los pasos deductivos $\mathcal{D}_{\mathrm{E}}^{\mathrm{FootComp}}$ que finalizan el reconocimiento del pie.
- Los pasos deductivos $\mathcal{D}_{\mathrm{E}}^{\mathrm{FootComp}}$ no comprueban que el árbol auxiliar predicho para adjunción en el nodo M^γ sea el mismo que el que ha provocado el reconocimiento del subárbol enraizado en dicho nodo mediante la aplicación de los pasos FootPred.

A continuación definimos un nuevo esquema de análisis **Ear** derivado a partir de **E**, sobre el que hemos aplicado las siguientes modificaciones:

- La aplicación de un filtro dinámico a los pasos deductivos FootPred y FootComp consistente en la verificación de la existencia de los ítems que representan el comienzo de la operación de adjunción en el nodo M^γ.
- La aplicación de un refinamiento al paso AdjComp, que se divide en dos pasos AdjComp^1 y AdjComp^2, el primero realizando las comprobaciones pertinentes en el caso de que se haya producido la adjunción de un árbol auxiliar en un nodo de la espina de otro árbol

auxiliar. En la figura 3.4 se muestra una representación gráfica de la aplicación del paso deductivo AdjComp^1, el más complejo de los dos ya que el árbol γ en el que se realiza la adjunción es un árbol auxiliar y el nodo de adjunción pertenece a su espina. Las partes de los árboles involucrados que no se encuentran representados por los ítems que intervienen en el paso deductivo se muestran en línea discontinua si el algoritmo de análisis tiene que haber pasado obligatoriamente por dicha parte del árbol al menos en la fase predictiva y en línea punteada si se trata de partes que serán analizadas posteriormente.

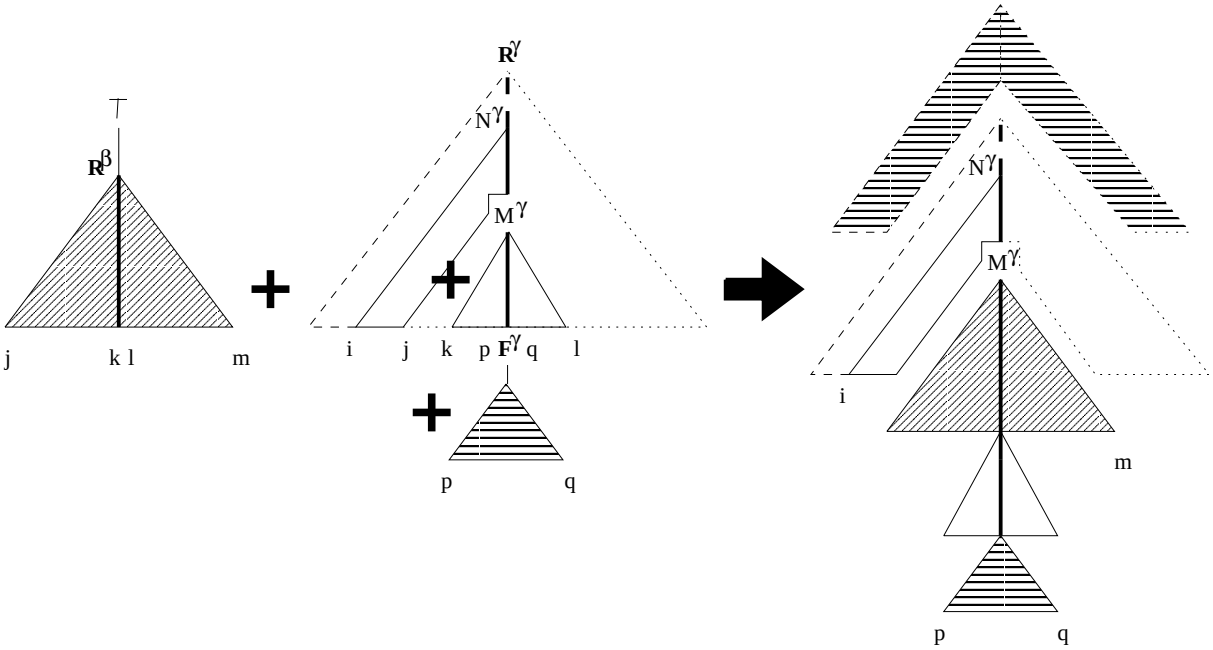

Figura 3.4: Descripción gráfica de un paso $\mathcal{D}_{\text{Ear}}^{\text{AdjComp}^1}$

Esquema de análisis sintáctico 3.5 El sistema de análisis $\mathbb{P}_{\text{Ear}}$ que se corresponde con una versión del algoritmo de Earley sin la propiedad del prefijo válido con predicción fuerte, dada una gramática de adjunción de árboles $\mathcal{T}$ y una cadena de entrada $a_1 \dots a_n$ se define como sigue:

$$\mathcal{I}_{\text{Ear}} = \mathcal{I}_{\text{buE}}$$

$$\mathcal{D}_{\text{Ear}}^{\text{Init}} = \mathcal{D}_{\text{E}}^{\text{Init}}$$

$$\mathcal{D}_{\text{Ear}}^{\text{Scan}} = \mathcal{D}_{\text{buE}}^{\text{Scan}}$$

$$\mathcal{D}_{\text{Ear}}^{\text{Pred}} = \mathcal{D}_{\text{E}}^{\text{Pred}}$$

$$\mathcal{D}_{\text{Ear}}^{\text{Comp}} = \mathcal{D}_{\text{buE}}^{\text{Comp}}$$

$$\mathcal{D}_{\text{Ear}}^{\text{AdjPred}} = \mathcal{D}_{\text{E}}^{\text{AdjPred}}$$

$$\mathcal{D}_{\text{Ear}}^{\text{FootPred}} = \frac{[\mathbf{F}^{\beta} \rightarrow \bullet\bot, k, k \mid -, -],\ [N^{\gamma} \rightarrow \delta \bullet M^{\gamma}\nu, i, j \mid p, q]}{[M^{\gamma} \rightarrow \bullet\delta, k, k \mid -, -]} \quad \beta \in \text{adj}(M^{\gamma})$$

$$\mathcal{D}_{\text{Ear}}^{\text{FootComp}} = \frac{\begin{array}{l}[M^{\gamma} \rightarrow \upsilon\bullet, k, l \mid p, q],\\ [\mathbf{F}^{\beta} \rightarrow \bullet\bot, k, k \mid -, -],\\ [N^{\gamma} \rightarrow \delta \bullet M^{\gamma}\nu, i, j \mid p', q']\end{array}}{[\mathbf{F}^{\beta} \rightarrow \bot\bullet, k, l \mid k, l]} \quad \begin{array}{l}\beta \in \text{adj}(M^{\gamma}),\\ p \cup p' \textit{ y } q \cup q' \textit{ está definido}\end{array}$$

$$\mathcal{D}_{\text{Ear}}^{\text{AdjComp}^1} = \frac{\begin{array}{l}[\top \rightarrow \mathbf{R}^{\beta}\bullet, j, m \mid k, l],\\ [M^{\gamma} \rightarrow \upsilon\bullet, k, l \mid p, q],\\ [\mathbf{F}^{\gamma} \rightarrow \bot\bullet, p, q \mid p, q],\\ [N^{\gamma} \rightarrow \delta \bullet M^{\gamma}\nu, i, j \mid -, -]\end{array}}{[N^{\gamma} \rightarrow \delta M^{\gamma} \bullet \nu, i, m \mid p, q]} \quad \beta \in \text{adj}(M^{\gamma})$$

$$\mathcal{D}_{\text{Ear}}^{\text{AdjComp}^2} = \frac{\begin{array}{l}[\top \rightarrow \mathbf{R}^{\beta}\bullet, j, m \mid k, l],\\ [M^{\gamma} \rightarrow \upsilon\bullet, k, l \mid -, -],\\ [N^{\gamma} \rightarrow \delta \bullet M^{\gamma}\nu, i, j \mid p', q']\end{array}}{[N^{\gamma} \rightarrow \delta M^{\gamma} \bullet \nu, i, m \mid p', q']} \quad \beta \in \text{adj}(M^{\gamma})$$

$$\mathcal{D}_{\text{Ear}} = \mathcal{D}_{\text{Ear}}^{\text{Init}} \cup \mathcal{D}_{\text{Ear}}^{\text{Scan}} \cup \mathcal{D}_{\text{Ear}}^{\text{Pred}} \cup \mathcal{D}_{\text{Ear}}^{\text{Comp}} \cup \mathcal{D}_{\text{Ear}}^{\text{AdjPred}} \cup \mathcal{D}_{\text{Ear}}^{\text{FootPred}} \cup \mathcal{D}_{\text{Ear}}^{\text{FootComp}} \cup \mathcal{D}_{\text{Ear}}^{\text{AdjComp}^1} \cup \mathcal{D}_{\text{Ear}}^{\text{AdjComp}^2}$$

$$\mathcal{F}_{\text{Ear}} = \mathcal{F}_{\text{buE}}$$

§

Proposición 3.4 $\mathbf{E} \overset{\mathbf{sr}}{\Longrightarrow} \mathbf{E}' \overset{\mathbf{df}}{\Longrightarrow} \mathbf{Ear}$.

Demostración:

Como primer paso definiremos el esquema de análisis $\mathbf{E}'$ que se obtiene a partir de $\mathbf{E}$ simplemente rompiendo el conjunto de pasos deductivos $\mathcal{D}_{\text{E}}^{\text{AdjComp}}$ en dos conjuntos $\mathcal{D}_{\text{E}'}^{\text{AdjComp}^1}$ y $\mathcal{D}_{\text{E}'}^{\text{AdjComp}^2}$. El sistema de análisis $\mathbb{P}_{\text{E}'}$ sería por tanto el siguiente:

$$\mathcal{I}_{\text{E}'} = \mathcal{I}_{\text{buE}}$$

$$\mathcal{D}_{\text{E}'}^{\text{Init}} = \mathcal{D}_{\text{E}}^{\text{Init}}$$
$$\mathcal{D}_{\text{E}'}^{\text{Scan}} = \mathcal{D}_{\text{buE}}^{\text{Scan}}$$
$$\mathcal{D}_{\text{E}'}^{\text{Pred}} = \mathcal{D}_{\text{E}}^{\text{Pred}}$$
$$\mathcal{D}_{\text{E}'}^{\text{Comp}} = \mathcal{D}_{\text{buE}}^{\text{Comp}}$$
$$\mathcal{D}_{\text{E}'}^{\text{AdjPred}} = \mathcal{D}_{\text{E}}^{\text{AdjPred}}$$
$$\mathcal{D}_{\text{E}'}^{\text{FootPred}} = \mathcal{D}_{\text{E}}^{\text{FootPred}}$$
$$\mathcal{D}_{\text{E}'}^{\text{FootComp}} = \mathcal{D}_{\text{E}}^{\text{FootComp}}$$

$$\mathcal{D}_{\text{E}'}^{\text{AdjComp}^1} = \frac{\begin{array}{l}[\top \rightarrow \mathbf{R}^{\beta}\bullet, j, m \mid k, l],\\ [M^{\gamma} \rightarrow \upsilon\bullet, k, l \mid p, q],\\ [N^{\gamma} \rightarrow \delta \bullet M^{\gamma}\nu, i, j \mid -, -]\end{array}}{[N^{\gamma} \rightarrow \delta M^{\gamma} \bullet \nu, i, m \mid p, q]} \quad \beta \in \text{adj}(M^{\gamma})$$

$$\mathcal{D}_{\text{E}'}^{\text{AdjComp}^2} = \frac{\begin{array}{l}[\top \rightarrow \mathbf{R}^{\beta}\bullet, j, m \mid k, l],\\ [M^{\gamma} \rightarrow \upsilon\bullet, k, l \mid -, -],\\ [N^{\gamma} \rightarrow \delta \bullet M^{\gamma}\nu, i, j \mid p', q']\end{array}}{[N^{\gamma} \rightarrow \delta M^{\gamma} \bullet \nu, i, m \mid p', q']} \quad \beta \in \text{adj}(M^{\gamma})$$

$$\mathcal{D}_{\text{E}'} = \mathcal{D}_{\text{E}'}^{\text{Init}} \cup \mathcal{D}_{\text{E}'}^{\text{Scan}} \cup \mathcal{D}_{\text{E}'}^{\text{Pred}} \cup \mathcal{D}_{\text{E}'}^{\text{Comp}} \cup \mathcal{D}_{\text{E}'}^{\text{AdjPred}} \cup \mathcal{D}_{\text{E}'}^{\text{FootPred}} \cup \mathcal{D}_{\text{E}'}^{\text{FootComp}} \cup \mathcal{D}_{\text{E}'}^{\text{AdjComp}^1} \cup \mathcal{D}_{\text{E}'}^{\text{AdjComp}^2}$$

$$\mathcal{F}_{\mathrm{E}'} = \mathcal{F}_{\mathrm{buE}}$$

Las condiciones a verificar son que $\mathcal{I}_{\mathrm{E}} \subseteq \mathcal{I}_{\mathrm{E}'}$ y que $\overset{*}{\vdash}_{\mathrm{E}} \subseteq \overset{*}{\vdash}_{\mathrm{E}'}$. La primera condición se verifica por la propia definición de los ítems mientras que la segunda se obtiene directamente considerando que el único cambio que se ha producido en $\mathbb{P}_{\mathrm{E}'}$ es hacer explícita la incompatibilidad del par de índices (p, q) con el par (p', q') de los pasos $\mathcal{D}_{\mathrm{E}}^{\mathrm{AdjComp}}$: si son p' y q' los índices que están definidos, entonces el paso $\mathcal{D}_{\mathrm{E}}^{\mathrm{AdjComp}}$ se convierte en $\mathcal{D}_{\mathrm{E}'}^{\mathrm{AdjComp}^1}$, mientras que si son p y q los índices que están definidos, entonces el paso $\mathcal{D}_{\mathrm{E}}^{\mathrm{AdjComp}}$ se convierte en $\mathcal{D}_{\mathrm{E}'}^{\mathrm{AdjComp}^2}$.

Para demostrar que el esquema de análisis sintáctico **Ear** es el resultado de aplicar un filtrado dinámico al esquema de análisis $\mathbf{E}'$, debemos demostrar que $\mathcal{I}_{\mathrm{E}'} \supseteq \mathcal{I}_{\mathrm{Ear}}$ y que $\vdash_{\mathrm{E}'} \supseteq \vdash_{\mathrm{Ear}}$ para los sistemas de análisis sintáctico $\mathbb{P}_{\mathrm{E}'}$ y $\mathbb{P}_{\mathrm{Ear}}$. Lo primero es cierto por definición puesto que $\mathcal{I}_{\mathrm{buE}} = \mathcal{I}_{\mathrm{E}'} = \mathcal{I}_{\mathrm{Ear}}$. Respecto a lo segundo es suficiente con mostrar que $\vdash_{\mathrm{E}'} \supseteq \vdash_{\mathrm{Ear}}$.

Los pasos $\mathcal{D}_{\mathrm{Ear}}^{\mathrm{Init}}$, $\mathcal{D}_{\mathrm{Ear}}^{\mathrm{Scan}}$, $\mathcal{D}_{\mathrm{Ear}}^{\mathrm{pred}}$, $\mathcal{D}_{\mathrm{Ear}}^{\mathrm{Comp}}$ y $\mathcal{D}_{\mathrm{Ear}}^{\mathrm{AdjPred}}$ son idénticos a sus homónimos del sistema $\mathbb{P}_{\mathrm{E}'}$. Respecto a los otros pasos:

$\mathcal{D}_{\mathrm{Ear}}^{\mathrm{FootPred}}$**:** Dado un paso $\frac{[\mathbf{F}^{\beta} \rightarrow \bullet\bot, k, k \mid -, -],\ [N^{\gamma} \rightarrow \delta \bullet M^{\gamma}\nu, i, j \mid p, q]}{[M^{\gamma} \rightarrow \bullet\delta, k, k \mid -, -]}$ existe un paso $\frac{[\mathbf{F}^{\beta} \rightarrow \bullet\bot, k, k \mid -, -]}{[M^{\gamma} \rightarrow \bullet\delta, k, k \mid -, -]} \in \mathcal{D}_{\mathrm{E}'}^{\mathrm{FootPred}}$ y por tanto existe la inferencia

$$[\mathbf{F}^{\beta} \rightarrow \bullet\bot, k, k \mid -, -],\ [N^{\gamma} \rightarrow \delta \bullet M^{\gamma}\nu, i, j \mid p, q] \vdash_{\mathrm{E}'} [M^{\gamma} \rightarrow \bullet\delta, k, k \mid -, -]$$

$\mathcal{D}_{\mathrm{Ear}}^{\mathrm{FootComp}}$**:** Dado un paso $\frac{[M^{\gamma} \rightarrow \upsilon\bullet, k, l \mid p, q],\ [\mathbf{F}^{\beta} \rightarrow \bullet\bot, k, k \mid -, -], [N^{\gamma} \rightarrow \delta\bullet M^{\gamma}\nu, i, j \mid p', q']}{[\mathbf{F}^{\beta} \rightarrow \bot\bullet, k, l \mid k, l]}$ existe un paso $\frac{[M^{\gamma} \rightarrow \upsilon\bullet, k, l \mid p, q],\ [\mathbf{F}^{\beta} \rightarrow \bullet\bot, k, k \mid -, -]}{[\mathbf{F}^{\beta} \rightarrow \bot\bullet, k, l \mid k, l]} \in \mathcal{D}_{\mathrm{E}'}^{\mathrm{FootComp}}$ y por tanto existe la inferencia

$$[M^{\gamma} \rightarrow \upsilon\bullet, k, l \mid p, q],\ [\mathbf{F}^{\beta} \rightarrow \bullet\bot, k, k \mid -, -],\ [N^{\gamma} \rightarrow \delta \bullet M^{\gamma}\nu, i, j \mid p', q'] \vdash_{\mathrm{E}'} [\mathbf{F}^{\beta} \rightarrow \bot\bullet, k, l \mid k, l]$$

$\mathcal{D}_{\mathrm{Ear}}^{\mathrm{AdjComp}^1}$**:** Dado un paso $\frac{[\top \rightarrow \mathbf{R}^{\beta}\bullet, j, m \mid k, l],\ [M^{\gamma} \rightarrow \upsilon\bullet, k, l \mid p, q],\ [\mathbf{F}^{\gamma} \rightarrow \bot\bullet, p, q \mid p, q],\ [N^{\gamma} \rightarrow \delta\bullet M^{\gamma}\nu, i, j \mid -, -]}{[N^{\gamma} \rightarrow \delta M^{\gamma}\bullet\nu, i, m \mid p, q]}$ existe un paso $\frac{[\top \rightarrow \mathbf{R}^{\beta}\bullet, j, m \mid k, l],\ [M^{\gamma} \rightarrow \upsilon\bullet, k, l \mid p, q],\ [N^{\gamma} \rightarrow \delta\bullet M^{\gamma}\nu, i, j \mid -, -]}{[N^{\gamma} \rightarrow \delta M^{\gamma}\bullet\nu, i, m \mid p, q]} \in \mathcal{D}_{\mathrm{E}'}^{\mathrm{AdjComp}}$ y por tanto existe la inferencia

$$[\top \rightarrow \mathbf{R}^{\beta}\bullet, j, m \mid k, l],\ [M^{\gamma} \rightarrow \upsilon\bullet, k, l \mid p, q],\ [\mathbf{F}^{\gamma} \rightarrow \bot\bullet, p, q \mid p, q],\ [N^{\gamma} \rightarrow \delta \bullet M^{\gamma}\nu, i, j \mid -, -] \vdash_{\mathrm{E}'} [N^{\gamma} \rightarrow \delta M^{\gamma} \bullet \nu, i, m \mid p, q]$$

$\mathcal{D}_{\mathrm{Ear}}^{\mathrm{AdjComp}^2}$**:** Dado un paso $\frac{[\top \rightarrow \mathbf{R}^{\beta}\bullet, j, m \mid k, l],\ [M^{\gamma} \rightarrow \upsilon\bullet, k, l \mid -, -],\ [N^{\gamma} \rightarrow \delta\bullet M^{\gamma}\nu, i, j \mid p, q]}{[N^{\gamma} \rightarrow \delta M^{\gamma}\bullet\nu, i, m \mid p', q']}$ existe un paso $\frac{[\top \rightarrow \mathbf{R}^{\beta}\bullet, j, m \mid k, l],\ [M^{\gamma} \rightarrow \upsilon\bullet, k, l \mid -, -],\ [N^{\gamma} \rightarrow \delta\bullet M^{\gamma}\nu, i, j \mid p', q']}{[N^{\gamma} \rightarrow \delta M^{\gamma}\bullet\nu, i, m \mid p', q']} \in \mathcal{D}_{\mathrm{E}'}^{\mathrm{AdjComp}}$ y por tanto existe la inferencia

$$[\top \rightarrow \mathbf{R}^{\beta}\bullet, j, m \mid k, l],\ [M^{\gamma} \rightarrow \upsilon\bullet, k, l \mid -, -],\ [N^{\gamma} \rightarrow \delta \bullet M^{\gamma}\nu, i, j \mid p', q'] \vdash_{\mathrm{E}'} [N^{\gamma} \rightarrow \delta M^{\gamma} \bullet \nu, i, m \mid p', q']$$

□

La complejidad temporal del esquema de análisis sintáctico **Ear** con respecto a la longitud n de la cadena de entrada es $\mathcal{O}(n^6)$ puesto que la aparente complejidad $\mathcal{O}(n^7)$ de los pasos deductivos $\mathcal{D}_{\mathrm{Ear}}^{\mathrm{AdjComp}^1}$ y $\mathcal{D}_{\mathrm{Ear}}^{\mathrm{AdjComp}^1}$ puede reducirse a $\mathcal{O}(n^6)$ mediante la aplicación parcial o *currificación* de dichos pasos, puesto que el índice l sólo involucra a los dos primeros ítems antecedentes en cada uno de ellos.

3.6. Algoritmos de tipo Earley con la propiedad del prefijo válido

El primer algoritmo de análisis sintáctico para TAG que satisfacía la propiedad del prefijo válido fue el descrito por Schabes y Joshi en [173] y por Schabes en [168]. La principal particularidad de dicho algoritmo es que su complejidad temporal con respecto a la cadena de entrada es $\mathcal{O}(n^7)$, tal como muestran Díaz Madrigal et al. en [65], mientras que los algoritmos sin la propiedad del prefijo válido presentan una complejidad $\mathcal{O}(n^6)$. Durante mucho tiempo cobró fuerza la opinión de que aquellos algoritmos que cumpliesen la propiedad del prefijo válido deberían tener una complejidad más alta que aquellos que no la cumpliesen. Sin embargo, Nederhof presentó en [125] un algoritmo para el análisis de TAG que cumple la propiedad del prefijo válido[6] y que presenta una complejidad temporal $\mathcal{O}(n^6)$. Veremos que dicho algoritmo es fácilmente derivable a partir del esquema de análisis **Ear** presentado en la sección anterior, correspondiente al algoritmo de tipo Earley sin la propiedad del prefijo válido.

El esquema de análisis sintáctico **Ear** describe un algoritmo que no cumple la propiedad del prefijo válido porque los pasos deductivos que se encargan de reconocer el nodo correspondiente al pie de un árbol auxiliar no pueden verificar la contigüidad de las fronteras del árbol al que pertenece el nodo de adjunción y del árbol auxiliar. Analicemos detalladamente dichos pasos:

- El paso deductivo $\mathcal{D}_{\text{Ear}}^{\text{FootPred}}$ puede verificar, mediante el ítem $[N^\gamma \rightarrow \delta \bullet M^\gamma \nu, i, j \mid p, q]$, que existe un nodo M^γ en el que el árbol auxiliar β puede ser adjuntado para reconocer la cadena de entrada a partir de la posición j. También puede verificar, mediante el ítem $[\mathbf{F}^\beta \rightarrow \bullet\bot, k, k \mid -, -]$, que se ha alcanzado el nodo pie del árbol auxiliar β. Lo que no puede verificar este paso deductivo es que el árbol al que pertenece dicho nodo pie se corresponda con la instancia de β que ha sido utilizada en la operación de adjunción que nos ocupa, pues para ello tendría que verificar que el extremo izquierdo de la frontera de la instancia de β es j, información que no es posible obtener a partir de los ítems definidos para el esquema de análisis **Ear**.

- El paso deductivo $\mathcal{D}_{\text{Ear}}^{\text{FootComp}}$ puede verificar, mediante el ítem $[N^\gamma \rightarrow \delta \bullet M^\gamma \nu, i, j \mid p, q]$, que existe un nodo M^γ en el que el árbol auxiliar β puede ser adjuntado para reconocer la cadena de entrada a partir de la posición j. También puede verificar, mediante el ítem $[\mathbf{F}^\beta \rightarrow \bullet\bot, k, k \mid -, -]$, que se ha alcanzado el nodo pie del árbol auxiliar β en la posición k de la cadena de entrada. Por último, el ítem $[M^\gamma \rightarrow \delta\bullet, k, l \mid p, q]$ permite verificar que la frontera del subárbol enraizado en M^γ comienza en la posición k de la cadena de entrada. Pero al igual que en el caso anterior y por las mismas razones, no puede verificar que el árbol al que pertenece el nodo pie se corresponde con la instancia de β que ha sido utilizada en la operación de adjunción que nos ocupa.

En consecuencia, para obtener un esquema de análisis que se corresponda con un algoritmo del tipo Earley para TAG que posea la propiedad del prefijo válido es necesario modificar la forma de los ítems para incluir un nuevo elemento, un índice que indique la posición del extremo izquierdo de la frontera del árbol al que se refieren los nodos de cada ítem que se genere. Esta operación se corresponde con la aplicación de un refinamiento de los ítems utilizados hasta el momento. Los nuevos ítems son de la forma

$$\left\{ [h, N^\gamma \rightarrow \delta \bullet \nu, i, j \mid p, q] \mid \begin{array}{l} \exists \alpha \in \boldsymbol{I},\ \mathbf{R}^\alpha \overset{*}{\Rightarrow} a_1 \dots a_h \mathbf{R}^\gamma \mu,\ \mathbf{R}^\gamma \overset{*}{\Rightarrow} a_{h+1} \dots a_i \delta \nu \upsilon \text{ y además:} \\ \delta \overset{*}{\Rightarrow} a_i \dots a_p\ \mathbf{F}^\gamma\ a_{q+1} \dots a_j \overset{*}{\Rightarrow} a_i \dots a_j \text{ sii } (p,q) \neq (-,-) \\ \delta \overset{*}{\Rightarrow} a_i \dots a_j \text{ sii } (p,q) = (-,-) \end{array} \right\}$$

[6] Para una comparación experimental entre los algoritmos de Schabes y Nederhof, consultar [63].

con lo cual un ítem $[N^\gamma \to \delta \bullet \nu, i, j \mid p, q]$ de **Ear** se corresponde ahora con el conjunto de ítems $[h, N^\gamma \to \delta \bullet \nu, i, j \mid p, q]\ \forall h \in [0, n]$.

Una vez definidos los nuevos ítems podemos pasar a describir el esquema de análisis **Earley** que corresponde a la primera versión de un algoritmo de tipo Earley para TAG que cumple la propiedad del prefijo válido. El correspondiente sistema de análisis sintáctico se define a continuación.

Esquema de análisis sintáctico 3.6 El sistema de análisis $\mathbb{P}_{\text{Earley}}$ que se corresponde con la el algoritmo de análisis sintáctico de tipo Earley para TAG que cumple la propiedad del prefijo válido, dada una gramática de adjunción de árboles $\mathcal{T}$ y una cadena de entrada $a_1 \dots a_n$ se define como sigue:

$$\mathcal{I}_{\text{Earley}} = \left\{ [h, N^\gamma \to \delta \bullet \nu, i, j \mid p, q] \mid \begin{array}{l} N^\gamma \to \delta\nu \in \mathcal{P}(\gamma),\ \gamma \in \boldsymbol{I} \cup \boldsymbol{A}, \\ 0 \leq h \leq i \leq j,\ (p, q) \leq (i, j) \end{array} \right\}$$

$$\mathcal{D}^{\text{Init}}_{\text{Earley}} = \frac{}{\vdash [0, \top \to \bullet \mathbf{R}^\alpha, 0, 0 \mid -, -]}\ \alpha \in \boldsymbol{I}$$

$$\mathcal{D}^{\text{Scan}}_{\text{Earley}} = \frac{[h, N^\gamma \to \delta \bullet a\nu, i, j \mid p, q],\ [a, j, j+1]}{[h, N^\gamma \to \delta a \bullet \nu, i, j+1 \mid p, q]}$$

$$\mathcal{D}^{\text{Pred}}_{\text{Earley}} = \frac{[h, N^\gamma \to \delta \bullet M^\gamma \nu, i, j \mid p, q]}{[h, M^\gamma \to \bullet \upsilon, j, j \mid -, -]}\ \mathbf{nil} \in \text{adj}(M^\gamma)$$

$$\mathcal{D}^{\text{Comp}}_{\text{Earley}} = \frac{[h, N^\gamma \to \delta \bullet M^\gamma \nu, i, k \mid p, q],\ [h, M^\gamma \to \upsilon \bullet, k, j \mid p', q']}{[h, N^\gamma \to \delta M^\gamma \bullet \nu, i, j \mid p \cup p', q \cup q']}\ \mathbf{nil} \in \text{adj}(M^\gamma)$$

$$\mathcal{D}^{\text{AdjPred}}_{\text{Earley}} = \frac{[h, N^\gamma \to \delta \bullet M^\gamma \nu, i, j \mid p, q]}{[j, \top \to \bullet \mathbf{R}^\beta, j, j \mid -, -]}\ \beta \in \text{adj}(M^\gamma)$$

$$\mathcal{D}^{\text{FootPred}}_{\text{Earley}} = \frac{[j, \mathbf{F}^\beta \to \bullet \bot, k, k \mid -, -],\ [h, N^\gamma \to \delta \bullet M^\gamma \nu, i, j \mid p, q]}{[h, M^\gamma \to \bullet \delta, k, k \mid -, -]}\ \beta \in \text{adj}(M^\gamma)$$

$$\mathcal{D}^{\text{FootComp}}_{\text{Earley}} = \frac{\begin{array}{l}[h, M^\gamma \to \delta \bullet, k, l \mid p, q], \\ [j, \mathbf{F}^\beta \to \bullet \bot, k, k \mid -, -], \\ [h, N^\gamma \to \delta \bullet M^\gamma \nu, i, j \mid p', q']\end{array}}{[j, \mathbf{F}^\beta \to \bot \bullet, k, l \mid k, l]}\quad \begin{array}{l}\beta \in \text{adj}(M^\gamma), \\ p \cup p' \text{ y } q \cup q' \text{ está definido}\end{array}$$

$$\mathcal{D}^{\text{AdjComp}^1}_{\text{Earley}} = \frac{\begin{array}{l}[j, \top \to \mathbf{R}^\beta \bullet, j, m \mid k, l], \\ [h, M^\gamma \to \upsilon \bullet, k, l \mid p, q], \\ [h, \mathbf{F}^\gamma \to \bot \bullet, p, q \mid p, q], \\ [h, N^\gamma \to \delta \bullet M^\gamma \nu, i, j \mid -, -]\end{array}}{[h, N^\gamma \to \delta M^\gamma \bullet \nu, i, m \mid p, q]}\ \beta \in \text{adj}(M^\gamma)$$

$$\mathcal{D}^{\text{AdjComp}^2}_{\text{Earley}} = \frac{\begin{array}{l}[j, \top \rightarrow \mathbf{R}^{\beta}\bullet, j, m \mid k, l],\\ [h, M^{\gamma} \rightarrow \upsilon\bullet, k, l \mid -, -],\\ [h, N^{\gamma} \rightarrow \delta \bullet M^{\gamma}\nu, i, j \mid p', q']\end{array}}{[h, N^{\gamma} \rightarrow \delta M^{\gamma} \bullet \nu, i, m \mid p', q']} \quad \beta \in \text{adj}(M^{\gamma})$$

$$\begin{aligned}\mathcal{D}_{\text{Earley}} = \; & \mathcal{D}^{\text{Init}}_{\text{Earley}} \cup \mathcal{D}^{\text{Scan}}_{\text{Earley}} \cup \mathcal{D}^{\text{Pred}}_{\text{Earley}} \cup \mathcal{D}^{\text{Comp}}_{\text{Earley}} \cup \mathcal{D}^{\text{AdjPred}}_{\text{Earley}} \\ & \cup \mathcal{D}^{\text{FootPred}}_{\text{Earley}} \cup \mathcal{D}^{\text{FootComp}}_{\text{Earley}} \cup \mathcal{D}^{\text{AdjComp}^1}_{\text{Earley}} \cup \mathcal{D}^{\text{AdjComp}^2}_{\text{Earley}}\end{aligned}$$

$$\mathcal{F}_{\text{Earley}} = \left\{ \; [0, \top \rightarrow \mathbf{R}^{\alpha}\bullet, 0, n \mid -, -] \mid \alpha \in \boldsymbol{I} \; \right\}$$

En la figura 3.5 se muestra una representación gráfica de la aplicación del paso deductivo $\mathcal{D}^{\text{AdjComp}^1}_{\text{Earley}}$. Es interesante comparar este nueva figura con la 3.4 correspondiente al mismo paso deductivo del algoritmo de tipo Earley sin la propiedad del prefijo válido con el fin de observar cómo se restringen los árboles candidatos en la aplicación del paso deductivo. Se puede observar que la única diferencia entre ambas radica en que el extremo izquierdo del árbol γ está explícitamente indicado por el índice h en la figura 3.5, mientras que en la figura 3.4 se consideraba universalmente cuantificado.

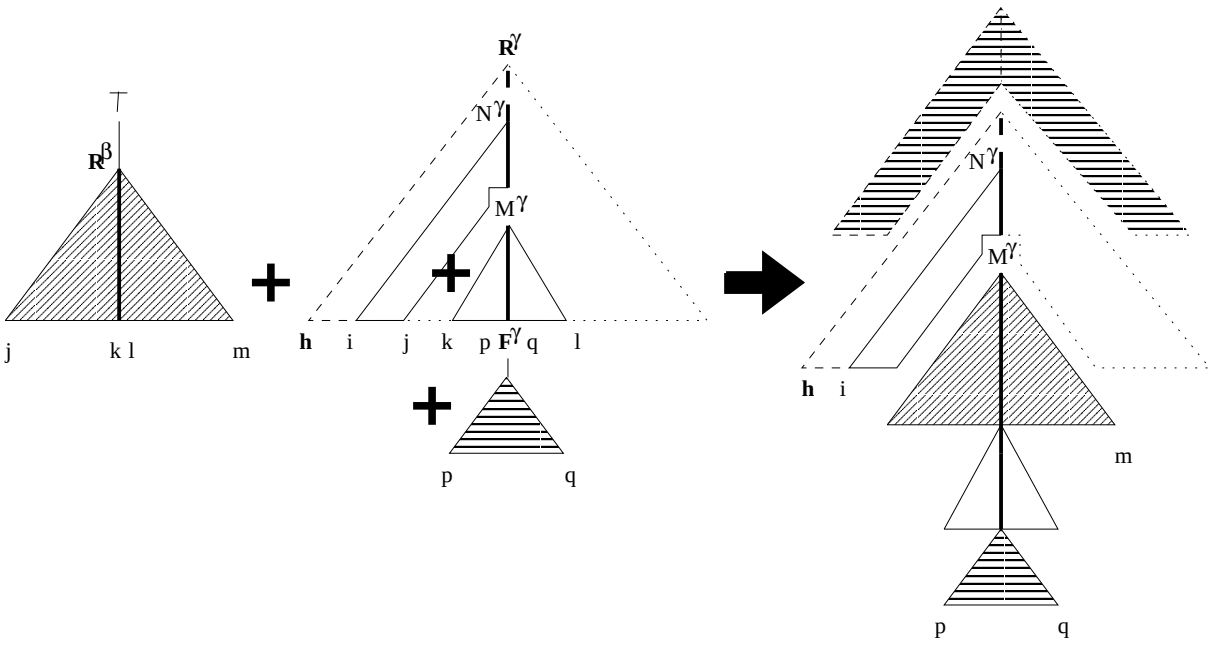

Figura 3.5: Descripción gráfica de un paso $\mathcal{D}^{\text{AdjComp}^1}_{\text{Earley}}$

Proposición 3.5 Ear $\overset{\text{ir}}{\Longrightarrow}$ Earley.

Demostración:

Para demostrar que el esquema de análisis **Earley** es derivable del esquema de análisis **Ear** mediante refinamiento de los ítems definiremos la siguiente función:

$$f([h, N^\gamma \rightarrow \delta \bullet \nu, i, j \mid p, q \mid adj]) = [N^\gamma \rightarrow \delta \bullet \nu, i, j \mid p, q \mid adj]$$

de la cual se obtiene directamente que $\mathcal{I}_{\text{Ear}} = f(\mathcal{I}_{\text{Earley}})$ y que $\Delta_{\text{Ear}} = f(\Delta_{\text{Earley}})$ por inducción en la longitud de las secuencias de derivación. En consecuencia, $\mathbb{P}_{\text{Ear}} \xrightarrow{\textbf{ir}} \mathbb{P}_{\text{Earley}}$, con lo que hemos probado lo que pretendíamos. □

Un aspecto interesante a tener en cuenta es que el ítem $[h, \mathbf{F}^\gamma \rightarrow \bot\bullet, p, q \mid p, q]$ es redundante en el paso $\mathcal{D}_{\text{Earley}}^{\text{AdjComp}^1}$, puesto que su existencia viene implícitamente determinada por la existencia del ítem $[h, M^\gamma \rightarrow \delta\bullet, k, l \mid p, q]$, ya que en otro caso este último sería inconsistente y por consiguiente algoritmo sería incorrecto. La finalidad de su presencia en dicho conjunto de pasos deductivos, así como en $\mathcal{D}_{\text{Ear}}^{\text{AdjComp}^1}$, es facilitar la transición hacia el esquema de análisis **Nederhof**. Si prescindimos de dicho ítem los pasos deductivos $\mathcal{D}_{\text{Earley}}^{\text{AdjComp}^1}$ y $\mathcal{D}_{\text{Earley}}^{\text{AdjComp}^2}$ se podrían fundir en uno solo:

$$\mathcal{D}_{\text{Earley}}^{\text{AdjComp}} = \frac{\begin{array}{c}[j, \top \rightarrow \mathbf{R}^\beta\bullet, j, m \mid k, l],\\ [h, M^\gamma \rightarrow \upsilon\bullet, k, l \mid p, q],\\ [h, N^\gamma \rightarrow \delta \bullet M^\gamma\nu, i, j \mid p', q']\end{array}}{[h, N^\gamma \rightarrow \delta M^\gamma \bullet \nu, i, m \mid p \cup p', q \cup q']} \quad \beta \in \text{adj}(M^\gamma)$$

Por idénticas razones, los pasos $\mathcal{D}_{\text{Ear}}^{\text{AdjComp}^1}$ y $\mathcal{D}_{\text{Ear}}^{\text{AdjComp}^1}$ del esquema **Ear** podrían fundirse en un nuevo paso $\mathcal{D}_{\text{Ear}}^{\text{AdjComp}}$:

$$\mathcal{D}_{\text{Ear}}^{\text{AdjComp}} = \frac{\begin{array}{c}[\top \rightarrow \mathbf{R}^\beta\bullet, j, m \mid k, l],\\ [M^\gamma \rightarrow \upsilon\bullet, k, l \mid p, q],\\ [N^\gamma \rightarrow \delta \bullet M^\gamma\nu, i, j \mid p', q']\end{array}}{[N^\gamma \rightarrow \delta M^\gamma \bullet \nu, i, m \mid p \cup p', q \cup q']} \quad \beta \in \text{adj}(M^\gamma)$$

En consecuencia, en lugar de la evolución $\mathbf{E} \overset{\textbf{sr}}{\Longrightarrow}\overset{\textbf{df}}{\Longrightarrow} \mathbf{Ear} \overset{\textbf{ir}}{\Longrightarrow} \mathbf{Earley}$ podríamos haber definido una línea evolutiva $\mathbf{E} \overset{\textbf{df}}{\Longrightarrow} \mathbf{Ear}' \overset{\textbf{ir}}{\Longrightarrow} \mathbf{Earley}' \overset{\textbf{sr}}{\Longrightarrow}\overset{\textbf{df}}{\Longrightarrow} \mathbf{Earley}$, donde $\mathbf{Ear}'$ y $\mathbf{Earley}'$ son como **Ear** y **Earley**, respectivamente, excepto por la sustitución de los pasos AdjComp^1 y AdjComp^2 por AdjComp. Este resultado viene a mostrar una vez más que existen varios caminos para transformar un esquema de análisis sintáctico en otro, tal como establece Sikkel en [189] para el caso de los algoritmos de análisis de gramáticas independientes del contexto y que nosotros mostramos aquí para el caso de las gramáticas de adjunción de árboles.

El algoritmo descrito por el esquema **Earley** presenta una complejidad temporal de $\mathcal{O}(n^7)$. Aunque aparentemente la utilización de 8 índices con respecto a la cadena de entrada en los pasos deductivos $\mathcal{D}_{\text{Earley}}^{\text{AdjComp}^1}$ y $\mathcal{D}_{\text{Earley}}^{\text{AdjComp}^2}$ hace pensar en una complejidad $\mathcal{O}(n^8)$, la utilización de aplicación parcial o *currificación* en dichos pasos reduce la complejidad hasta $\mathcal{O}(n^7)$. En el caso de $\mathcal{D}_{\text{Earley}}^{\text{AdjComp}^1}$ la aplicación parcial sobre los dos primeros ítems involucra combinar 7 índices, de donde resulta la complejidad $\mathcal{O}(n^7)$ con respecto a la cadena de entrada, pero únicamente los 5 índices j, m, h, p y q forman parte del resultado intermedio puesto que son los únicos que se necesitarán en posteriores aplicaciones parciales. La siguiente aplicación parcial combina el ítem intermedio con el tercer ítem del paso deductivo, operación que involucra únicamente a los 5 ítems mencionados anteriormente, que son conservados en el resultado intermedio producido. Por último, la aplicación parcial con el cuarto ítem involucra la combinación de los 6 índices h, i, j, m, p, q. El caso de $\mathcal{D}_{\text{Earley}}^{\text{AdjComp}^2}$ es análogo al de $\mathcal{D}_{\text{Earley}}^{\text{AdjComp}^1}$.

El aumento de la complejidad de $\mathcal{O}(n^6)$ a $\mathcal{O}(n^7)$ se debe al índice adicional incorporado en los ítems y que indica la posición del extremo izquierdo de la frontera del árbol que se está considerando. La inclusión en los ítems de este índice había surgido por la necesidad de controlar que se están utilizando los árboles correctos para reconocer el pie de un árbol auxiliar. En consecuencia, dicho índice sólo es de real utilidad en los pasos FootPred y FootComp. El resto de los pasos deductivos únicamente propagan el valor de dicho índice. Por consiguiente, en caso de que sea necesario estos últimos pasos deductivos pueden ser refinados, dividiéndolos en varios pasos con el fin de generar ítems intermedios carentes de dicho índice. Esta técnica, si se aplica convenientemente, puede llegar a reducir la complejidad de los algoritmos de análisis. Evidentemente, hay que verificar que los ítems intermedios portan la información necesaria para que el resultado de la composición de los pasos deductivos obtenidos mediante el refinamiento de uno dado sea equivalente al resultado obtenido por aplicar directamente el paso deductivo sin refinar.

En el caso completo del esquema de análisis **Earley**, para reducir la complejidad a $\mathcal{O}(n^6)$ es suficiente con hacer uso de la *propiedad de independencia del contexto de TAG* [214]. Básicamente, lo que dicha propiedad establece es que cada operación de adjunción es independiente de la previa o posterior aplicación de cualquier otra operación de adjunción. Una consecuencia de esta propiedad es que si en un nodo M^γ de un árbol γ está permitida la adjunción de un árbol auxiliar β y se cumplen las tres condiciones siguientes:

1. la parte izquierda de la frontera de $\gamma - M^\gamma$ se extiende desde la posición h hasta la posición j de la cadena de entrada, donde $\gamma - M^\gamma$ denota el resultado de escindir el subárbol enraizado por M^γ de γ;

2. la frontera del árbol β se expande desde la posición j hasta la posición m de la cadena de entrada con una discontinuidad en el pie desde la posición k hasta la l;

3. la frontera del subárbol enraizado por M^γ abarca precisamente desde la posición k hasta la posición l;

como resultado de la adjunción de β en M^γ la frontera del subárbol enraizado por este último nodo se expande desde las posición j hasta la m sin discontinuidad y dicho subárbol puede ser insertado en todo árbol $\beta - M^\gamma$ cuya frontera izquierda finalice en la posición j independientemente de la posición h en la que se sitúe el extremo izquierdo de dicha frontera. Precisamente, los algoritmos de tipo Earley para TAG que no cumplen la propiedad del prefijo válido hacen uso de esta propiedad para verificar la corrección de la adjunción realizada en los pasos AdjComp. Como se recordará de la sección precedente, los ítems de los esquemas de análisis de dichos algoritmos no incorporan el índice h del extremo izquierdo.

En consecuencia, los ítems que utilizaremos en el esquema de análisis sintáctico **Nederhof** correspondiente al algoritmo de tipo Earley para TAG presentado por Nederhof en [125] que preserva la propiedad del prefijo válido con una complejidad $\mathcal{O}(n^6)$, son de dos tipos: los definidos para el esquema **Earley** y los pseudo-ítem que definimos a continuación

$$\left\{ [[N^\gamma \rightarrow \delta \bullet \nu, i, j \mid p, q]] \mid \begin{array}{ll} \delta \stackrel{*}{\Rightarrow} a_i \ldots a_p \, \mathbf{F}^\gamma \, a_{q+1} \ldots a_j \stackrel{*}{\Rightarrow} a_i \ldots a_j & \text{sii } (p,q) \neq (-,-) \\ \delta \stackrel{*}{\Rightarrow} a_i \ldots a_j & \text{sii } (p,q) = (-,-) \end{array} \right\}$$

Ahora podemos definir el esquema de análisis **Nederhof** cuyo sistema de análisis mostramos a continuación.

Esquema de análisis sintáctico 3.7 El sistema de análisis $\mathbb{P}_{\text{Nederhof}}$ que se corresponde con la el algoritmo de análisis sintáctico de tipo Earley para TAG que cumple la propiedad del prefijo válido y posee una complejidad $\mathcal{O}(n^6)$, dada una gramática de adjunción de árboles $\mathcal{T}$ y una cadena de entrada $a_1 \dots a_n$ se define como sigue:

$$\mathcal{I}^{(1)}_{\text{Nederhof}} = \mathcal{I}_{\text{Earley}} = \left\{ [h, N^\gamma \to \delta \bullet \nu, i, j \mid p, q] \mid \begin{array}{l} N^\gamma \to \delta\nu \in \mathcal{P}(\gamma),\ \gamma \in \boldsymbol{I} \cup \boldsymbol{A}, \\ 0 \leq h \leq i \leq j,\ (p,q) \leq (i,j) \end{array} \right\}$$

$$\mathcal{I}^{(2)}_{\text{Nederhof}} = \left\{ [[N^\gamma \to \delta \bullet \nu, i, j \mid p, q]] \mid \begin{array}{l} N^\gamma \to \delta\nu \in \mathcal{P}(\gamma),\ \gamma \in \boldsymbol{I} \cup \boldsymbol{A}, \\ 0 \leq h \leq i \leq j,\ (p,q) \leq (i,j) \end{array} \right\}$$

$$\mathcal{I}_{\text{Nederhof}} = \mathcal{I}^{(1)}_{\text{Nederhof}} \cup \mathcal{I}^{(2)}_{\text{Nederhof}}$$

$$\mathcal{D}^{\text{Init}}_{\text{Nederhof}} = \mathcal{D}^{\text{Init}}_{\text{Earley}} = \frac{}{\vdash [0, \top \to \bullet \mathbf{R}^\alpha, 0, 0 \mid -, -]} \quad \alpha \in \boldsymbol{I}$$

$$\mathcal{D}^{\text{Scan}}_{\text{Nederhof}} = \mathcal{D}^{\text{Scan}}_{\text{Earley}} = \frac{[h, N^\gamma \to \delta \bullet a\nu, i, j \mid p, q],\ [a, j, j+1]}{[h, N^\gamma \to \delta a \bullet \nu, i, j+1 \mid p, q]}$$

$$\mathcal{D}^{\text{Pred}}_{\text{Nederhof}} = \mathcal{D}^{\text{Pred}}_{\text{Earley}} = \frac{[h, N^\gamma \to \delta \bullet M^\gamma\nu, i, j \mid p, q]}{[h, M^\gamma \to \bullet\upsilon, j, j \mid -, -]} \quad \mathbf{nil} \in \text{adj}(M^\gamma)$$

$$\mathcal{D}^{\text{Comp}}_{\text{Nederhof}} = \mathcal{D}^{\text{Comp}}_{\text{Earley}} = \frac{[h, N^\gamma \to \delta \bullet M^\gamma\nu, i, k \mid p, q],\ [h, M^\gamma \to \upsilon\bullet, k, j \mid p', q']}{[h, N^\gamma \to \delta M^\gamma \bullet \nu, i, j \mid p \cup p', q \cup q']} \quad \mathbf{nil} \in \text{adj}(M^\gamma)$$

$$\mathcal{D}^{\text{AdjPred}}_{\text{Nederhof}} = \mathcal{D}^{\text{AdjPred}}_{\text{Earley}} = \frac{[h, N^\gamma \to \delta \bullet M^\gamma\nu, i, j \mid p, q]}{[j, \top \to \bullet \mathbf{R}^\beta, j, j \mid -, -]} \quad \beta \in \text{adj}(M^\gamma)$$

$$\mathcal{D}^{\text{FootPred}}_{\text{Nederhof}} = \mathcal{D}^{\text{FootPred}}_{\text{Earley}} = \frac{[j, \mathbf{F}^\beta \to \bullet\bot, k, k \mid -, -],\ [h, N^\gamma \to \delta \bullet M^\gamma\nu, i, j \mid p, q]}{[h, M^\gamma \to \bullet\delta, k, k \mid -, -]} \quad \beta \in \text{adj}(M^\gamma)$$

$$\mathcal{D}^{\text{FootComp}}_{\text{Nederhof}} = \mathcal{D}^{\text{FootComp}}_{\text{Earley}} = \frac{\begin{array}{l}[h, M^\gamma \to \upsilon\bullet, k, l \mid p, q], \\ [j, \mathbf{F}^\beta \to \bullet\bot, k, k \mid -, -], \\ [h, N^\gamma \to \delta \bullet M^\gamma\nu, i, j \mid p', q']\end{array}}{[j, \mathbf{F}^\beta \to \bot\bullet, k, l \mid k, l]} \quad \begin{array}{l}\beta \in \text{adj}(M^\gamma), \\ p \cup p' \text{ y } q \cup q' \textit{ está definido}\end{array}$$

$$\mathcal{D}^{\text{AdjComp}^0}_{\text{Nederhof}} = \frac{\begin{array}{l}[j, \top \to \mathbf{R}^\beta\bullet, j, m \mid k, l], \\ [h, M^\gamma \to \upsilon\bullet, k, l \mid p, q],\end{array}}{[[M^\gamma \to \upsilon\bullet, j, m \mid p, q]]} \quad \beta \in \text{adj}(M^\gamma)$$

$$\mathcal{D}^{\text{AdjComp}^1}_{\text{Nederhof}} = \frac{\begin{array}{l}[[M^\gamma \to \upsilon\bullet, j, m \mid p, q]], \\ [h, \mathbf{F}^\gamma \to \bot\bullet, p, q \mid p, q], \\ [h, N^\gamma \to \delta \bullet M^\gamma\nu, i, j \mid -, -]\end{array}}{[h, N^\gamma \to \delta M^\gamma \bullet \nu, i, m \mid p, q]}$$

$$\mathcal{D}^{\text{AdjComp}^2}_{\text{Nederhof}} = \frac{[[M^\gamma \rightarrow \upsilon\bullet, j, m \mid -, -]],\ [h, N^\gamma \rightarrow \delta \bullet M^\gamma \nu, i, j \mid p', q']}{[h, N^\gamma \rightarrow \delta M^\gamma \bullet \nu, i, m \mid p', q']}$$

$$\mathcal{D}_{\text{Nederhof}} = \mathcal{D}^{\text{Init}}_{\text{Nederhof}} \cup \mathcal{D}^{\text{Scan}}_{\text{Nederhof}} \cup \mathcal{D}^{\text{Pred}}_{\text{Nederhof}} \cup \mathcal{D}^{\text{Comp}}_{\text{Nederhof}} \cup \mathcal{D}^{\text{AdjPred}}_{\text{Nederhof}} \cup \mathcal{D}^{\text{FootPred}}_{\text{Nederhof}} \cup \mathcal{D}^{\text{FootComp}}_{\text{Nederhof}} \cup \mathcal{D}^{\text{AdjComp}^0}_{\text{Nederhof}} \cup \mathcal{D}^{\text{AdjComp}^1}_{\text{Nederhof}} \cup \mathcal{D}^{\text{AdjComp}^2}_{\text{Nederhof}}$$

$$\mathcal{F}_{\text{Nederhof}} = \mathcal{F}_{\text{Earley}} = \left\{ \ [0, \top \rightarrow \mathbf{R}^\alpha\bullet, 0, n \mid -, -] \mid \alpha \in \boldsymbol{I} \ \right\}$$

§

Obsérvese que se ha aplicado un refinamiento al paso deductivo $\mathcal{D}^{\text{AdjComp}^1}_{\text{Earley}}$ para obtener los pasos $\mathcal{D}^{\text{AdjComp}^0}_{\text{Nederhof}}$ y $\mathcal{D}^{\text{AdjComp}^1}_{\text{Nederhof}}$. Análogamente, el paso deductivo $\mathcal{D}^{\text{AdjComp}^2}_{\text{Earley}}$ ha sido refinado en los pasos $\mathcal{D}^{\text{AdjComp}^0}_{\text{Nederhof}}$ y $\mathcal{D}^{\text{AdjComp}^2}_{\text{Nederhof}}$. Para garantizar la corrección se verifica que:

- La aplicación consecutiva de $\mathcal{D}^{\text{AdjComp}^0}_{\text{Nederhof}}$ y $\mathcal{D}^{\text{AdjComp}^1}_{\text{Earley}}$ (resp.$\mathcal{D}^{\text{AdjComp}^0}_{\text{Nederhof}}$ y $\mathcal{D}^{\text{AdjComp}^2}_{\text{Nederhof}}$) es equivalente a la aplicación de $\mathcal{D}^{\text{AdjComp}^1}_{\text{Earley}}$ (resp. $\mathcal{D}^{\text{AdjComp}^2}_{\text{Earley}}$). Es fácil comprobar que ambos utilizan la misma información: todos los antecedentes de $\mathcal{D}^{\text{AdjComp}^1}_{\text{Nederhof}}$ (resp. $\mathcal{D}^{\text{AdjComp}^2}_{\text{Nederhof}}$) son utilizados por pasos del esquema **Nederhof** y toda la información presente en los antecedentes de los pasos del esquema **Nederhof** es utilizada por el paso $\mathcal{D}^{\text{AdjComp}^1}_{\text{Nederhof}}$ (resp. $\mathcal{D}^{\text{AdjComp}^2}_{\text{Nederhof}}$), puesto que el ítem intermedio no crea nueva información, sino que simplemente es un "resumen" de información contenida en los otros antecedentes. Es también fácil comprobar que en ambos casos se genera la misma información, puesto que los ítems generados (excluyendo pseudo-ítems) son idénticos en ambos casos.

- El paso deductivo $\mathcal{D}^{\text{AdjComp}^1}_{\text{Nederhof}}$ (resp. $\mathcal{D}^{\text{AdjComp}^2}_{\text{Nederhof}}$) solo puede ser aplicado si previamente se ha aplicado el paso $\mathcal{D}^{\text{AdjComp}^0}_{\text{Nederhof}}$. Se puede verificar fácilmente puesto que el paso $\mathcal{D}^{\text{AdjComp}^0}_{\text{Nederhof}}$ genera un ítem intermedio y los únicos pasos que toman un ítem intermedio como antecedente son $\mathcal{D}^{\text{AdjComp}^1}_{\text{Nederhof}}$ y $\mathcal{D}^{\text{AdjComp}^2}_{\text{Nederhof}}$.

En la figura 3.5 se muestra una representación gráfica de la aplicación de los pasos deductivos $\mathcal{D}^{\text{AdjComp}^0}_{\text{Nederhof}}$ y $\mathcal{D}^{\text{AdjComp}^1}_{\text{Nederhof}}$.

La complejidad del algoritmo descrito por el esquema de análisis **Nederhof** es $\mathcal{O}(n^6)$, puesto que en la combinación de los ítems de cualquier paso intervienen activamente a los sumo 6 índices con respecto a la cadena de entrada.

En [64] se describe una versión de este algoritmo en la que se utiliza una representación plana de los árboles elementales en lugar de la representación multicapa que se ha utilizado aquí.

Proposición 3.6 **Earley** $\overset{\mathbf{sr}}{\Longrightarrow}$ **Nederhof**.

Demostración:

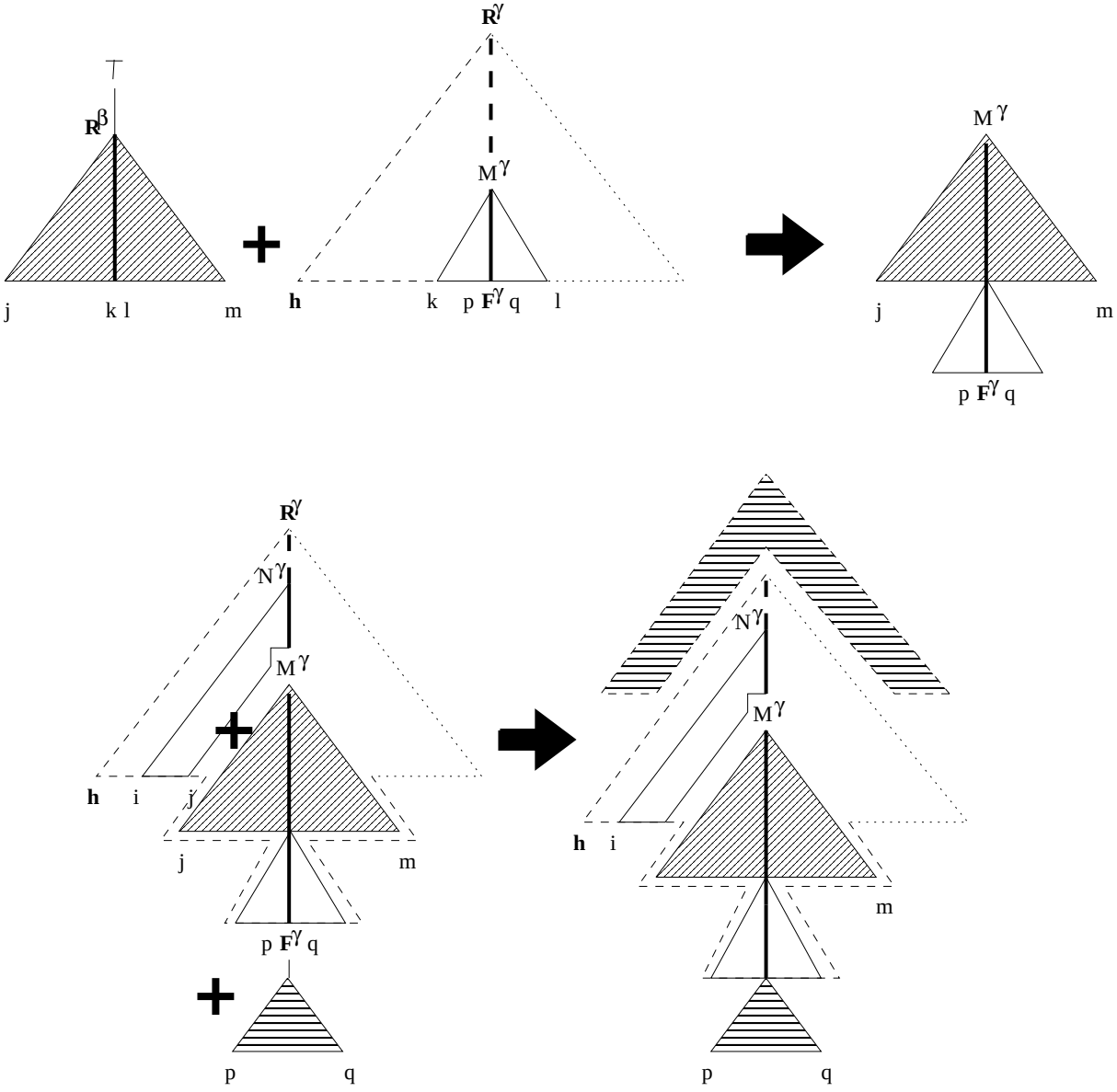

Figura 3.6: Descripción gráfica de la aplicación consecutiva de los pasos $\mathcal{D}^{\text{AdjComp}^0}_{\text{Nederhof}}$ y $\mathcal{D}^{\text{AdjComp1}}_{\text{Nederhof}}$

Para demostrar que el esquema de análisis **Nederhof** puede ser obtenido mediante un refinamiento de los pasos deductivos del esquema **Earley** debemos probar que para todo sistema de análisis $\mathbb{P}_{\text{Earley}}$ y $\mathbb{P}_{\text{Nederhof}}$ se cumple $\mathbb{P}_{\text{Earley}} \overset{\textbf{sr}}{\Longrightarrow} \mathbb{P}_{\text{Nederhof}}$. Ello conlleva demostrar que $\mathcal{I}_{\text{Earley}} \subseteq \mathcal{I}_{\text{Nederhof}}$ y que $\overset{*}{\vdash}_{\text{Earley}} \subseteq \overset{*}{\vdash}_{\text{Nederhof}}$. Lo primero se obtiene directamente puesto que $\mathcal{I}_{\text{Earley}} \subseteq \mathcal{I}_{\text{Nederhof}}$ por definición de los sistemas de análisis. Lo segundo se obtiene demostrando que $\mathcal{D}_{\text{Earley}} \subseteq \overset{*}{\vdash}_{\text{Nederhof}}$. Los únicos pasos deductivos de $\mathbb{P}_{\text{Earley}}$ que no se han incorporado directamente en $\mathbb{P}_{\text{Nederhof}}$ son $\mathcal{D}_{\text{Earley}}^{\text{AdjComp}^1}$ y $\mathcal{D}_{\text{Earley}}^{\text{AdjComp}^2}$ y para ellos se cumple que:

- Un paso $\mathcal{D}_{\text{Earley}}^{\text{AdjComp}^1}$ es equivalente a la aplicación de un paso $\mathcal{D}_{\text{Nederhof}}^{\text{Comp}^0}$ seguido de la aplicación de un paso $\mathcal{D}_{\text{Nederhof}}^{\text{Comp}^1}$:

$$\frac{[j, \top \rightarrow \mathbf{R}^{\beta}\bullet, j, m \mid k, l],\ [h, M^{\gamma} \rightarrow \upsilon\bullet, k, l \mid p, q],}{[[M^{\gamma} \rightarrow \upsilon\bullet, j, m \mid p, q]]}$$

$$\frac{[[M^{\gamma} \rightarrow \upsilon\bullet, j, m \mid p, q]],\ [h, \mathbf{F}^{\gamma} \rightarrow \bot\bullet, p, q \mid p, q],\ [h, N^{\gamma} \rightarrow \delta \bullet M^{\gamma}\nu, i, j \mid -, -]}{[h, N^{\gamma} \rightarrow \delta M^{\gamma} \bullet \nu, i, m \mid p, q]}$$

- Un paso $\mathcal{D}_{\text{Earley}}^{\text{AdjComp}^2}$ es equivalente a la aplicación de un paso $\mathcal{D}_{\text{Nederhof}}^{\text{Comp}^0}$ seguido de la aplicación de un paso $\mathcal{D}_{\text{Nederhof}}^{\text{Comp}^2}$:

$$\frac{[j, \top \rightarrow \mathbf{R}^{\beta}\bullet, j, m \mid k, l],\ [h, M^{\gamma} \rightarrow \upsilon\bullet, k, l \mid p, q],}{[[M^{\gamma} \rightarrow \upsilon\bullet, j, m \mid p, q]]}$$

$$\frac{[[M^{\gamma} \rightarrow \upsilon\bullet, j, m \mid p, q]],\ [h, N^{\gamma} \rightarrow \delta \bullet M^{\gamma}\nu, i, j \mid p', q']}{[h, N^{\gamma} \rightarrow \delta M^{\gamma} \bullet \nu, i, m \mid p', q']}$$

□

3.7. Análisis sintáctico de TAG lexicalizadas

Las gramáticas lexicalizadas poseen una propiedad muy interesante desde el punto de vista del análisis sintáctico: son finitamente ambiguas. Puesto que cada componente de la gramática (en el caso de TAG, cada árbol elemental) está asociado con un componente léxico, solamente un conjunto finito de tales estructuras pueden ser utilizadas para el análisis de una cadena de entrada dada y además solamente existe un número finito de combinaciones de dichas estructuras. En resumen, las gramáticas lexicalizadas impiden la aparición de análisis cíclicos.

El análisis de gramáticas lexicalizadas puede realizarse en dos fases, una primera en la cual se seleccionan todas las estructuras relevantes para la cadena de entrada que se pretende analizar y una segunda en la cual se aplica un algoritmo de análisis sintáctico que combine dichas estructuras. Este tipo de procesamiento se corresponde con un análisis fuera de línea. Las gramáticas lexicalizadas también pueden ser analizadas en línea, de tal modo que según se va avanzado en la lectura de la cadena de entrada se proporcionen las estructuras elementales correspondientes. Algunos autores [174, 168] sugieren que en análisis fuera de línea está mejor adaptado a este tipo de gramáticas puesto que las estructuras seleccionadas en la primera fase posibilitan al analizador sintáctico la utilización de información ascendente no local[7], restringiendo de este modo

[7]Esta información puede incluso no estar acotada con respecto a la distancia [168], de tal modo que no se puede imitar su efecto mediante la utilización de un número limitado de símbolos de preanálisis. Esta características se puede aplicar por ejemplo al reconocimiento de frases hechas con expresiones arbitrarias intercaladas.

las posibilidades de combinación de las estructuras e incluso el número de estructuras a considerar. En efecto, al actuar de este modo, el analizador sintáctico solamente considerará aquellas estructuras relevantes para la cadena a analizar, por lo que se podría decir que trabaja sobre una subgramática relevante para la cadena de entrada.

Los beneficios que se obtiene de la lexicalización dependen del algoritmo de análisis que se vaya a aplicar. Los algoritmos puramente ascendentes, del tipo CYK, únicamente se benefician de la reducción del número de estructuras a considerar durante el proceso de análisis. Un algoritmo puramente descendente, basado en una exploración en profundidad con retroceso, conseguiría mayores beneficios, puesto que al ser las gramáticas lexicalizadas finitamente ambiguas el espacio de búsqueda es finito y por lo tanto el análisis terminará en todos los casos[8]. Loa algoritmos mixtos que utilizan información ascendente y descendente, como por ejemplo los algoritmos de tipo Earley, se ven también beneficiados por la lexicalización. Una primera ventaja surge del hecho de que ningún árbol elemental tiene la cadena vacía por frontera, lo cual significa que una adjunción no puede ser predicha y completada sin avanzar en el reconocimiento de la cadena de entrada. Por tanto, al terminación está asegurada. Adicionalmente, la utilización de una estrategia de dos fases permite que la selección de estructuras de acuerdo a los componentes léxicos presentes en la cadena de entrada ayude al analizador sintáctico en la tarea de filtrar las predicciones y/o compleciones para la adjunción y la sustitución. Resultados experimentales realizados Schabes y Joshi [174, 168] muestran que la estrategia de dos fases aumenta considerablemente la eficiencia de un algoritmo de análisis sintáctico de tipo Earley para TAG.

Todos los algoritmos mostrados en este capítulo pueden ser fácilmente adaptados a gramáticas de adjunción de árboles lexicalizadas. Para ello sólo es preciso incluir un paso deductivo para tratar la sustitución de un árbol en un nodo de sustitución. Puesto que dicho operación es independiente del contexto, no afecta a la complejidad espacial ni temporal de los algoritmos.

3.8. El bosque de análisis

Los algoritmos mostrados hasta el momento, tal y como han sido descritos, son realmente reconocedores y no analizadores sintácticos, puesto que no construyen árboles de derivación. Sin embargo, cada uno de los pasos deductivos contiene la información necesaria para generar la parte correspondiente de un árbol de derivación y, puesto que todos los algoritmos recorren todas las posibles derivaciones, se pueden reconstruir todos los posibles árboles de derivación.

Puesto que estamos tratando con analizadores no deterministas se trata de construir una estructura, denominada *bosque de análisis* [30, 215] que permita representar todas las derivaciones de un forma compacta, compartiendo subderivaciones comunes, y que permita extraer cada una de las derivaciones en tiempo lineal con respecto al tamaño del bosque de análisis. El problema de la construcción del bosque de análisis para TAG ha sido estudiado con anterioridad por Vijay-Shanker y Weir en [215], que han propuesto dos posibles soluciones: la utilización de gramáticas independientes del contexto y la utilización de gramáticas lineales de índices. Cualquiera de las soluciones es aplicable a los algoritmos de análisis sintáctico mostrados en este capítulo.

3.8.1. Gramáticas independientes del contexto como bosque de análisis

Es posible representar el bosque compartido mediante una gramática independiente del contexto que capture la independencia al contexto de la operación de adjunción. Los no-terminales

[8] Un analizador sintáctico puramente descendente puede no terminar para una gramática no lexicalizada puesto que puede intentar repetir indefinidamente el análisis del mismo conjunto de estructuras sin avanzar en el reconocimiento de la cadena de entrada.

de la gramática serán de la forma $\langle tb, N^\gamma, i, j, p, q\rangle$ donde $tb \in \{\top, \bot\}$ se utiliza para indicar si el no-terminal representa al nodo N^γ antes ($\bot$) o después ($\top$) de una adjunción. Es interesante observar que los no-terminales son casi idénticos a los ítems utilizados en el esquema de análisis **CYK**. Mediante una pequeña modificación en los pasos deductivos[9] sería posible hacer $adj =$ true siempre que $tb = \top$ y que $adj =$ false siempre que $tb = \bot$. Puesto que los ítems de los restantes esquemas son un refinamiento de los ítems de **CYK** la información necesaria para los no-terminales se puede obtener directamente a partir de los ítems.

Respecto a la forma de las producciones, a modo de ejemplo, mostramos la producción correspondiente a la adjunción del árbol auxiliar β en el nodo N^γ:

$$\langle \top, N^\gamma, i, j, r, s\rangle \rightarrow \langle \top, \mathbf{R}^\beta, i, j, p, q\rangle \ \langle \bot, N^\gamma, p, q, r, s\rangle$$

la cual se corresponde con el paso deductivo de adjunción del esquema de análisis **CYK**. Al igual que ocurría con los no-terminales, las producciones del bosque de análisis se pueden obtener directamente a partir de los pasos deductivos en los diferentes esquemas de análisis.

El número de producciones es $\mathcal{O}(n^6)$ y la construcción de la gramática tienen una complejidad temporal $\mathcal{O}(n^6)$, por lo que la complejidad temporal de los algoritmos permanece inalterable, aunque la complejidad espacial aumenta de $\mathcal{O}(n^4)$ ó $\mathcal{O}(n^5)$ a $\mathcal{O}(n^6)$.

Un aspecto interesante a destacar es que aunque el bosque de análisis construido de esta forma codifica las derivaciones para una cadena de entrada dada, el lenguaje derivado por la gramática independiente del contexto no es importante. Lo que importa es que el lenguaje generado es no vacío si la cadena pertenece a la TAG original y en tal caso las derivaciones para la TAG original puede ser obtenidas en tiempo lineal a partir de las derivaciones de la gramática independiente del contexto que codifica el bosque compartido, siempre que esta haya sido podada para eliminar los símbolos inútiles.

3.8.2. Gramáticas lineales de índices como bosque de análisis

Se puede representar el bosque compartido mediante una gramática lineal de índices utilizando la transformación de TAG en LIG definida en [214]. A modo de ejemplo, las siguiente producciones representan la adjunción del árbol auxiliar β en el nodo N^γ:

$$\langle \top, i, j\rangle[\circ\circ N^\gamma] \rightarrow \langle \top, i, j\rangle[\circ\circ N^\gamma \mathbf{R}^\beta]$$

$$\langle \bot, p, q\rangle[\circ\circ N^\gamma \mathbf{F}^\beta] \rightarrow \langle \bot, p, q\rangle[\circ\circ N^\gamma]$$

La primera producción representa el final de la adjunción mientras que la segunda representa el reconocimiento del nodo pie del árbol auxiliar.

La información contenida en los no-terminales y producciones de la gramática lineal de índices puede obtenerse directamente a partir de los ítems y pasos deductivos de los esquemas de análisis sintáctico.

El tamaño de la gramática es $\mathcal{O}(n^3)$ y el tiempo necesario para construirla es $\mathcal{O}(n^6)$, por lo que las complejidades temporal y espacial de los algoritmos no se ven afectadas.

El inconveniente de esta representación estriba en que no se pueden extraer directamente los árboles de derivación individuales, sino que es preciso construir una estructura auxiliar que toma la forma de un autómata finito que reconoce las pilas de índices asociadas a cada terminal. Una vez construido dicho autómata finito, cada uno de los árboles de derivación puede ser extraído con una complejidad temporal que tiene por cota inferior $\mathcal{O}(n^4)$ y que en el caso de las gramáticas de adjunción de árboles lexicalizadas tiene como cota superior $\mathcal{O}(n^5)$. Puesto que el tamaño del

[9] Esencialmente la adición de un paso deductivo $\mathcal{D}_{\text{CYK}}^{\text{NoAdj}} = \frac{[N^\gamma, i, j | p, q | \text{false}]}{[N^\gamma, i, j | p, q | \text{true}]}$ es aplicable siempre que la realización de una adjunción sobre N^γ sea opcional.

bosque compartido es $\mathcal{O}(n^3)$, no se asegura que en todos los casos la recuperación de los árboles de derivación individuales se pueda realizar en tiempo lineal con respecto al tamaño del bosque compartido.

3.9. Otros algoritmos de análisis sintáctico para TAG

Presentamos a continuación un conjunto de algoritmos de análisis sintáctico de TAG que si bien no están en el camino principal de la evolución de los algoritmos de análisis mostrado anteriormente, presentan interés bien por constituir caminos laterales del camino principal de evolución, bien por haber constituido hitos importantes aunque posteriormente hayan quedado relegados, o bien por constituir ejemplos singulares de aplicación a TAG de ciertas áreas del análisis sintáctico, como la incrementalidad o el paralelismo.

3.9.1. El algoritmo de Lang

Lang describe en [106] un algoritmo tabular de análisis de TAG, con el objetivo principal de mostrar que técnicas de análisis muy generales pueden ser utilizadas para desarrollar un analizador sintáctico de tipo Earley para TAG. Concretamente, Lang utiliza las siguientes técnicas: gramáticas de cláusulas definidas (DCG) [143], autómatas lógicos a pila (LPDA) [56] y evaluación en programación dinámica de autómatas a pila. Efectivamente, su algoritmo se desglosa en tres fases principales:

1. Traducción de la gramática TAG a una gramática DCG.
2. Construcción de un LPDA ascendente predictivo.
3. Modificación de la técnica general de evaluación en programación dinámica de este LPDA con el fin de obtener un analizador que actúe de izquierda a derecha a pesar de la presencia de discontinuidades, modificación que se basa en propiedades generales de los LPDA.

Trataremos de describir un esquema de análisis **Lang** para el algoritmo de Lang, tarea que no es ciertamente fácil dada su complejidad: como muestra de ello, basta pensar en que su comportamiento fue descrito en [106] mediante 28 tipos de transiciones LPDA. En [23] se muestra una implementación del algoritmo.

El primer paso para la descripción de **Lang** consiste en describir el conjunto de ítems que serán utilizados. En este caso, dicho conjunto puede subdividirse en 6 subconjuntos diferentes, que denominaremos 1a, 1b, 1c, 2a, 2b y 2c. Los tres primeros estarán dedicados al reconocimiento de nodos que no forman parte de la espina de un árbol auxiliar, mientras que los tres restantes tratarán precisamente con dichos elementos. Definimos a continuación los ítems que forman cada uno de dichos conjuntos:

1a Los ítems del primer conjunto son de la forma

$$\left\{ [X,i,j] \mid \begin{array}{ll} i=j & \text{si } X=\overline{N^\gamma},\ N^\gamma \notin \text{espina}(\gamma) \\ N^\gamma \stackrel{*}{\Rightarrow} a_i \ldots a_{j-1} & \text{sii } X=\overline{\overline{N^\gamma}},\ N^\gamma \notin \text{espina}(\gamma) \end{array} \right\}$$

donde un $\overline{N^\gamma}$ denota que el nodo N^γ ha sido visitado en la fase descendente del algoritmo, mientras que $\overline{\overline{N^\gamma}}$ denota que el nodo N^γ ha sido visitado en la fase ascendente del algoritmo, esto es, que el subárbol que cuelga de dicho nodo ha sido completamente analizado. Este tipo de ítems se utiliza para representar la parte del análisis que es análoga al caso de gramáticas independientes del contexto, es decir, aquella que consiste simplemente en recorrer un árbol elemental en forma prefija sin realizar adjunciones.

1b El segundo conjunto de ítems se utiliza para determinar la parte que ha sido reconocida de cada producción que forma parte de un árbol elemental, en el caso de que no intervenga directamente en un operación de adjunción. Excluimos entonces de este tipo de ítems aquellos que se refieran a producciones que definen la espina de un árbol auxiliar, puesto que estas siempre están relacionadas con la resolución de una operación de adjunción ya que son las encargadas de propagar la información relativa al pie. Los ítems en este conjunto son de la forma

$$\left\{ \; [N^\gamma \rightarrow \delta \bullet \nu, i, j] \mid \quad N^\gamma \notin \text{espina}(\gamma),\ \delta \stackrel{*}{\Rightarrow} a_i \ldots a_{j-1} \; \right\}$$

1c El tercer conjunto de ítems es especial en el sentido de que se utiliza únicamente para representar un resultado intermedio cuando se ha terminado de recorrer un árbol auxiliar pero no se conoce todavía si el pie ha sido correctamente predicho. La forma de estos ítems es

$$\left\{ \begin{array}{lll} [N^\gamma \rightarrow \delta \bullet \nu, i, j \mid\mid p, q] \mid & N^\gamma \notin \text{espina}(\gamma) \text{ y además:} & \\ & \delta \stackrel{*}{\Rightarrow} a_i \ldots a_{p-1} \mathbf{F}^\beta a_q \ldots a_{j-1} & \text{sii } (p,q) \neq (-,-),\ \beta \in \text{adj}(N^\gamma) \\ & \delta \stackrel{*}{\Rightarrow} a_i \ldots a_{j-1} & \text{sii } (p,q) = (-,-) \end{array} \right\}$$

2a El cuarto conjunto de ítems se utiliza para propagar la información del pie a través de los nodos que forman la espina de un árbol auxiliar β. Los ítems son en este caso de la forma

$$\left\{ \begin{array}{lll} [X, i, j \mid p, q] \mid & i = j,\ p = q = - & \text{sii } X = \overline{N^\beta},\ N^\beta \in \text{espina}(\beta) \\ & N^\beta \stackrel{*}{\Rightarrow} a_i \ldots a_{p-1} \mathbf{F}^\beta a_q \ldots\ldots a_{j-1} & \text{sii } X = \overline{\overline{N^\beta}},\ N^\beta \in \text{espina}(\beta) \end{array} \right\}$$

donde $\overline{N^\beta}$ indica que el nodo N^β de la espina del árbol auxiliar β ha sido visitado en la fase descendente del algoritmo, mientras que $\overline{\overline{N^\beta}}$ indica que dicho nodo ha sido visitado en la fase ascendente del algoritmo y por tanto el subárbol que cuelga de él ha sido analizado, teniendo en cuenta que el pie puede haber una discontinuidad con respecto a la cadena de entrada.

2b Los ítems del quinto conjunto se utilizan para determinar que una parte de una producción en la espina de un árbol auxiliar ha sido reconocida y para propagar, en el caso de que el nodo pie ya haya sido predicho, la información correspondiente a la discontinuidad producida por dicho pie. Concretamente, los ítems son de la forma

$$\left\{ \begin{array}{lll} [N^\beta \rightarrow \delta \bullet \nu, i, j \mid p, q] \mid & \beta \in \boldsymbol{A},\ N^\beta \in \text{espina}(\beta) \text{ y además:} & \\ & \delta \stackrel{*}{\Rightarrow} a_i \ldots a_{p-1} \mathbf{F}^\beta a_q \ldots a_{j-1} & \text{sii } (p,q) \neq (-,-) \\ & \delta \stackrel{*}{\Rightarrow} a_i \ldots a_{j-1} & \text{sii } (p,q) = (-,-) \end{array} \right\}$$

2c El sexto conjunto de ítems es especial en el sentido de que se utiliza únicamente para representar un resultado intermedio cuando se ha terminado de recorrer un árbol auxiliar que ha sido adjuntado en la espina de otro árbol auxiliar, y cuando no se conoce todavía si el pie ha sido reconocido correctamente. Los ítems de este conjunto presentan la forma

$$\left\{ \begin{array}{lll} [N^{\beta_1} \rightarrow \delta \bullet \nu, i, j \mid -, - \mid p, q] \mid & N^{\beta_1} \in \text{espina}(\beta_1) \text{ y además:} & \\ & \delta \stackrel{*}{\Rightarrow} a_i \ldots a_{p-1} \mathbf{F}^{\beta_2} a_q \ldots a_{j-1} & \text{sii } (p,q) \neq (-,-), \\ & & \beta_2 \in \text{adj}(N^{\beta_1}) \\ & \delta \stackrel{*}{\Rightarrow} a_i \ldots a_{j-1} & \text{sii } (p,q) = (-,-) \end{array} \right\}$$

Si bien la utilización de tipos de ítems especializados puede ser beneficiosa en aras de una optimización del algoritmo, puesto que se evita el almacenamiento de elementos no relevantes en cada punto del proceso de análisis, tiene como inconveniente que hace necesario utilizar un elevado número de pasos deductivos, algunos de los cuales pueden considerarse redundantes, pues realizan la misma operación pero sobre diferentes tipos de ítems. Hemos tratado de reducir en lo posible el número de pasos deductivos, pero respetando lo más fielmente posible la filosofía original del algoritmo. Finalmente, el esquema de análisis **Lang** incluye 19 pasos deductivos, frente a las 28 transiciones LPDA del algoritmo original. La tabla 3.2 muestra la relación entre los pasos deductivos y las acciones del algoritmo tal y como son descritas por Lang [106]. A continuación mostramos el sistema de análisis correspondiente al esquema que estamos tratando.

Pasos deductivos	**Tarea realizada**
$\mathcal{D}^{\text{Init}}_{\text{Lang}}$	*inicialización*
$\mathcal{D}^{\text{Call}}_{\text{Lang}}$ $\mathcal{D}^{\text{Sel}}_{\text{Lang}}$ $\mathcal{D}^{\text{Tab}}_{\text{Lang}}$ $\mathcal{D}^{\text{Ret}}_{\text{Lang}}$	*expansión normal* de un nodo no hoja
$\mathcal{D}^{\text{Scan}}_{\text{Lang}}$	*reconocimiento de terminal*
$\mathcal{D}^{\text{AdjCall}}_{\text{Lang}}$ $\mathcal{D}^{\text{Adjret}}_{\text{Lang}}$ $\mathcal{D}^{\text{FootCall}}_{\text{Lang}}$ $\mathcal{D}^{\text{FootRet}}_{\text{Lang}}$	*decisión de adjunción normal* y *adjunción*
$\mathcal{D}^{\text{SpineCallNormal}}_{\text{Lang}}$ $\mathcal{D}^{\text{SpineRetNormal}}_{\text{Lang}}$ $\mathcal{D}^{\text{SpineCall}}_{\text{Lang}}$ $\mathcal{D}^{\text{SpineSel}}_{\text{Lang}}$ $\mathcal{D}^{\text{SpineTab}}_{\text{Lang}}$ $\mathcal{D}^{\text{SpineRet}}_{\text{Lang}}$	*expansión de la espina*
$\mathcal{D}^{\text{SpineAdjCall}}_{\text{Lang}}$ $\mathcal{D}^{\text{SpineAdjRet}}_{\text{Lang}}$	*decisión de adjunción en la espina* y *adjunción*
$\mathcal{D}^{\text{Foot}}_{\text{Lang}}$	*salto del pie*

Tabla 3.2: Actividades realizadas por los pasos deductivos del esquema **Lang**

Esquema de análisis sintáctico 3.8 El sistema de análisis $\mathbb{P}_{\text{Lang}}$ que se corresponde con el algoritmo de análisis sintáctico para TAG descrito por Lang, dada una gramática de adjunción

de árboles $\mathcal{T}$ y una cadena de entrada $a_1 \dots a_n$ se define como sigue:

$$\mathcal{I}_{\text{Lang}}^{(1a)} = \left\{ \; [X,i,j] \mid X \in \{\overline{N^\gamma}, \overline{\overline{N^\gamma}}\},\ \exists N^\gamma \to \delta \in \mathcal{P}(\gamma),\ \gamma \in \boldsymbol{I} \cup \boldsymbol{A},\ N^\gamma \notin \text{espina}(\gamma),\ 0 \le i \le j \; \right\}$$

$$\mathcal{I}_{\text{Lang}}^{(1b)} = \left\{ \; [N^\gamma \to \delta \bullet \nu, i, j] \mid N^\gamma \to \delta\nu \in \mathcal{P}(\gamma),\ \gamma \in \boldsymbol{I} \cup \boldsymbol{A},\ N^\gamma \notin \text{espina}(\gamma),\ 0 \le i \le j \; \right\}$$

$$\mathcal{I}_{\text{Lang}}^{(1c)} = \left\{ \begin{array}{ll} [N^\gamma \to \delta \bullet \nu, i, j \mid\mid p, q] \mid & N^\gamma \to \delta\nu \in \mathcal{P}(\gamma),\ \gamma \in \boldsymbol{I} \cup \boldsymbol{A},\ N^\beta \notin \text{espina}(\beta), \\ & 0 \le i \le p \le q \le j,\ (p,q) \le (i,j) \end{array} \right\}$$

$$\mathcal{I}_{\text{Lang}}^{(2a)} = \left\{ \begin{array}{ll} [X, i, j \mid p, q] \mid & X \in \{\overline{N^\beta}, \overline{\overline{N^\beta}}\},\ \exists N^\beta \to \delta \in \mathcal{P}(\beta),\ N^\beta \in \text{espina}(\beta), \\ & \beta \in \boldsymbol{A},\ 0 \le i \le p \le q \le j,\ (p,q) \le (i,j) \end{array} \right\}$$

$$\mathcal{I}_{\text{Lang}}^{(2b)} = \left\{ \begin{array}{ll} [N^\beta \to \delta \bullet \nu, i, j \mid p, q] \mid & N^\beta \to \delta\nu \in \mathcal{P}(\beta),\ \beta \in \boldsymbol{A},\ N^\beta \in \text{espina}(\beta), \\ & 0 \le i \le p \le q \le j,\ (p,q) \le (i,j) \end{array} \right\}$$

$$\mathcal{I}_{\text{Lang}}^{(2c)} = \left\{ \begin{array}{ll} [N^\beta \to \delta \bullet \nu, i, j \mid -, - \mid p, q] \mid & N^\beta \to \delta\nu \in \mathcal{P}(\beta),\ \beta \in \boldsymbol{A},\ N^\beta \in \text{espina}(\beta), \\ & 0 \le i \le p \le q \le j,\ (p,q) \le (i,j) \end{array} \right\}$$

$$\mathcal{I}_{\text{Lang}} = \mathcal{I}_{\text{Lang}}^{(1a)} \cup \mathcal{I}_{\text{Lang}}^{(1b)} \cup \mathcal{I}_{\text{Lang}}^{(1c)} \cup \mathcal{I}_{\text{Lang}}^{(2a)} \cup \mathcal{I}_{\text{Lang}}^{(2b)} \cup \mathcal{I}_{\text{Lang}}^{(2c)}$$

$$\mathcal{D}_{\text{Lang}}^{\text{Init}} = \frac{}{[\overline{\mathbf{R}^\alpha}, 0, 0]} \quad \alpha \in \boldsymbol{I}$$

$$\mathcal{D}_{\text{Lang}}^{\text{Call}} = \frac{[N^\gamma \to \delta \bullet M^\gamma \nu, i, j]}{[\overline{M^\gamma}, j, j]} \quad \textbf{nil} \in \text{adj}(M^\gamma)$$

$$\mathcal{D}_{\text{Lang}}^{\text{Sel}} = \frac{[\overline{M^\gamma}, j, j]}{[M^\gamma \to \bullet\delta, j, j]}$$

$$\mathcal{D}_{\text{Lang}}^{\text{Tab}} = \frac{[M^\gamma \to \bullet\delta, j, k],}{[\overline{\overline{M^\gamma}}, j, k]}$$

$$\mathcal{D}_{\text{Lang}}^{\text{Ret}} = \frac{[N^\gamma \to \delta \bullet M^\gamma \nu, i, j],\ [\overline{\overline{M^\gamma}}, j, k]}{[N^\gamma \to \delta M^\gamma \bullet \nu, i, k]} \quad \textbf{nil} \in \text{adj}(M^\gamma)$$

$$\mathcal{D}_{\text{Lang}}^{\text{Scan}} = \frac{[N^\gamma \to \delta \bullet a\nu, i, j-1],\ [a, j-1, j]}{[N^\gamma \to \delta a \bullet \nu, i, j]}$$

$$\mathcal{D}_{\text{Lang}}^{\text{AdjCall}} = \frac{[N^\gamma \to \delta \bullet M^\gamma \nu, i, j]}{[\overline{\mathbf{R}^\beta}, j, j \mid -, -]} \quad \beta \in \text{adj}(M^\gamma)$$

$$\mathcal{D}^{\text{AdjRet}}_{\text{Lang}} = \frac{[N^\gamma \to \delta \bullet M^\gamma \nu, i, j],\ [\overline{\overline{\mathbf{R}^\beta}}, j, k \mid p, q]}{[N^\gamma \to \delta \bullet M^\gamma \nu, i, k \mid\mid p, q]} \quad \beta \in \text{adj}(M^\gamma),\ M^\gamma \notin \text{espina}(\gamma)$$

$$\mathcal{D}^{\text{FootCall}}_{\text{Lang}} = \frac{[N^\gamma \to \delta \bullet M^\gamma \nu, i, k \mid\mid p, q],}{[\overline{M^\gamma}, p, p]} \quad M^\gamma \notin \text{espina}(\gamma)$$

$$\mathcal{D}^{\text{FootRet}}_{\text{Lang}} = \frac{[N^\gamma \to \delta \bullet M^\gamma \nu, i, k \mid\mid p, q],\ [\overline{\overline{M^\gamma}}, p, q]}{[N^\gamma \to \delta M^\gamma \bullet \nu, i, k]} \quad M^\gamma \notin \text{espina}(\gamma)$$

$$\mathcal{D}^{\text{SpineCallNormal}}_{\text{Lang}} = \frac{[N^\beta \to \delta \bullet M^\beta \nu, i, j \mid p, q]}{[\overline{M^\beta}, j, j]} \quad \begin{array}{l}\beta \in \boldsymbol{A},\ \textbf{nil} \in \text{adj}(M^\beta),\\ N^\beta \in \text{espina}(\beta),\ M^\beta \notin \text{espina}(\beta),\end{array}$$

$$\mathcal{D}^{\text{SpineRetNormal}}_{\text{Lang}} = \frac{[N^\beta \to \delta \bullet M^\beta \nu, i, j \mid p, q],\ [\overline{\overline{M^\beta}}, j, k]}{[N^\beta \to \delta M^\beta \bullet \nu, i, k \mid p, q]} \quad \begin{array}{l}\beta \in \boldsymbol{A},\ \textbf{nil} \in \text{adj}(M^\beta),\\ N^\beta \in \text{espina}(\beta),\ M^\beta \notin \text{espina}(\beta),\end{array}$$

$$\mathcal{D}^{\text{SpineCall}}_{\text{Lang}} = \frac{[N^\beta \to \delta \bullet M^\beta \nu, i, j \mid -, -]}{[\overline{M^\beta}, j, j \mid -, -]} \quad \beta \in \boldsymbol{A},\ \textbf{nil} \in \text{adj}(M^\beta),\ M^\beta \in \text{espina}(\beta)$$

$$\mathcal{D}^{\text{SpineSel}}_{\text{Lang}} = \frac{[\overline{M^\beta}, j, j \mid -, -]}{[M^\beta \to \bullet\delta, j, j \mid -, -]} \quad \beta \in \boldsymbol{A},\ M^\beta \in \text{espina}(\beta)$$

$$\mathcal{D}^{\text{SpineTab}}_{\text{Lang}} = \frac{[M^\beta \to \delta\bullet, j, k \mid p, q]}{[\overline{M^\beta}, j, k \mid p, q]} \quad \beta \in \boldsymbol{A},\ M^\beta \in \text{espina}(\beta)$$

$$\mathcal{D}^{\text{SpineRet}}_{\text{Lang}} = \frac{[N^\beta \to \delta \bullet M^\beta \nu, i, j \mid -, -],\ [\overline{M^\beta}, j, k \mid p, q]}{[N^\beta \to \delta M^\beta \bullet \nu, i, k \mid p, q]} \quad \beta \in \boldsymbol{A},\ \textbf{nil} \in \text{adj}(M^\beta),\ M^\beta \in \text{espina}(\beta)$$

$$\mathcal{D}^{\text{SpineAdjCall}}_{\text{Lang}} = \frac{[N^{\beta_1} \to \delta \bullet M^{\beta_2} \nu, i, j \mid -, -]}{[\overline{\mathbf{R}^{\beta_2}}, j, j \mid -, -]} \quad \beta_2 \in \text{adj}(M^{\beta_1}),\ M^{\beta_1} \in \text{espina}(\beta_1)$$

$$\mathcal{D}^{\text{SpineAdjRet}}_{\text{Lang}} = \frac{[N^{\beta_1} \to \delta \bullet M^{\beta_1} \nu, i, j \mid -, -],\ [\overline{\overline{\mathbf{R}^{\beta_2}}}, j, k \mid p, q]}{[N^{\beta_1} \to \delta \bullet M^{\beta_1} \nu, i, k \mid -, - \mid p, q]} \quad \beta_2 \in \text{adj}(M^{\beta_1}),\ M^{\beta_1} \in \text{espina}(\beta_1)$$

$$\mathcal{D}^{\text{SpineFootCall}}_{\text{Lang}} = \frac{[N^\beta \to \delta \bullet M^\beta \nu, i, k \mid -, - \mid p, q],}{[\overline{M^\beta}, p, p \mid -, -]} \quad M^\beta \in \text{espina}(\beta)$$

$$\mathcal{D}^{\text{SpineFootRet}}_{\text{Lang}} = \frac{[N^\beta \to \delta \bullet M^\beta \nu, i, k \mid -, - \mid p, q],\ [\overline{\overline{M^\beta}}, p, q \mid p', q']}{[N^\beta \to \delta M^\beta \bullet \nu, i, k \mid p', q']} \quad M^\beta \in \text{espina}(\beta)$$

$$\mathcal{D}^{\text{Foot}}_{\text{Lang}} = \frac{[\overline{\mathbf{F}^{\beta}}, j, j \mid -, -]}{[\overline{\overline{\mathbf{F}^{\beta}}}, j, k \mid j, k]}$$

$$\begin{aligned}\mathcal{D}_{\text{Lang}} = \;& \mathcal{D}^{\text{Init}}_{\text{Lang}} \cup \mathcal{D}^{\text{Call}}_{\text{Lang}} \cup \mathcal{D}^{\text{Sel}}_{\text{Lang}} \cup \mathcal{D}^{\text{Tab}}_{\text{Lang}} \cup \mathcal{D}^{\text{Ret}}_{\text{Lang}} \cup \mathcal{D}^{\text{Scan}}_{\text{Lang}} \cup \\ & \mathcal{D}^{\text{AdjCall}}_{\text{Lang}} \cup \mathcal{D}^{\text{AdjRet}}_{\text{Lang}} \cup \mathcal{D}^{\text{FootCall}}_{\text{Lang}} \cup \mathcal{D}^{\text{FootRet}}_{\text{Lang}} \cup \\ & \mathcal{D}^{\text{SpineCallNormal}}_{\text{Lang}} \cup \mathcal{D}^{\text{SpineRetNormal}}_{\text{Lang}} \cup \mathcal{D}^{\text{SpineCall}}_{\text{Lang}} \cup \mathcal{D}^{\text{SpineSel}}_{\text{Lang}} \cup \mathcal{D}^{\text{SpineTab}}_{\text{Lang}} \cup \mathcal{D}^{\text{SpineRet}}_{\text{Lang}} \cup \\ & \mathcal{D}^{\text{SpineAdjCall}}_{\text{Lang}} \cup \mathcal{D}^{\text{SpineAdjRet}}_{\text{Lang}} \cup \mathcal{D}^{\text{SpineFootCall}}_{\text{Lang}} \cup \mathcal{D}^{\text{SpineFootRet}}_{\text{Lang}} \cup \mathcal{D}^{\text{Foot}}_{\text{Lang}}\end{aligned}$$

$$\mathcal{F}_{\text{Lang}} = \left\{ \; [\overline{\overline{\mathbf{R}^{\alpha}}}, 0, n] \mid \alpha \in \boldsymbol{I} \; \right\}$$

§

En interesante observar que ningún paso deductivo tiene más de dos antecedentes, lo cual no resulta sorprendente si tenemos en cuenta que el algoritmo fue originalmente descrito para LPDA, un formalismo cuyas transiciones tienen a lo sumo dos antecedentes. La necesidad de mantener el límite de dos antecedentes probablemente haya sido una de las causas que motivaron la necesidad de definir un número tan elevado de ítems para este esquema de análisis.

Con respecto a la forma de proceder del algoritmo descrito por el esquema de análisis **Lang**, diremos que este comienza el procesamiento de una cadena de entrada prediciendo el nodo raíz de un árbol inicial mediante el paso deductivo $\mathcal{D}^{\text{Init}}_{\text{Lang}}$. Posteriormente, siempre que puede trata de proceder como si las producciones que conforman los árboles fuesen totalmente independientes del contexto: los pasos $\mathcal{D}^{\text{Call}}_{\text{Lang}}$ predicen un nodo, los pasos $\mathcal{D}^{\text{Sel}}_{\text{Lang}}$ seleccionan la producción que tienen a dicho nodo como padre (no terminal del lado izquierdo), los pasos $\mathcal{D}^{\text{Ret}}_{\text{Lang}}$ permiten ir avanzando en el reconocimiento de la parte derecha de dicha producción y los pasos $\mathcal{D}^{\text{Tab}}_{\text{Lang}}$ indican que la producción ha sido totalmente analizada y por consiguiente el nodo padre. Los pasos $\mathcal{D}^{\text{Scan}}_{\text{Lang}}$ se utilizan para reconocer los nodos etiquetados por símbolos terminales.

Las adjunciones en los nodos que no forman parte de una espina se predicen mediante los pasos $\mathcal{D}^{\text{AdjCall}}_{\text{Lang}}$. Una vez terminado el reconocimiento del árbol auxiliar, se genera un ítem intermedio mediante un paso $\mathcal{D}^{\text{AdjRet}}_{\text{Lang}}$ a partir del cual se trata de reconocer, mediante un paso $\mathcal{D}^{\text{FootPred}}_{\text{Lang}}$, si el subárbol que cuelga del nodo en el que se realizó la adjunción reconoce una parte de la cadena de entrada que coincide con la discontinuidad indicada por el pie del árbol auxiliar. Si es así, la adjunción se terminará aplicando un paso $\mathcal{D}^{\text{FootRet}}_{\text{Lang}}$.

Las producciones que forman parte de la espina de un árbol auxiliar reciben un tratamiento especial, pues existe un conjunto de pasos deductivos que realizan un tratamiento análogo al descrito en los párrafos anteriores, pero sólo para este tipo de producciones:

- El paso deductivo $\mathcal{D}^{\text{SpineCallNormal}}_{\text{Lang}}$ se encarga de predecir aquellos no terminales que se encuentran a derecha o izquierda del nodo de la producción que forma parte de la espina, mientras que el paso $\mathcal{D}^{\text{SpineCallNormal}}_{\text{Lang}}$ avanza en el reconocimiento de la producción cuando ya se ha terminado el análisis de uno de dichos nodos.
- Los pasos deductivos $\mathcal{D}^{\text{SpineCall}}_{\text{Lang}}$, $\mathcal{D}^{\text{SpineSel}}_{\text{Lang}}$, $\mathcal{D}^{\text{SpineTab}}_{\text{Lang}}$ y $\mathcal{D}^{\text{SpineRet}}_{\text{Lang}}$ se encargan de expandir el nodo que están en la espina, propagando la información relativa a la discontinuidad producida por el pie.

- Los pasos $\mathcal{D}_{\text{Lang}}^{\text{SpineAdjCall}}$ y $\mathcal{D}_{\text{Lang}}^{\text{SpineAdjRet}}$ se encargan de realizar la operación de adjunción, generando un ítem intermedio a partir del cual se puede aplicar un paso deductivo $\mathcal{D}_{\text{Lang}}^{\text{SpineFootCall}}$ para comenzar el reconocimiento del subárbol correspondiente al pie, proceso que finalizará mediante la aplicación de un paso $\mathcal{D}_{\text{Lang}}^{\text{SpineFootRet}}$.

Por la forma de reconocer el pie en el paso deductivo $\mathcal{D}_{\text{Lang}}^{\text{Foot}}$ este algoritmo es semejante a las versiones del algoritmo de Earley ascendente para TAG, puesto que se predicen todas las posibles posiciones del pie y posteriormente, en la finalización de la adjunción, mediante los pasos deductivos $\mathcal{D}_{\text{Lang}}^{\text{AdjRet}}$, $\mathcal{D}_{\text{Lang}}^{\text{FootCall}}$, $\mathcal{D}_{\text{Lang}}^{\text{FootRet}}$, $\mathcal{D}_{\text{Lang}}^{\text{SpineAdjRet}}$, $\mathcal{D}_{\text{Lang}}^{\text{SpineFootCall}}$ y $\mathcal{D}_{\text{Lang}}^{\text{SpineFootRet}}$, se comprueba qué discontinuidad del pie es compatible con la cadena de entrada analizada.

Sin embargo, el algoritmo también comparte ciertas características de los algoritmos de tipo Earley que no preservan la propiedad del prefijo válido, puesto que realiza un recorrido en orden prefijo de los árboles elementales. Sin embargo dicho orden es alterado por las operaciones de adjunción.

Se podría decir entonces que el algoritmo de Lang está a medio camino entre un Earley ascendente y un Earley sin propiedad del prefijo válido. Ciertamente, fortaleciendo la predicción del pie y utilizando un sólo tipo de ítem obtendríamos un algoritmo similar al descrito por el esquema de análisis sintáctico **E**.

El esquema de análisis sintáctico **Lang** es susceptible de ser simplificado si observamos que ciertos pasos se aplican en cadena. Por ejemplo, los pasos $\mathcal{D}_{\text{Lang}}^{\text{Call}}$ generan ítems que sólo pueden ser utilizados como antecedentes de los pasos $\mathcal{D}_{\text{Lang}}^{\text{Sel}}$ y por tanto dichos pasos pueden ser contraídos para formar un nuevo conjunto de pasos $\mathcal{D}_{\text{Lang}}^{\text{Pred}}$ análogo al conjunto de pasos predictivos del algoritmo de Earley. Análogamente, los pasos $\mathcal{D}_{\text{Lang}}^{\text{Tab}}$ y $\mathcal{D}_{\text{Lang}}^{\text{Ret}}$ pueden contraerse para formar un conjunto de pasos $\mathcal{D}_{\text{Lang}}^{\text{Comp}}$ similar al conjunto de pasos de compleción del algoritmo de Earley. El mismo procedimiento puede ser aplicado a los pasos deductivos que se encargan de realizar la expansión de la espina.

De forma similar, los pasos $\mathcal{D}_{\text{Lang}}^{\text{AdjComp}}$ generan ítems que sólo pueden ser utilizados en los pasos $\mathcal{D}_{\text{Lang}}^{\text{FootPred}}$ y $\mathcal{D}_{\text{Lang}}^{\text{FootComp}}$. Podemos suprimir $\mathcal{D}_{\text{Lang}}^{\text{AdjComp}}$ si en los pasos que tratan el pie añadimos como antecedentes los antecedentes de dicho paso. El mismo procedimiento puede ser aplicado a los pasos deductivos que se encargan de realizar el reconocimiento del pie cuando el nodo de adjunción forma parte de la espina.

También se puede simplificar el conjunto de ítems si tenemos en cuenta que $\mathcal{A}^{\gamma}$ es equivalente a $N^{\gamma} \rightarrow \bullet\delta$ y que $\overline{\overline{N^{\gamma}}}$ es equivalente a $N^{\gamma} \rightarrow \delta\bullet$. Como consecuencia, el conjunto 1a de ítems se puede considerar incluido en el conjunto 1b, mientras que el 2a se encuentra incluido en el 2b. Por tanto, aplicando un filtrado estático al esquema de análisis **Lang** podemos eliminar dichos conjuntos de ítems redundantes, modificando adecuadamente los ítems de los pasos deductivos que se vean afectados.

A continuación definimos el sistema de análisis correspondiente al esquema de análisis **Lang′** resultado de aplicar la contracción de pasos deductivos y el filtrado estático mencionados anteriormente.

Esquema de análisis sintáctico 3.9 El sistema de análisis $\mathbb{P}_{\text{Lang}'}$, dada una gramática de adjunción de árboles $\mathcal{T}$ y una cadena de entrada $a_1 \ldots a_n$ se define como sigue:

$$\mathcal{I}_{\text{Lang}'} = \mathcal{I}_{\text{Lang}}^{(1\text{b})} \cup \mathcal{I}_{\text{Lang}}^{(2\text{b})}$$

$$\mathcal{D}_{\text{Lang}'}^{\text{Init}} = \frac{}{[\top \rightarrow \bullet\mathbf{R}^{\alpha}, 0, 0]} \quad \alpha \in \boldsymbol{I}$$

$$\mathcal{D}^{\text{Pred}}_{\text{Lang}'} = \frac{[N^\gamma \to \delta \bullet M^\gamma \nu, i, j]}{[M^\gamma \to \bullet\delta, j, j]} \quad \textbf{nil} \in \text{adj}(M^\gamma)$$

$$\mathcal{D}^{\text{Comp}}_{\text{Lang}'} = \frac{[N^\gamma \to \delta \bullet M^\gamma \nu, i, j],\ [M^\gamma \to \bullet\delta, j, k]}{[N^\gamma \to \delta M^\gamma \bullet \nu, i, k]} \quad \textbf{nil} \in \text{adj}(M^\gamma)$$

$$\mathcal{D}^{\text{Scan}}_{\text{Lang}'} = \mathcal{D}^{\text{Scan}}_{\text{Lang}}$$

$$\mathcal{D}^{\text{AdjPred}}_{\text{Lang}'} = \frac{[N^\gamma \to \delta \bullet M^\gamma \nu, i, j]}{[\top \to \bullet \mathbf{R}^\beta, j, j \mid -, -]} \quad \beta \in \text{adj}(M^\gamma)$$

$$\mathcal{D}^{\text{FootPred}}_{\text{Lang}'} = \frac{\begin{array}{l}[N^\gamma \to \delta \bullet M^\gamma \nu, i, j],\\ [\top \to \mathbf{R}^\beta \bullet, j, k \mid p, q],\end{array}}{[M^\gamma \to \bullet \upsilon, p, p]} \quad \beta \in \text{adj}(M^\gamma),\ M^\gamma \notin \text{espina}(\gamma)$$

$$\mathcal{D}^{\text{FootComp}}_{\text{Lang}'} = \frac{\begin{array}{l}[N^\gamma \to \delta \bullet M^\gamma \nu, i, j],\\ [\top \to \mathbf{R}^\beta \bullet, j, k \mid p, q],\\ [M^\gamma \to \upsilon \bullet, p, q]\end{array}}{[N^\gamma \to \delta M^\gamma \bullet \nu, i, k]} \quad \beta \in \text{adj}(M^\gamma),\ M^\gamma \notin \text{espina}(\gamma)$$

$$\mathcal{D}^{\text{SpinePredNormal}}_{\text{Lang}'} = \frac{[N^\beta \to \delta \bullet M^\beta \nu, i, j \mid p, q]}{[M^\beta \to \bullet \upsilon, j, j]} \quad \begin{array}{l}\beta \in \boldsymbol{A},\ \textbf{nil} \in \text{adj}(M^\beta),\\ N^\beta \in \text{espina}(\beta),\ M^\beta \notin \text{espina}(\beta)\end{array}$$

$$\mathcal{D}^{\text{SpineCompNormal}}_{\text{Lang}'} = \frac{\begin{array}{l}[N^\beta \to \delta \bullet M^\beta \nu, i, j \mid p, q],\\ [M^\beta \to \upsilon \bullet, j, k]\end{array}}{[N^\beta \to \delta M^\beta \bullet \nu, i, k \mid p, q]} \quad \begin{array}{l}\beta \in \boldsymbol{A},\ \textbf{nil} \in \text{adj}(M^\beta),\\ N^\beta \in \text{espina}(\beta),\ M^\beta \notin \text{espina}(\beta)\end{array}$$

$$\mathcal{D}^{\text{SpinePred}}_{\text{Lang}'} = \frac{[N^\beta \to \delta \bullet M^\beta \nu, i, j \mid -, -]}{[M^\beta \to \bullet\delta, j, j \mid -, -]} \quad \beta \in \boldsymbol{A},\ \textbf{nil} \in \text{adj}(M^\beta),\ M^\beta \in \text{espina}(\beta)$$

$$\mathcal{D}^{\text{SpineComp}}_{\text{Lang}'} = \frac{\begin{array}{l}[N^\beta \to \delta \bullet M^\beta \nu, i, j \mid -, -],\\ [M^\beta \to \delta \bullet, j, k \mid p, q]\end{array}}{[N^\beta \to \delta M^\beta \bullet \nu, i, k \mid p, q]} \quad \beta \in \boldsymbol{A},\ \textbf{nil} \in \text{adj}(M^\beta),\ M^\beta \in \text{espina}(\beta)$$

$$\mathcal{D}^{\text{SpineAdjPred}}_{\text{Lang}'} = \frac{[N^{\beta_1} \to \delta \bullet M^{\beta_2} \nu, i, j \mid -, -]}{[\top \to \bullet \mathbf{R}^{\beta_2}, j, j \mid -, -]} \quad \beta_2 \in \text{adj}(M^{\beta_1}),\ M^{\beta_1} \in \text{espina}(\beta_1)$$

$$\mathcal{D}^{\text{SpineFootPred}}_{\text{Lang}'} = \frac{\begin{array}{l}[N^{\beta_1} \to \delta \bullet M^{\beta_1} \nu, i, j \mid -, -],\\ [\top \to \mathbf{R}^{\beta_2} \bullet, j, k \mid p, q]\end{array}}{[M^{\beta_1} \to \bullet \upsilon, p, p \mid -, -]} \quad \beta_2 \in \text{adj}(M^{\beta_1}),\ M^{\beta_1} \in \text{espina}(\beta_1)$$

$$\mathcal{D}^{\text{SpineFootComp}}_{\text{Lang}'} = \frac{\begin{array}{l}[N^{\beta_1} \rightarrow \delta \bullet M^{\beta_1}\nu, i, j \mid -, -],\\ [\top \rightarrow \mathbf{R}^{\beta_2}\bullet, j, k \mid p, q],\\ [M^{\beta_1} \rightarrow \upsilon\bullet, p, q \mid p', q']\end{array}}{[N^{\beta_1} \rightarrow \delta M^{\beta_1} \bullet \nu, i, k \mid p', q']} \quad \beta_2 \in \text{adj}(M^{\beta_1}),\ M^{\beta_1} \in \text{espina}(\beta_1)$$

$$\mathcal{D}^{\text{Foot}}_{\text{Lang}'} = \frac{[\mathbf{F}^{\beta} \rightarrow \bullet\bot, j, j \mid -, -]}{[\mathbf{F}^{\beta} \rightarrow \bot\bullet, j, k \mid j, k]}$$

$$\begin{aligned}\mathcal{D}_{\text{Lang}'} = \quad & \mathcal{D}^{\text{Init}}_{\text{Lang}'} \cup \mathcal{D}^{\text{Pred}}_{\text{Lang}'} \cup \mathcal{D}^{\text{Comp}}_{\text{Lang}'} \cup \mathcal{D}^{\text{Scan}}_{\text{Lang}'} \cup\\ & \mathcal{D}^{\text{AdjPred}}_{\text{Lang}'} \cup \mathcal{D}^{\text{FootPred}}_{\text{Lang}'} \cup \mathcal{D}^{\text{FootComp}}_{\text{Lang}'} \cup\\ & \mathcal{D}^{\text{SpinePredNormal}}_{\text{Lang}'} \cup \mathcal{D}^{\text{SpineCompNormal}}_{\text{Lang}'} \cup \mathcal{D}^{\text{SpinePred}}_{\text{Lang}'} \cup \mathcal{D}^{\text{SpineComp}}_{\text{Lang}'} \cup\\ & \mathcal{D}^{\text{SpineAdjPred}}_{\text{Lang}'} \cup \mathcal{D}^{\text{SpineFootPred}}_{\text{Lang}'} \cup \mathcal{D}^{\text{SpineFootComp}}_{\text{Lang}'} \cup \mathcal{D}^{\text{Foot}}_{\text{Lang}'}\end{aligned}$$

$$\mathcal{F}_{\text{Lang}'} = \left\{ \ [\top \rightarrow \mathbf{R}^{\alpha}\bullet, 0, n] \mid \alpha \in \boldsymbol{I} \ \right\}$$

§

Proposición 3.7 $\mathbf{Lang} \overset{\mathbf{sc}}{\Longrightarrow} \overset{\mathbf{sf}}{\Longrightarrow} \mathbf{Lang}'$.

El esquema de análisis **Lang′** es muy similar al esquema de análisis **E**, radicando las diferencias en los dos puntos siguientes:

- La utilización de diferentes tipos de ítems por parte de **Lang′**. Efectivamente, podemos considerar que los ítems del tipo 1b son casos particulares del tipo 2b en los que los últimos índices toman el valor $-$, reduciendo así los diferentes tipos de ítems a 2b y 2c. Con ello se lograría también contraer numerosos pasos de análisis, pues no sería necesario distinguir entre operaciones realizadas sobre producciones que forman la espina y operaciones realizadas sobre otro tipo de producciones.

- El reconocimiento del subárbol que cuelga del pie en un árbol de derivación se realiza de forma muy particular en **Lang′**, prediciendo todas las posibles posiciones del nodo pie y comprobando al finalizar la adjunción si alguna de dichas predicciones es compatible con la cadena de entrada. En la figura 3.7 se muestra una representación gráfica de la predicción del pie en el caso de que el nodo de adjunción forme parte de la espina, por ser este el caso más complejo. En la figura 3.8 se muestra una representación gráfica de la finalización del reconocimiento del pie.

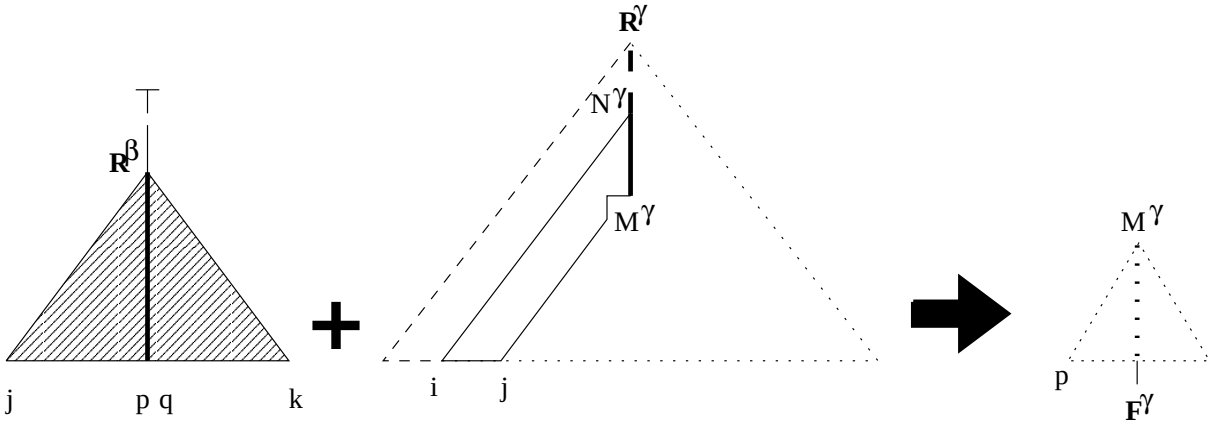

Figura 3.7: Descripción gráfica de la aplicación de un paso $\mathcal{D}^{\text{SpineFootPred}}_{\text{Lang}}$

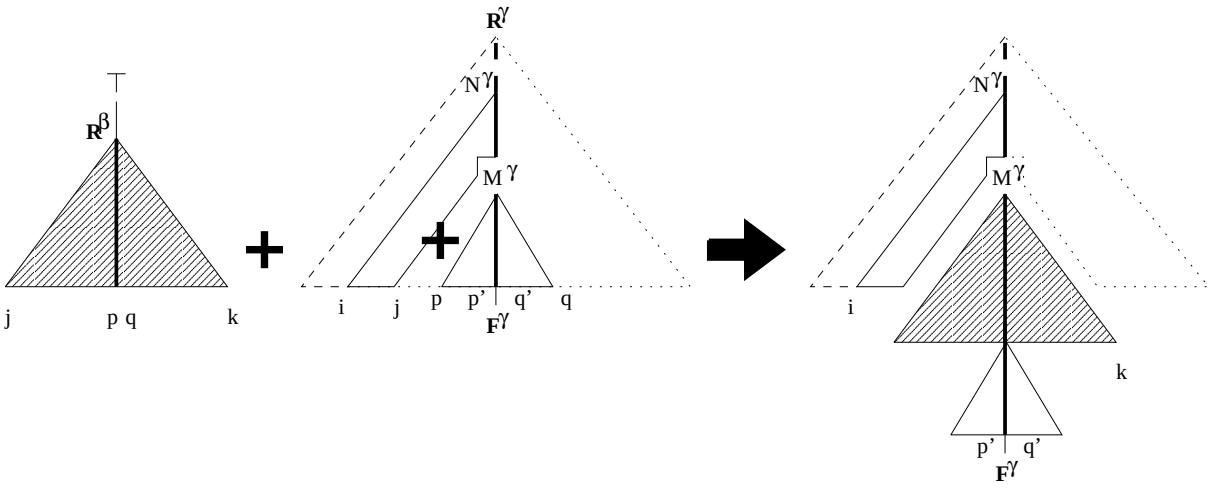

Figura 3.8: Descripción gráfica de la aplicación de un paso $\mathcal{D}^{\text{SpineFootComp}}_{\text{Lang}}$

3.9.2. Algoritmos bidireccionales

Lavelli y Satta presentan en [109] un algoritmo de tipo *head-corner* para el análisis de gramáticas de adjunción lexicalizadas que toma como *núcleo* o *head* el ancla léxica de cada árbol elemental. Dicho algoritmo puede considerarse como una extensión bidireccional de un algoritmo de tipo Earley para TAG en el cual el paso inicial reconoce las anclas léxicas y posteriormente trata de expandir el árbol elemental tanto a izquierda como a derecha[10]. Durante estas expansiones el algoritmo realiza predicciones descendentes con el fin de reconocer los nodos que no se encuentran en el camino de la raíz al ancla del árbol que se está analizando. La motivación lingüística de este algoritmo radica en que el ancla de cada árbol es el nodo del árbol que aporta mayor información acerca de la estructura sintáctica que dicho árbol representa y por tanto parece conveniente comenzar el análisis por dichos nodos.

Van Noord presenta en [205] otro algoritmo *head-corner* para gramáticas de adjunción de árboles lexicalizadas[11]. La singularidad con respecto al de Lavelli y Satta es que este algoritmo considera los nodos pie como núcleo de los árboles auxiliares[12]. Esta elección de los núcleos viene determinada por el funcionamiento del algoritmo, que se define como dirigido por el núcleo cuando se trata de analizar árboles iniciales y dirigido por el pie cuando se trata de analizar árboles auxiliares. A diferencia del algoritmo de Lavelli y Satta, el algoritmo de van Noord procede de modo totalmente ascendente, sin realizar predicciones. Otra diferencia importante es que el reconocimiento de los árboles auxiliares no se inicia por el reconocimiento de su ancla, sino bajo la demanda de una operación de adjunción, una consecuencia de la consideración de los nodos pie como núcleo de los árboles auxiliares. Sarkar presenta en [165] los resultados obtenidos en diversos experimentos realizados con el analizador de Van Noord.

Díaz Madrigal et al. presentan en [59, 62] un algoritmo bidireccional ascendente para el análisis de gramáticas de adjunción de árboles lexicalizadas inspirado en el algoritmo de De Vreugh y Honig para gramáticas independientes del contexto [57, 189]. El algoritmo resultante es similar al propuesto por Van Noord, con la importante salvedad de que todo nodo de un árbol elemental etiquetado por un terminal, así como el nodo pie de los árboles auxiliares, es considerado núcleo. Los mismos autores presentan en [61, 58] diversas optimizaciones, relativas al tratamiento de los nodos de sustitución y de los nodos etiquetados por ϵ, que mejoran el comportamiento práctico del algoritmo. El algoritmo resultante es comparado en [60] con otros algoritmos tabulares para el análisis de TAG.

Evans y Weir proponen en [73] un algoritmo bidireccional para el análisis sintáctico de gramáticas de adjunción lexicalizadas en el cual el recorrido de un árbol elemental comienza por el ancla y asciende nivel a nivel examinando primero los nodos a la izquierda del camino desde el ancla a la raíz y después los nodos a la derecha de dicho camino. La principal aportación de este algoritmo consiste en la codificación en forma de autómata finito del recorrido de los árboles elementales. Posteriormente dicho autómata finito es utilizado para construir un conjunto de ítems mediante un algoritmo que puede considerarse una versión bidireccional del algoritmo descrito por el esquema de análisis sintáctico $\mathbf{buE_1}$.

Lopez presenta en [114, 112] un algoritmo bidireccional ascendente para gramáticas de adjunción de árboles lexicalizadas. La principal innovación del algoritmo consiste en la definición de los *caminos conectados*. Dado un árbol elemental, cada uno de los caminos conectados representa una parte de dicho árbol que puede ser recorrida propagando una posición determinada de la cadena de entrada. Por tanto, un camino conectado comienza en un nodo raíz, un nodo de

[10] Una consecuencia de la bidireccionalidad del análisis es que los ítems deberán contener producciones con dos puntos en lugar de producciones con punto simples.

[11] Van Noord describe en [204] una versión anterior del mismo algoritmo que involucra una transformación de TAG a HPSG.

[12] El núcleo de los árboles iniciales sigue siendo el ancla.

sustitución, un nodo ancla o un nodo pie y termina en el siguiente nodo raíz, nodo de sustitución, nodo ancla o nodo pie. El algoritmo maneja ítems de la forma $[i, j, \Gamma_L, \Gamma_R, p, q, star]$, donde i y j representan las posiciciones de un segmento de la cadena de entrada, Γ_L y Γ_R son caminos conectados, p y q son las posiciones de la cadena de entrada correspondientes al pie y *star* es la dirección del nodo de adjunción predicho más cercano. Los ítems se combinan mediante pasos deductivos similares a los utilizados en el esquema Earley ascendente $\mathbf{buE_1}$ con la salvedad de que los caminos conectados reducen el número de ítems que es necesario generar para recorrer un árbol. La complejidad temporal permanece en $\mathcal{O}(n^6)$, donde n es la longitud de la cadena de entrada, pero se reduce el factor de la constante asociada al tamaño de la gramática. El algoritmo ha sido adaptado al tratamiento de frases orales espontáneas mediante la incorporación de pasos deductivos que tratan diversos fenómenos que aparecen frecuentemente en ese contexto, como son la duda, la elipsis y la autoreparación [113, 115, 112].

Halber presenta en [78] un algoritmo bidireccional basado en *chart* para el análisis de gramáticas de adjunción de árboles lexicalizadas. Este algoritmo puede verse como una extensión bidireccional del algoritmo descrito por el esquema de análisis **E** con el añadido de los *enlaces gramaticales* que son almacenados en cada ítem y que son utilizados para el cálculo de diferentes tipos de probabilidades sobre los árboles generados durante el proceso de análisis.

3.9.3. Algoritmos de varias fases

Existe un conjunto de algoritmos de análisis sintáctico de gramáticas de adjunción que proceden en varias fases, la primera de las cuales suele ser un análisis del esqueleto independiente del contexto y las siguientes dedicadas a la extracción de los árboles de derivación de TAG de entre la salida de la primera fase.

Harbusch presenta en [80] un algoritmo para el análisis sintáctico de TAG que pretende trabajar con una complejidad en el peor caso de $\mathcal{O}(n^4 \log n)$, resultado cuestionado posteriormente [172, 148]. El algoritmo de Harbusch trabaja con gramáticas de adjunción en las que cada nodo no terminal tiene a lo sumo dos descendientes y en las que no se permite que ningún nodo esté etiquetado por ϵ, a excepción de un único nodo de un árbol inicial especialmente concebido para la derivación de la cadena vacía. Además, cada árbol elemental debe producir al menos un terminal, excepto el árbol inicial especial para ϵ. El proceso de análisis se divide en las siete fases siguientes:

1. Paso especial para el tratamiento de la cadena vacía.

2. Definición de números de nodo que identifiquen unívocamente cada uno de los nodos de los árboles elementales de la gramática.

3. Obtención del esqueleto independiente de contexto de la gramática de adjunción que se pretendiese analizar.

4. Aplicación el algoritmo CYK para gramáticas independientes del contexto sobre el esqueleto independiente del contexto. Además de las acciones habituales realizadas en un algoritmo CYK, también se realizan ciertos cálculos relativos a la posición del pie en los árboles auxiliares.

5. Obtención de *números de nodo extendidos* para todos los ítems resultantes de la aplicación del algoritmo CYK, con los que se pretende representar la información concerniente a los árboles auxiliares necesaria en la resolución de las operaciones de adjunción, típicamente información relativa a los nodos pie.

6. Poda de aquellos árboles que no sean válidos por no respetar las restricciones propias de la formación de árboles en las gramáticas de adjunción de árboles.

7. Obtención del resultado, determinando si la cadena de entrada pertenece o no al lenguaje definido por la gramática de adjunción utilizada para el análisis.

Poller describe en [148] un algoritmo incremental de izquierda a derecha, utilizado para analizar una variante de TAG con la misma capacidad generativa denominada TAG(LD/TLP) en el que los árboles elementales se descomponen en árboles elementales de descripción de dominancia local (LD) y relaciones de precedencia lineal (TLP). El proceso de análisis se reparte entre las dos fases siguientes:

1. Análisis del esqueleto independiente del contexto de la gramática de adjunción mediante una variación del algoritmo de Earley [69] que genera ítems con punteros adicionales con el fin de permitir navegar entre las derivaciones independientes del contexto para, en una segunda fase, obtener los árboles derivados de acuerdo con la gramática de adjunción.

2. Recuperación de los árboles TAG que no tienen operaciones de adjunción pendientes. Para ello se van reconociendo de modo ascendente los árboles elementales que forman parte de un árbol de derivación.

Poller y Becker presentan en [149] una versión simplificada del algoritmo anterior para TAG limpias[13]. El primer paso consiste en la aplicación del algoritmo de Earley estándar sobre el esqueleto independiente del contexto de la gramática de adjunción a analizar. El segundo paso consiste en un proceso iterativo de eliminación de árboles adjuntados a partir de los ítems generados en el primer paso, lo que se corresponde con una estrategia ascendente para la obtención del árbol derivado. Como los autores señalan, la eficiencia práctica del algoritmo varía según el método elegido para la obtención del esqueleto independiente de contexto de la TAG a analizar. Se proponen los dos métodos siguientes.:

1. Considerar únicamente las etiquetas de los nodos para generar las producciones independientes del contexto.

2. Considerar la dirección de los nodos para construir dichas producciones. En este caso, durante la aplicación del algoritmo de Earley se pueden chequear algunas restricciones de adjunción durante la predicción de no-terminales en el algoritmo de Earley.

Con el primer método se obtiene una mayor compartición en la aplicación de la primera fase del algoritmo de análisis mientras que con el segundo se produce un mejor filtrado de derivaciones incorrectas con respecto a la gramática de adjunción.

3.9.4. Algoritmos basados en LIG

Algunos autores consideran que aunque las gramáticas de adjunción son un formalismo adecuado para la descripción de fenómenos lingüísticos, su análisis sintáctico puede realizarse de modo más eficiente aplicando una transformación previa a otro formalismo más adecuado para

[13]Una TAG es limpia si los nodos raíz de los árboles elementales y los nodos pie de los árboles auxiliares tienen restricciones de adjunción nulas. Cualquier TAG puede ser transformada fácilmente en una TAG limpia *casi* fuertemente equivalente [149] añadiendo un nodo adicional a cada árbol elemental que tenga como hijo el nodo raíz de dicho árbol, con la misma etiqueta que el nodo raíz y restricción de adjunción nula, más un nodo adicional por cada árbol auxiliar que tenga como padre al nodo pie de dicho árbol, con la misma etiqueta que el nodo pie y restricción de adjunción nula.

el tratamiento computacional. Generalmente, el formalismo elegido suele ser el de las gramáticas lineales de índices (LIG) [75].

En esta línea, Vijay-Shanker y Weir describen en [213] y [214] dos métodos muy similares de transformación de gramáticas de adjunción de árboles en gramáticas lineales de índices, a las que posteriormente aplican un algoritmo tabular de tipo CYK, lo que conlleva una limitación en la forma de las gramáticas de adjunción que pueden ser tratadas, puesto que el número de hijos de cada nodo no puede ser mayor que dos. Este inconveniente puede evitarse aplicando el método general de transformación de TAG a LIG descrito en la sección 2.4.3, el cual no establece ninguna restricción en el número de hijos de cada nodo.

Para permitir sustitución simultánea, Schabes y Shieber utilizan en [175] un algoritmo de tipo Earley para LIG que no es general, sólo apto para la transformación de TAG a LIG propuesta.

Schabes muestra en [170] cómo analizar TAG lexicalizadas estocásticas transformándolas a LIG estocásticas y aplicando un algoritmo de tipo CYK para analizar estas últimas.

Díaz Madrigal y Toro Bonilla presentan en [66] un algoritmo de análisis sintáctico de TAG basado en una representación plana de los árboles elementales [64] implementable en Prolog que realiza implícitamente una conversión a LIG, puesto que en cada momento del proceso de análisis se tienen en cuenta la pila de adjunciones asociada al nodo que se está tratando y justamente, cuando se transforma una TAG a LIG se pretende que la pila de índices asociada a un nodo represente las adjunciones pendientes en la espina de la que forma parte dicho nodo.

De Kercadio sigue un enfoque similar en [51], utilizando una representación plana de los árboles elementales muy similar a la presentada por Díaz Madrigal et al. en [64] e introduciendo una pila en los ítems que permite correlacionar adecuadamente las predicciones de adjunción con el reconocimiento de los nodos pie. El algoritmo resultante satisface la propiedad del prefijo válido.

3.9.5. Algoritmos LR

Los algoritmos de análisis de la familia LR [6] pueden considerarse entre los algoritmos de análisis más potentes para gramáticas independientes del contexto. Su potencia se basa en la precompilación de información descendente de la gramática en un conjunto de tablas de análisis sintáctico que posteriormente serán utilizadas para guiar un analizador sintáctico ascendente durante el análisis de cada cadena de entrada.

Algunos autores han intentado extender los algoritmos LR a la clase de las gramáticas de adjunción. Sin embargo la tarea es dificultosa puesto que la información compilada en las tablas de análisis LR para gramáticas independientes del contexto se corresponde básicamente con la información proporcionada por los pasos predictivos del algoritmo de Earley [11, 12]. Sin embargo, en el caso de gramáticas de adjunción de árboles los algoritmos de tipo Earley realizan dos tipos adicionales de predicción acerca de la operación de adjunción y del reconocimiento del pie.

A continuación presentamos los distintos intentos de extender la técnica LR a las gramáticas de adjunción de árboles que han sido publicados hasta el momento. Normalmente, estos algoritmos no están basados en técnicas tabulares, sino que trabajan directamente sobre cierto tipo de extensión de los autómatas a pila. Sin embargo, los autores de los distintos algoritmos argumentan que es posible extender al caso de TAG las técnicas de tabulación diseñadas para los algoritmos LR que trabajan con gramáticas independientes del contexto [199, 12] con el fin de poder tratar gramáticas ambiguas.

El algoritmo LR de Schabes y Vijay-Shanker

Schabes y Vijay-Shanker [176, 168] han propuesto un algoritmo de análisis sintáctico para gramáticas de adjunción de árboles deterministas[14] basado en un extensión de la técnica de análisis LR para lenguajes independientes del contexto. En este caso, en lugar de utilizar un autómata a pila como mecanismo operacional, se utiliza la versión ascendente del autómata a pila embebido, denominada BEPDA[15].

Un analizador sintáctico LR para TAG consiste de una cadena de entrada, una salida, una secuencia de pilas, un programa conductor y una tabla de análisis sintáctico con tres partes (ACCIÓN, IR_A$_{derecha}$, IR_A$_{foot}$). Al igual que en el caso de los analizadores LR para lenguajes independientes del contexto, el programa de análisis sintáctico es el mismo para todos los analizadores LR, de modo que para analizar una gramática u otra sólo es preciso cambiar las tablas de análisis. El programa de análisis lee la cadena de entrada de izquierda a derecha, un carácter cada vez, y utiliza la secuencia de pilas para almacenar los estados.

En los analizadores LR para lenguajes independientes del contexto, cada estado del autómata finito utilizado para guiar el proceso de análisis se construye mediante la cerradura de un conjunto de producciones con punto, cerradura que se realiza aplicando sucesivamente el paso predictivo del algoritmo de Earley a cada una de las producciones del conjunto. En el caso de los analizadores LR para TAG también existe una fuerte relación entre estos y los algoritmos de análisis de tipo Earley para TAG sin la propiedad del prefijo válido, concretamente con la versión definida mediante el esquema de análisis **E**, de tal modo que cada estado del autómata finito asociado con un analizador LR para TAG se define como la cerradura bajo los pasos deductivos $\mathcal{D}_{\mathrm{E}}^{\mathrm{AdjPred}}$, $\mathcal{D}_{\mathrm{E}}^{\mathrm{FootPred}}$, $\mathcal{D}_{\mathrm{E}}^{\mathrm{Pred}}$ y $\mathcal{D}_{\mathrm{E}}^{\mathrm{Comp}}$ de un conjunto de producciones con punto, prescindiendo de los índices sobre la cadena de entrada. Una consecuencia de aplicar la operación cerradura mediante este conjunto de operaciones, será que los algoritmos LR para TAG no posean la propiedad del prefijo válido, pues mediante $\mathcal{D}_{\mathrm{E}}^{\mathrm{FootPred}}$ se puede predecir el reconocimiento de un subárbol que no se corresponde con ninguna operación de adjunción.

Con respecto a las transiciones entre estados, existen tres tipos diferentes de transiciones que pueden partir de un estado S_i:

1. *Desplazamiento*: si $N^\gamma \rightarrow \delta \bullet a\nu \in S_i$ y $a \in V_T$ entonces hay una transición etiquetada por a hacia el estado S_j, que contiene la producción $N^\gamma \rightarrow \delta a \bullet \nu$. Dicha transición se denota por $S_j \in \mathrm{trans}(S_i, a)$.

2. *Adjunción*: si $N^\gamma \rightarrow \delta \bullet M^\gamma \nu \in S_i$ y $\beta \in \mathrm{adj}(M^\gamma)$, entonces $S_j \in \mathrm{trans}(S_i, \beta_{derecha})$, tal que el estado S_j contiene la producción $N^\gamma \rightarrow \delta M^\gamma \bullet \nu$.

3. *Pie*: si $\mathbf{F}^\beta \rightarrow \bullet\bot \in S_i$, entonces hay una transición $S_j \in \mathrm{trans}(S_i, \mathbf{F}^\beta)$, tal que el estado S_j contiene la producción $\mathbf{F}^\beta \rightarrow \bot\bullet$.

Las partes IR_A$_{derecha}$ y IR_A$_{foot}$ de la tabla de análisis sintáctico almacenan las transiciones etiquetadas por $\beta_{derecha}$ y por $\mathbf{F}^\beta$, respectivamente.

Con respecto a la tabla de análisis sintáctico, esta se construye a partir de la información proporcionada por el autómata finito. Definimos $\mathrm{trans}(i, x)$ como el conjunto de estados alcanzables desde el esto i mediante la transición etiquetada por x. Los cinco tipos de acciones almacenadas en ACCIÓN(S_i, a) son:

[14] Además de ser deterministas, las gramáticas deben satisfacer la condición de que cada nodo tiene o bien una restricción de adjunción nula o bien una restricción de adjunción obligatoria, aunque este último condicionante no limita la capacidad de reconocimiento puesto que todo lenguaje de adjunción puede describirse mediante una gramática escrita en esta forma.

[15] Los BEPDA se estudian en detalle en el capítulo 7.

1. *Desplazar al estado* S_j: se aplica si y sólo si $j \in \text{trans}(S_i, a)$.
2. *Restaurar derecha de* $M^\gamma \rightarrow \upsilon$: se aplica si y sólo si $N^\gamma \rightarrow \delta \bullet M^\gamma \nu \in S_i$ y M^γ tiene una restricción de adjunción obligatoria.
3. *Reducir raíz de* β: se aplica si y sólo si en el estado S_i hay una producción $\top \rightarrow \mathbf{R}^\beta\bullet$ y $\beta \in \boldsymbol{A}$.
4. *Aceptar*: ocurre si y sólo si el siguiente elemento en la cadena de entrada es el marcador de fin de cadena y en el estado S_i hay una producción $\top \rightarrow \mathbf{R}^\alpha\bullet$ y $\alpha \in \boldsymbol{I}$.
5. *Error*: se aplica si y sólo si ninguna de las otras acciones es aplicable.

El programa conductor procede leyendo el componente léxico a de la cadena de entrada que se está tratando y el estado S_i que está en la cima de la secuencia de pilas y consultando la entrada ACCIÓN(S_i, a) de la tabla de acciones. Los posibles movimientos vienen dados por:

1. ACCIÓN(S_i, a)=*Desplazar al estado* S_j. En este caso se crea una nueva pila conteniendo S_j. Esta acción se corresponde con el paso deductivo $\mathcal{D}_{\text{E}}^{\text{Scan}}$ del esquema de análisis **E**.
2. ACCIÓN(S_i, a)=*Restaurar derecha de* $M^\gamma \rightarrow \upsilon$. Esta acción se corresponde con el paso deductivo $\mathcal{D}_{\text{E}}^{\text{FootComp}}$ del esquema de análisis **E**. Se trata por tanto de continuar el análisis a partir del árbol auxiliar β que ha sido adjuntado al nodo M^γ cuyo subárbol acabamos de reconocer. Se pueden dar dos casos:
   a) Si $\gamma \in \boldsymbol{I}$ o bien $\gamma \in \boldsymbol{A}$ pero el subárbol enraizado por M^γ no incluye al nodo pie de γ, se toman las k pilas unitarias superiores de la secuencia de pilas, donde k es el número de terminales en la frontera del subárbol enraizado por M^γ, que se unen para formar una sola pila, y se sitúa una nueva pila en la cima conteniendo $S_m = \text{IR\_A}_{foot}(S_k, \beta)$, donde S_k es el estado contenido en la pila situada inmediatamente debajo de las k pilas unidas y $\beta \in \text{adj}(M^\gamma)$.
   b) Si $\gamma \in \boldsymbol{A}$ y el subárbol enraizado por M^γ incluye al nodo pie de γ, la operación es más compleja: si k_1 y k_2 son respectivamente el número de terminales a la derecha y a la izquierda del pie de γ, debemos unir los elementos de la pila en posición k_1+1 (que corresponden al análisis del pie) con los elementos de las k_2 pilas unitarias bajo esa pila (que contienen los terminales de la frontera a la izquierda del nodo pie) y los elementos de las k_1 pilas unitarias superiores (que contienen los terminales de la frontera a la derecha del pie), situando una nueva pila unitaria conteniendo $m \in \text{IR\_A}_{foot}(S_{k'}, \beta)$ en la cima de la secuencia de pilas, donde $S_{k'}$ es el elemento de la pila unitaria situada inmediatamente debajo de todas las pilas unidas.
3. ACCIÓN(S_i, a)=*Reducir raíz de* β. Esta acción se corresponde con el paso deductivo $\mathcal{D}_{\text{E}}^{\text{AdjComp}}$ del esquema de análisis **E**. Se realiza, por tanto, cuando el analizador sintáctico ha terminado el análisis del árbol auxiliar β y debe eliminarse la información acerca de β que hay en la pila para continuar el análisis en el nodo en el cual se realizó la adjunción. Denotaremos mediante k_1 y k_2 al número de terminales a la derecha y a la izquierda del pie de β y mediante k_3 al número de terminales a la izquierda del nodo pie de β que son subsumidos por el nodo de la espina más cercano a la raíz que tiene una restricción de adjunción obligatoria. Los $k_1 + k_3 + 1$ símbolos de la pila en la cima de la secuencia de pilas son extraídos y las $k_2 - k_3$ pilas unitarias que se encuentran bajo la nueva pila de la cima son eliminadas. Sea S_j el elemento de la cima de la nueva secuencia de pilas. Dicho elemento es extraído, mientras una nueva pila unitaria es situada en la cima, cuyo contenido es $m \in \text{IR\_A}_{derecha}(S_j, \beta)$.

Lamentablemente, el algoritmo es incorrecto ya que el último paso descrito no realiza todas las comprobaciones pertinentes [99, 123]. Puesto que el paso deductivo $\mathcal{D}_{\mathrm{E}}^{\mathrm{FootPred}}$ se utiliza para realizar la cerradura de los estados, para cada pie se predicen todos los posibles nodos de *cualquier* árbol auxiliar en el que se pueda realizar la adjunción de β. Al aplicar la operación *Restaura Derecha de* $M^{\gamma} \rightarrow \upsilon$ se supone que M^{γ} es el nodo sobre el cual se ha realizado la adjunción de β, pero esto no tiene porqué ser cierto. La primera consecuencia de este hecho es la pérdida de la propiedad del prefijo válido pero una consecuencia más importante es que, al no realizarse ninguna comprobación adicional durante la operación *Reduce Raíz de* β puede que demos por terminada la adjunción de β a un árbol α de modo incorrecto, puesto que estaremos asociado al pie un subárbol incorrecto. Comprobar el número de terminales no es suficiente para garantizar que el subárbol que se ha utilizado para reconocer el pie del árbol auxiliar β coincide con el árbol que se está utilizando para terminar la adjunción de β.

El algoritmo LR de Kinyon

Kinyon ha propuesto un nuevo algoritmo de análisis LR para gramáticas de adjunción de árboles lexicalizadas (LTAG) que toma como punto de partida el algoritmo de Schabes y Vijay-Shanker pero que presenta las siguientes modificaciones respecto a este:

- Admite múltiples acciones en cada entrada de la tabla, permitiendo realizar análisis no deterministas.
- Introduce un nuevo tipo de reducciones *Reducir inicial* α sobre los árboles iniciales con el fin de integrar la operación de sustitución utilizada en LTAG.
- Aplica un filtro en la tabla de análisis para sacar partido de la lexicalización de la gramática, puesto que sólo se considerará aquella parte de la tabla que sea consistente con los árboles de análisis anclados en los componentes léxicos de la cadena de entrada.
- La operación de cerradura de los estados del autómata finito utilizado para construir las tablas de análisis sintáctico asocia a cada producción la pila de los nodos de adjunción antecesores que han sido predichos al llegar al pie de un árbol auxiliar. Por tanto, dicha operación cerradura trabaja sobre *ítems* de la forma $[N^{\gamma} \rightarrow \delta \bullet \nu, \mathit{stars}]$, donde *stars* representa la pila de nodos de adjunción. Mediante está pila se corregirá el comportamiento defectuoso que tenía el algoritmo de Schabes y Vijay-Shanker.

La inclusión de una pila en los ítems que conforman la cerradura de los estados LR constituye el principal inconveniente de este algoritmo, puesto que al no estar limitado el tamaño de dicha pila ni por el tamaño de la gramática ni por el de la cadena de entrada, la complejidad tanto espacial como temporal del proceso de construcción del autómata LR aumenta considerablemente. De hecho, la operación cerradura puede no terminar para algunas gramáticas.

A continuación pasamos a describir las particularidades del algoritmo LR de Kinyon con respecto al de Schabes y Vijay-Shanker. La operación cerradura utilizada para crear los estados se realiza aplicando los siguientes pasos a partir de un estado inicial que contiene el ítem $[\top \rightarrow \bullet\mathbf{R}^{\alpha}, \{\ \}]$, donde α es un árbol inicial:

1. AdjPred: Si $[N^{\gamma} \rightarrow \delta \bullet M^{\gamma}\nu, \mathit{stars}] \in S_i$ y $\beta \in \mathrm{adj}(M^{\gamma})$ entonces
   - si $M^{\gamma} \in \mathrm{espina}(\gamma)$ entonces $[\top \rightarrow \bullet\mathbf{R}^{\beta}, \mathit{stars}] \in S_i$
   - si $M^{\gamma} \notin \mathrm{espina}(\gamma)$ entonces $[\top \rightarrow \bullet\mathbf{R}^{\beta}, \{\ \}] \in S_i$
2. FootPred: Si $[\mathbf{F}^{\beta} \rightarrow \bullet\bot, \mathit{stars}] \in S_i$ y $\beta \in \mathrm{adj}(M^{\gamma})$ entonces $[M^{\gamma} \rightarrow \bullet\delta,\ \mathrm{apilar}(M^{\gamma}, \mathit{stars})] \in S_i$.

3. Pred: Si $[N^\gamma \rightarrow \delta \bullet M^\gamma \nu, \mathit{stars}] \in S_i$ y $\mathbf{nil} \in \text{adj}(M^\gamma)$ entonces
   - si $M^\gamma \in \text{espina}(\gamma)$ entonces $[M^\gamma \rightarrow \bullet \upsilon, \mathit{stars}] \in S_i$.
   - si $M^\gamma \notin \text{espina}(\gamma)$ entonces $[M^\gamma \rightarrow \bullet \upsilon, \{\ \}] \in S_i$.
4. Comp: Si $[N^\gamma \rightarrow \delta \bullet M^\gamma \nu, \mathit{stars}'] \in S_i$ y $[M^\gamma \rightarrow \upsilon \bullet, \mathit{stars}] \in S_i$ y $M^\gamma \neq \mathbf{F}^\gamma$ entonces
   - si $M^\gamma \in \text{espina}(\gamma)$ entonces $[N^\gamma \rightarrow \delta M^\gamma \bullet \nu, \mathit{stars}] \in S_i$
   - si $M^\gamma \notin \text{espina}(\gamma)$ entonces $[N^\gamma \rightarrow \delta M^\gamma \bullet \nu, \mathit{stars}'] \in S_i$
5. SubsPred: Si $[N^\gamma \rightarrow \delta \bullet M^\gamma \nu, \mathit{stars}] \in S_i$ y el nodo M^γ está marcado para sustitución y etiqueta(M^γ) = etiqueta$(\mathbf{R}^\alpha)$, entonces $[\top \rightarrow \mathbf{R}^\alpha, \{\ \}] \in S_i$.

Para cada estado S_i se definen los tres tipos siguientes de transiciones:

1. *Desplazamiento*: si $[N^\gamma \rightarrow \delta \bullet M^\gamma \nu, \mathit{stars}] \in S_i$ y etiqueta$(M^\gamma) \in V_T$ o bien M^γ es un nodo de sustitución, entonces hay una transición etiquetada por etiqueta(M^γ) hacia el estado S_j que contiene el ítem $[N^\gamma \rightarrow \delta M^\gamma \bullet \nu, \mathit{stars}]$. Dicha transición se denota $S_j \in \text{trans}(S_i, \text{etiqueta}(M^\gamma))$.
2. *Adjunción*: si $[M^\gamma \rightarrow \delta \bullet, \mathit{stars}] \in S_i$ y $M^\gamma = \text{cima}(\mathit{stars})$ y $\beta \in \text{adj}(M^\gamma)$ entonces $S_j \in \text{trans}(S_i, \beta_{derecha})$, tal que el estado S_j contiene el ítem $[N^\gamma \rightarrow \delta M^\gamma \bullet \nu, \mathit{stars}'']$ donde stars'' es el subconjunto de *stars* formado por los nodos que dominan N^γ .
3. *Pie*: si $[\mathbf{F}^\beta \rightarrow \bullet \bot, \mathit{stars}] \in S_i$ entonces $S_j \in \text{trans}(S_i, \mathbf{F}^\beta)$, tal que el estado S_j contiene el ítem $[\mathbf{F}^\beta \rightarrow \bot \bullet, \mathit{stars}]$.

Con respecto a la tabla de análisis sintáctico, los cinco tipos de acciones almacenadas en ACCIÓN(S_i, a) son:

1. *Desplazar al estado* S_j: se aplica si y sólo si $j \in \text{trans}(S_i, a)$.
2. *Restaurar derecha de* $M^\gamma \rightarrow \upsilon$ *con* β: se aplica si y sólo si $[N^\gamma \rightarrow \delta \bullet M^\gamma \nu, \mathit{stars}] \in S_i$, $M^\gamma = \text{cima}(\mathit{stars})$ y $\beta \in \text{adj}(M^\gamma)$.
3. *Reducir raíz de* β *con* star: se aplica si y sólo si $[\top \rightarrow \mathbf{R}^\beta \bullet, \{\mathit{star}\}] \in S_i$ y $\beta \in \boldsymbol{A}$
4. *Reducir Árbol Inicial* α: se aplica si y sólo si $[\top \rightarrow \mathbf{R}^\beta \bullet, \{\ \}] \in S_i$.
5. *Aceptar*: ocurre si y sólo si el siguiente elemento en la cadena de entrada es el marcador de fin de cadena y $[\top \rightarrow \mathbf{R}^\alpha \bullet, \{\ \}] \in S_i$ y $\alpha \in \boldsymbol{I}$.
6. *Error*: se aplica si y sólo si ninguna de las otras acciones es aplicable.

El programa conductor procede leyendo terminal a de la cadena de entrada que se está tratando y el estado S_i que está en la cima de la secuencia de pilas, a partir de los cuales determina al movimiento a realizar mediante la consulta de la entrada ACCIÓN(S_i, a) de la tabla de acciones:

1. ACCIÓN(S_i, a)=*Desplazar al estado* S_j. En este caso apila S_j.
2. ACCIÓN(S_i, a)=*Restaurar derecha de* $M^\gamma \rightarrow \upsilon$ *con* β. Se pueden dar dos casos:

a) Si M^γ no domina el nodo pie de γ, sea k el número de terminales más el número de nodos sustituibles dominados por M^γ. Se fusionan las k pilas de la cima, situando una nueva pila unitaria conteniendo $m \in \text{IR\_A}_{foot}(S_{k'}, \beta)$ en la cima de la secuencia de pilas, donde $S_{k'}$ es el elemento de la pila unitaria situada inmediatamente debajo de todas las pilas fusionadas.

b) Si M^γ domina el nodo pie de γ, sean k_1 y k_2 el número de terminales más el número de nodos sustituibles a la derecha y a la izquierda del pie de γ, respectivamente. Debemos unir los elementos de la pila en posición $k_1 + 1$ con los elementos de las k_2 pilas unitarias bajo la pila k_1+2, situándola en la posición k_1+2. Crearemos una nueva pila unitaria en la cima de la secuencia de pilas conteniendo $m \in \text{IR\_A}_{foot}(S_{k_1+1}, \beta)$.

3. ACCIÓN(S_i, a)=*Reducir Raíz de* β *con star*. Sean k_1 y k_2 el número de terminales más el número de nodos sustituibles a la derecha y a la izquierda del pie de β. Sea $A = \text{etiqueta}(star)$ y p el número de símbolos terminales más el número de nodos de sustitución a la izquierda del nodo pie dominado por *star*. En caso de que $\{star\} = \emptyset$ tomaremos $p = 0$. Se extraerán de la pila los $p + k_2 + 1$ elementos superiores y a continuación se extraerán $k_1 - p$ pilas de tamaño 1 . Sea S_i el nuevo elemento en la cima de la pila. Dicho elemento de extraerá y en su lugar se insertará $S_j \in \text{IR\_A}_{derecha}\beta$.

4. ACCIÓN(S_i, a)=*Reducir Árbol Inicial* α. Sea k el número de símbolos terminales más el número de nodos sustituibles dominados por la raíz de α. Entonces k elementos serán eliminados de la pila y se apilará $m \in \text{trans}(S_{k+1}, \text{etiqueta}(\mathbf{R}^\alpha))$.

El algoritmo procede a aplicar todas las acciones posibles hasta que se alcanza el reconocimiento de la cadena de entrada o se obtiene un error. Cuando en un momento dado son posibles varias acciones alternativas y el filtrado no es suficiente para eliminar el no determinismo, se deberán explorar todas las posibles alternativas utilizando técnicas basadas en retroceso o técnicas pseudo-paralelas basadas en programación dinámica [107, 199].

El algoritmo LR de Nederhof

Nederhof presenta en [123] un algoritmo de tipo LR para TAG que trata de solventar los inconvenientes del algoritmo de Kinyon. Entre las diferencias más destacadas, reseñamos las siguientes:

- Utiliza una autómata a pila en vez de un BEPDA como modelo operacional.
- No utiliza una tabla de acciones para almacenar las acciones a realizar durante el proceso. En su lugar, las acciones se determinan dinámicamente consultando el contenido de la pila.
- Es preciso realizar modificaciones para que admita árboles con nodos hoja etiquetados por ϵ.

La operación cerradura utilizada para la construcción del autómata finito que guía el proceso de análisis trabaja con ítems que contienen, además de una producción independiente del contexto con punto, el nodo correspondiente a la raíz del subárbol sobre el que se está trabajando[16]. Este elemento muestra su utilidad en las operaciones relacionadas con el reconocimiento del pie de un árbol auxiliar involucrado en una operación de adjunción. Los ítems son por tanto de la forma $[M^\gamma, M^\gamma \rightarrow \delta \bullet \nu]$. Para realizar la operación cerradura es necesario aplicar los siguientes pasos hasta que no puedan crear más estados, partiendo de un estado con el ítem $[\top^\alpha, \top \rightarrow \bullet \mathbf{R}^\alpha]$:

[16]Para representar que se está trabajando con el árbol γ al completo, se utilizará el nodo artificial $\top^\gamma$.

1. AdjPred: Si $[N^\gamma, M^\gamma \rightarrow \delta \bullet P^\gamma \nu] \in S_i$ y $\beta \in \text{adj}(P^\gamma)$ entonces $[\top^\beta, \top \rightarrow \bullet \mathbf{R}^\beta] \in S_i$.
2. FootPred: Si $[N^\beta, \mathbf{F}^\beta \rightarrow \bullet \bot] \in S_i$ y $\beta \in \text{adj}(P^\gamma)$ entonces $[P^\gamma, P^\gamma \rightarrow \bullet \delta] \in S_i$.
3. Pred: Si $N^\gamma, M^\gamma \rightarrow \delta \bullet P^\gamma \nu] \in S_i$ y $\mathbf{nil} \in \text{adj}(P^\gamma)$ entonces $[N^\gamma, P^\gamma \rightarrow \bullet \upsilon] \in S_i$.
4. Comp: Si $N^\gamma, P^\gamma \rightarrow \upsilon \bullet] \in S_i$ entonces $[N^\gamma, M^\gamma \rightarrow \delta P^\gamma \bullet \nu] \in S_i$.

Con respecto a las transiciones entre estados, existen los tres tipos siguientes:

1. *Desplazamiento*: Si $[N^\gamma, M^\gamma \rightarrow \delta \bullet a\nu] \in S_i$ donde $a \in V_T$ entonces existe una transición etiquetada por a hasta el estado S_j que contiene el ítem $[N^\gamma, M^\gamma \rightarrow \delta a \bullet \nu]$. Dicha transición se denota por $\text{IR\_A}(S_i, a)$.
2. *Adjunción*: Si $[N^\gamma, M^\gamma \rightarrow \delta \bullet P^\gamma \nu] \in S_i$ tal que $\beta \in \text{adj}(P^\gamma)$ para algún β, entonces existe una transición etiquetada por P^γ hasta el estado S_j que contiene el ítem $[N^\gamma, M^\gamma \rightarrow \delta P^\gamma \bullet \nu]$. Dicha transición se denota, al igual que en el caso precedente, por $S_j \in \text{IR\_A}(S_i, P^\gamma)$.
3. *Pie*: Si $[N^\beta, \mathbf{F}^\beta \rightarrow \bullet \bot] \in S_i$ y $\beta \in \text{adj}(P^\gamma)$ entonces existe una transición etiquetada por P^γ hasta el estado S_j que contiene el ítem $[N^\beta, \mathbf{F}^\beta \rightarrow \bot \bullet]$. Dicha transición se denota $S_j \in \text{IR\_A}_\bot(S_i, P^\gamma)$.

Para determinar las acciones a realizar durante el proceso de análisis se calculan las dos tablas siguientes a partir de los estados del autómata finito:

1. *Reducciones*: La tabla de reducciones contiene una entrada para cada estado de modo que $reducciones(S_i) = \{\beta \in \boldsymbol{A} \mid [\top^\beta, \top \rightarrow \mathbf{R}^\beta \bullet] \in S_i\} \cup \{N^\gamma \mid [N^\gamma, N^\gamma \rightarrow \delta \bullet]\}$.
2. *Secciones cruzadas*: La tabla de secciones cruzadas tiene una entrada por cada nodo de cada árbol elemental, de tal modo que $CS(N^\gamma)$ contiene todas las posibles fronteras del subárbol enraizado por N^γ, suponiendo que se eliminan sucesivamente los subárboles que lo constituyen. Denotamos mediante $CS(N^\gamma)^+$ a $CS(N^\gamma) - \{N^\gamma\}$. Estas tablas realizan una labor esencial en el algoritmo de análisis, ya que permiten decidir si se puede realizar la reducción de un nodo N^γ. Para ello es necesario que los elementos de la cima de la pila se correspondan con una de las secuencias almacenadas en la entrada de $CS(N^\gamma)^+$. Una forma eficiente de implementar las tablas de secciones cruzadas consiste en utilizar un autómata finito para almacenar el contenido de cada entrada, en lugar de un conjunto de secuencias de nodos.

Como hemos comentado anteriormente, este algoritmo trabaja sobre un autómata a pila. La estructura de la pila es compleja, pues almacena tres tipos diferentes de elementos:

1. Estados S_i del autómata finito.
2. Nodos N^γ de árboles elementales.
3. Nodos de árboles elementales junto a una pila de nodos $M^\gamma[N \mid L]$, donde N representa el nodo en la cima de la pila y L el resto de la pila[17]. Cada una de estas pilas contiene una parte de la espina del árbol derivado en la que se van acumulando los nodos en los que se ha realizado una adjunción que aún no ha sido completada. Como el algoritmo procede de modo ascendente, las adjunciones son reconocidas en el pie y completadas en la raíz de un árbol auxiliar. Esta pilas se corresponden con las pilas asociadas a los símbolos

[17]Las secciones cruzadas para este tipo de elementos se calculan ignorando la pila de nodos.

gramaticales de una LIG que representa a una TAG. También constituyen una diferencia fundamental con el algoritmo de Kinyon, puesto que saca dichas pilas de los ítems del autómata finito para introducirlas en la pila que se manipula en tiempo de ejecución.

Dada una determinada configuración de la pila, la acción a realizar es una de las cinco siguientes:

1. *Desplazar al estado S_j*: se aplica cuando en la cima de la pila se encuentra el estado S_i, el siguiente elemento de la cadena de entrada es un terminal a y $S_j \in \text{IR\_A}(S_i, a)$.

2. *Reducir subárbol*: se aplica cuando en la cima de pila se encuentra la secuencia $S_0X_1S_1X_2S_2 \ldots X_mS_m$ tal que $X_1 \ldots X_m \in CS(N^\gamma)^+$ y $S' \in \text{IR\_A}_\perp(S_0, N^\gamma)$. Entonces dicha secuencia es reemplazada por $S_0 \perp[N^\gamma \mid L]\ S'$. Si N^γ domina un nodo pie entonces L se obtiene del elemento $X_i = M^\gamma[L]$, en caso contrario L es la pila vacía. Lo que se ha hecho es reconocer la sección cruzada de un subárbol de un árbol elemental γ enraizado por un nodo N^γ que admite adjunciones. Como resultado se ha sustituido dicha sección cruzada y los estados asociados por el pie de un posible árbol auxiliar adjuntado con el fin continuar el reconocimiento ascendente de este último. Esta acción se corresponde con la acción *Restaurar Derecha* del algoritmo de Kinyon.

3. *Reducir árbol auxiliar*: se aplica cuando en la cima de la pila se encuentra la secuencia $S_0X_1S_1X_2 \ldots X_mq_m$ y $\beta \in reducciones(S_m)$, $X_1 \ldots X_m \in CS(\mathbf{R}^\beta)^+$ y $S' \in \text{IR\_A}(S_0, N^\gamma)$, donde N^γ se obtiene a partir del único $X_i = M[N \mid L]$. Entonces dicha secuencia es reemplazada por S_0XS', donde $X = N$ si L es la pila vacía y $X = N[L]$ en otro caso. Lo que se ha hecho es reconocer la sección cruzada de la raíz de un árbol auxiliar β, sustituyendo dicha sección y sus estados asociados por el nodo en el cual se realizó la adjunción de β. Esta acción se corresponde con la acción *Reducir Raíz de β* del algoritmo de Kinyon.

4. *Aceptar*: ocurre si y sólo si en la cima de pila se encuentra el estado final y el siguiente elemento en la cadena de entrada es el marcador de fin de cadena.

5. *Error*: se aplica si y sólo si ninguna de las otras acciones es aplicable.

Los nodos hoja etiquetados por ϵ no dejan ninguna traza en la pila, lo cual presenta problemas a la hora de comprobar si en la cima de pila se encuentra la sección cruzada de algún nodo. Para solventarlo es necesario refinar los ítems utilizados para construir el autómata finito introduciendo un tercer componente que indica los nodos con etiqueta ϵ que han sido visitados por la operación cerradura. Igualmente, es preciso extender el concepto de sección cruzada de modo que al calcularla se tenga en cuenta qué nodos tienen etiqueta ϵ.

Prolo presenta en [150] una variante del algoritmo LR de Nederhof que genera tablas de análisis de menor tamaño.

Generación incremental de las tablas de análisis

La principal ventaja del análisis LR, tanto en el caso de gramáticas independientes del contexto como en el caso de gramáticas de adjunción de árboles es que la información predictiva presente en la gramática se compila en un conjunto de tablas que son utilizadas durante la fase de análisis de las cadenas de entrada. Esta ventaja se convierte en un inconveniente cuando se trabaja en entornos en los que la gramática se ve sometida a cambios frecuentes, puesto que la fase de generación de las tablas consume muchos recursos, tanto en tiempo como en espacio de memoria. Sarkar propone solventar este problema extendiendo las técnicas propuestas para el

caso de analizadores LR de gramáticas independientes del contexto [157, 82, 83] a los analizadores LR para TAG. Su propuesta [163] consiste en combinar la generación perezosa de las tablas de análisis[18] con generación incremental de las tablas de análisis. Cuando se produce algún cambio en la gramática, se seleccionan aquellos estados del autómata finito que se ven afectados por dicha modificación, manteniendo únicamente los ítems núcleo[19]. El autómata será reconstruido de acuerdo con la nueva gramática mediante la aplicación de la generación perezosa durante el análisis de nuevas cadenas de entrada.

Aunque diseñada originalmente para el algoritmo de Schabes y Vijay-Shanker [176], esta técnica de generación incremental de analizadores LR para TAG se puede adaptar fácilmente al algoritmo propuesto por Kinyon.

3.9.6. Algoritmos paralelos

Los algoritmos mostrados hasta el momento son algoritmos secuenciales, pensados para ser programados y ejecutados en ordenadores secuenciales. Su límite de complejidad temporal es $\mathcal{O}(n^p)$. Mediante la paralelización de los algoritmos tabulares se pueden conseguir límites inferiores a costa de aumentar las unidades de procesamiento paralelo necesarias para implementar el algoritmo. El principio en que se basan los algoritmos paralelos es el de explotar la redundancia presente en los algoritmos tabulares para el análisis de TAG.

Se ha propuesto varios algoritmos paralelos para el análisis sintáctico de TAG. Palis, Shende y Wei proponen en [139] una versión paralelizada del algoritmo tabular de tipo CYK [209] para el análisis de gramáticas de adjunción que presenta una complejidad lineal y que precisa una malla 5-dimensional de n^5 procesadores para ejecutarse (donde n es la longitud de la cadena de entrada), con la ventaja adicional de que cada procesador realiza una función que no es dependiente de la longitud de la cadena de entrada sino únicamente de la gramática que se está analizando. Este algoritmo, al estar derivado del algoritmo CYK, sólo trabaja con gramáticas de adjunción de árboles en forma binaria, en la que los nodos de los árboles elementales no tienen más de dos hijos. Posteriormente Palis y Wei han propuesto un nuevo algoritmo para gramáticas de adjunción de árboles en forma general [141]. El comportamiento de dicho algoritmo depende fuertemente del tamaño de la gramática, aunque siempre obtiene complejidades inferiores a las versiones secuenciales. Las versiones SIMD de sus algoritmos [140, 141], implementadas en una *Connection Machine CM-2* ofrecen una aceleración con respecto a las versiones secuenciales que muestra un comportamiento asintótico con respecto al tamaño de la gramática.

Nurkkala y Kumar [136] proponen una variación del algoritmo de Palis y Wei [141] para máquinas MIMD conectadas en hipercubo, implementada en una máquina nCUBE/2. Su algoritmo proporciona mayor granularidad al asignar un conjunto dado de árboles elementales a cada procesador. Ciertos resultados parciales del análisis, concretamente el reconocimiento de la raíz de un árbol elemental, se distribuyen asíncronamente a través del multicomputador. Concurrentemente con la ejecución del proceso de análisis, se aplica un algoritmo de detección de terminación distribuido asíncrono. Los resultados experimentales muestran una mejora en la aceleración con respecto al algoritmo de Palis y Wei. Los mismo autores presentan en [137] un algoritmo de tipo Earley para TAG, paralelizado para un multicomputador *KSR1* de memoria compartida con tiempo de acceso no uniforme.

Rajasekaran propone en [151] un algoritmo de tipo Earley ascendente para gramáticas de adjunción de árboles en forma binaria que presenta una complejidad temporal $\mathcal{O}(n \log n)$ cuando

[18] Las tablas de análisis se van generando a medida que son requeridas durante el análisis de una cadena de entrada. Esta técnica es útil cuando la gramática es muy grande y sólo una pequeña parte de la misma será utilizada para analizar una cadena de entrada. Su inconveniente es el elevado consumo de memoria, puesto que el autómata finito utilizado para generar las tablas debe estar también presente durante la fase de análisis.

[19] Aquellos que provocan la aplicación de las operaciones de cerradura de un estado.

se implementa en una máquina paralela *EREW PRAM* con $\frac{n^2 M(n)}{\log n}$ procesadores, donde $M(k)$ es el tiempo necesario para multiplicar dos matrices booleanas de tamaño $k \times k$.