Algoritmos tabulares para el análisis de TAG*

Miguel A. Alonso Pardo
Departamento de Computación
Facultad de Informática
Campus de Elviña s/n
15071 La Coruña, España
alonso@dc.fi.udc.es

David Cabrero Souto
Centro Ramón Piñeiro para a
Investigación en Humanidades
Estrada Santiago-Noia km 3, A Barcia
15896 Santiago de Compostela, España
dcabrero@cirp.es

Eric de la Clergerie
INRIA
Domaine de Voluceau
Rocquencourt, B.P. 105
78153 Le Chesnay Cedex, Francia
Eric.De_La_Clergerie@inria.fr

Manuel Vilares Ferro
Departamento de Computación
Facultad de Informática
Campus de Elviña s/n
15071 La Coruña, España
vilares@dc.fi.udc.es

Resumen

En este trabajo se realiza una descripción de varios algoritmos tabulares para el análisis sintáctico de las Gramáticas de Adjunción de Árboles, creando una línea evolutiva continua desde los algoritmos más simples a los más complejos y mostrando las transformaciones que deben ser aplicadas a cada uno de ellos para obtener el siguiente en la cadena evolutiva. Varios de los algoritmos descritos lo son por vez primera, tal es el caso del algoritmo de tipo Earley ascendente y la versión propuesta del algoritmo de tipo Earley sin la propiedad del prefijo válido.

1. Introducción

Las Gramáticas de Adjunción de Árboles (TAG) [4] son una extensión de las gramáticas independientes del contexto que utilizan árboles en vez de producciones como estructuras elementales. Formalmente, una TAG es una quíntupla $(V_N, V_T, S, \boldsymbol{I}, \boldsymbol{A})$, donde V_N es un conjunto finito de no-terminales, V_T es un conjunto finito de terminales, S es el axioma de la gramática, $\boldsymbol{I}$ es un conjunto finito de *árboles iniciales* y $\boldsymbol{A}$ es un conjunto finito de *árboles auxiliares*. $I \cup A$ es el conjunto de *árboles elementales*. Los nodos internos de dichos árboles están etiquetados por no-terminales mientras que los nodos hoja lo están por terminales o bien por la cadena vacía ϵ, excepto un único nodo hoja, denominado *pie*, de cada árbol auxiliar que está etiquetado por el mismo no-terminal que etiqueta la raíz del árbol. El camino que recorre los nodos desde la raíz hasta el pie se denomina *espina*.

La derivación de nuevos árboles se realiza mediante la operación de *adjunción*: dado un árbol elemental α con un nodo etiquetado por el mismo no-terminal que la raíz y el pie de un árbol auxiliar β, la adjunción de β en α se realiza escindiendo el subárbol de α que tiene como raíz el nodo que se está considerando (que recibe el nombre de *nodo de adjunción*), pegando β a dicho nodo y pegando el subárbol escindido al pie de β. Notaremos mediante $\beta \in adj(A^\alpha)$ la posibilidad de que el nodo A de α sea nodo de adjunción de β.

En la descripción de los algoritmos de análisis para TAG, tenemos que representar el reconocimiento parcial de los árboles elementales. En las gramáticas independientes del contexto se utilizan habitualmente reglas punteadas para este fin. En el caso de TAG, consideraremos cada árbol elemental

*Financiado en parte por la Unión Europea mediante el proyecto 1FD97-0047-C04-02, por los ministerios de Educación y Cultura y Asuntos Exteriores a través de la acción integrada HF97-223 y por la Xunta de Galicia mediante los proyectos XUGA10505B96 y XUGA20402B97.

γ como constituido por un conjunto de reglas independientes del contexto $\mathcal{P}(\gamma)$ e indicaremos la posición del punto en el árbol indicando la posición en la regla correspondiente. Los elementos de las reglas serán los nodos del árbol, excepto en el caso de los elementos del lado derecho de las reglas cuyas etiquetas pertenecen a $V_T \cup \varepsilon$, puesto que al no poder tener descendientes ni ser susceptibles de ser nodos de adjunción consideraremos directamente su etiqueta a la hora de escribir las reglas.

Con el fin de simplificar la descripción de los algoritmos seguiremos el enfoque de [5] de considerar la regla adicional $\top \rightarrow \mathbf{R}^{\alpha}$ para cada árbol inicial α y las dos reglas 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, los nuevos nodos $\top$ y $\bot$ no pueden ser nodos de adjunción.

1.1 Esquemas de análisis sintáctico

Describiremos los algoritmos de análisis mediante esquemas de análisis sintáctico [8], una estructura para realizar descripciones de alto nivel de algoritmos de análisis sintáctico que permite estudiar fácilmente las relaciones entre diferentes algoritmos mediante el análisis de las relaciones formales entre los esquemas de análisis subyacentes.

Un *sistema de análisis sintáctico* para una gramática $\mathcal{G}$ y una cadena de entrada $a_1 \ldots a_n$ es un triple $\mathbb{P} = \langle \mathcal{I}, \mathcal{H}, \mathcal{D} \rangle$, donde $\mathcal{I}$ es un conjunto de *ítems* que representan resultados intermedios del proceso de análisis, $\mathcal{H}$ es un conjunto inicial de ítems denominado *hipótesis* que representan la cadena que va a ser analizada y $\mathcal{D}$ es un conjunto de reglas de inferencia denominadas *pasos deductivos*, mediante las cuales se derivan nuevos ítems ξ a partir de los ítems η_i existentes. Estos pasos deductivos tienen la forma $\frac{\eta_1,\ldots,\eta_k}{\xi}$ y su significado es el siguiente: si todos los antecedentes $\eta_i \in \mathcal{H} \cup \mathcal{I}$ de un paso deductivo ya existen, entonces el consecuente ξ deberá ser generado por el analizador sintáctico. Los pasos deductivos se suponen cuantificados universalmente para todas las posibles variables que aparezcan en los ítems a menos que se indique explícitamente lo contrario.

La presencia de un conjunto $\mathcal{F} \subseteq \mathcal{I}$ de *ítems finales* indica el reconocimiento de la cadena de entrada.

Un *sistema de análisis sintáctico no instanciado* para una gramática $\mathcal{G}$ es un triple $\langle \mathcal{I}, \mathcal{H}, \mathcal{D} \rangle$ con $\mathcal{H}$ una función que asigna un conjunto de hipótesis a cada cadena de entrada $a_1 \ldots a_n \in V_T^*$, tal que $\langle \mathcal{I}, \mathcal{H}(a_1 \ldots a_n), \mathcal{D} \rangle$ es un sistema de análisis sintáctico.

Un *esquema de análisis sintáctico* $\mathbf{P}$ para una clase de gramáticas es una función que asigna un sistema de análisis sintáctico no instanciado a cada gramática en dicha clase.

Un esquema de análisis puede ser generalizado a partir de otro mediante las siguientes transformaciones:

- *Refinamiento de los ítems*, rompiendo un ítem individual en varios.
- *Refinamiento de los pasos deductivos*, descomponiendo un paso deductivo simple en una secuencia de pasos.
- *Extensión* de un esquema considerando una clase más amplia de gramáticas.

Las siguientes operaciones de *filtrado* permiten disminuir el número de ítems y pasos deductivos de un esquema de análisis:

- *Filtrado estático*, en el cual los pasos redundantes son simplemente descartados.
- *Filtrado dinámico*, utilizando información contextual para determinar la validez de los ítems .
- *Contracción de pasos*, en la cual una secuencia de pasos deductivos es reemplazada por un sólo paso.

2. Algoritmo de tipo CYK

Como punto de partida tomaremos el algoritmo de tipo CYK descrito en [9]. Asumiremos que todo nodo de un árbol elemental de la gramática tiene a lo sumo dos descendientes. Definimos el siguiente conjunto de ítems de la forma

$$[A^\gamma, i, j \mid p, q \mid adj]$$

tal que $A^\gamma \stackrel{*}{\Rightarrow} a_{i+1} \ldots a_p \ \mathbf{F}^\gamma \ a_{q+1} \ldots a_j \stackrel{*}{\Rightarrow} a_{i+1} \ldots a_j$ si y sólo si $(p,q) \neq (-,-)$ mientras que $A^\gamma \stackrel{*}{\Rightarrow} a_{i+1} \ldots a_j$ si y sólo si $(p,q) = (-,-)$.

Estos ítems corrigen los definidos en [9] introduciendo un nuevo elemento que toma su valor del conjunto $\{\text{true}, \text{false}\}$ y que indica, si su valor es *true* que el ítem es el resultado de una operación de adjunción ya totalmente realizada, si su valor es *false* que el ítem no es el resultado de una adjunción. Con ello podremos limitar a una sola la cantidad de adjunciones que se pueden realizar sobre cada nodo de un árbol elemental, ajustándonos de este modo a lo establecido en el formalismo de las gramáticas de adjunción [7]. También elimina la restricción de que los árboles auxiliares deban derivar al menos un símbolo terminal [9].

Esquema 1 *El sistema de análisis* $\mathbb{I}\mathrm{P}_{\mathrm{CYK}}$ *que se corresponde con el algoritmo CYK para una TAG* $\mathcal{G}$ *y una cadena de entrada* $a_1 \ldots a_n$ *se define como sigue:*

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

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

$$\mathcal{D}^{\mathrm{Scan}}_{\mathrm{CYK}} = \frac{[a, i-1, i]}{[A^\gamma, i-1, i \mid -,- \mid false]} \quad A^\gamma \rightarrow a$$

$$\mathcal{D}^{\varepsilon}_{\mathrm{CYK}} = \frac{}{[A^\gamma, i, i \mid -,- \mid false]} \quad A^\gamma \rightarrow \varepsilon$$

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

$$\mathcal{D}^{\mathrm{LeftDom}}_{\mathrm{CYK}} = \frac{\begin{array}{c}[B^\gamma, i, k \mid p, q \mid adj], \\ [C^\gamma, k, j \mid -,- \mid adj']\end{array}}{[A^\gamma, i, j \mid p, q \mid false]}$$

tal que $A^\gamma \rightarrow B^\gamma C^\gamma$ *y* B^γ *domina* $\mathbf{F}^\gamma$.

$$\mathcal{D}^{\mathrm{RightDom}}_{\mathrm{CYK}} = \frac{\begin{array}{c}[B^\gamma, i, k \mid -,- \mid adj], \\ [C^\gamma, k, j \mid p, q \mid adj']\end{array}}{[A^\gamma, i, j \mid p, q \mid false]}$$

tal que $A^\gamma \rightarrow B^\gamma C^\gamma$ *y* C^γ *domina* $\mathbf{F}^\gamma$.

$$\mathcal{D}^{\mathrm{NoDom}}_{\mathrm{CYK}} = \frac{\begin{array}{c}[B^\gamma, i, k \mid -,- \mid adj], \\ [C^\gamma, k, j \mid -,- \mid adj']\end{array}}{[A^\gamma, i, j \mid -,- \mid false]}$$

tal que $A^\gamma \rightarrow B^\gamma C^\gamma$ *y ni* B^γ *ni* C^γ *domina al nodo pie de* γ.

$$\mathcal{D}^{\mathrm{Unary}}_{\mathrm{CYK}} = \frac{[B^\gamma, i, j \mid p, q \mid adj]}{[A^\gamma, i, j \mid p, q] \mid false} \quad A^\gamma \rightarrow B^\gamma$$

$$\mathcal{D}^{\mathrm{Adj}}_{\mathrm{CYK}} = \frac{\begin{array}{c}[\mathbf{R}^\beta, i', j' \mid i, j \mid adj], \\ [A^\gamma, i, j \mid p, q \mid false]\end{array}}{[A^\gamma, i', j' \mid p, q \mid true]}$$

tal que $\beta \in A$ *y* $\beta \in adj(A^\gamma)$.

$$\begin{array}{rl} \mathcal{D}_{\mathrm{CYK}} = & \mathcal{D}^{\mathrm{Scan}}_{\mathrm{CYK}} \cup \mathcal{D}^{\varepsilon}_{\mathrm{CYK}} \cup \mathcal{D}^{\mathrm{Foot}}_{\mathrm{CYK}} \cup \\ & \mathcal{D}^{\mathrm{LeftDom}}_{\mathrm{CYK}} \cup \mathcal{D}^{\mathrm{RightDom}}_{\mathrm{CYK}} \cup \mathcal{D}^{\mathrm{NoDom}}_{\mathrm{CYK}} \cup \\ & \mathcal{D}^{\mathrm{Unary}}_{\mathrm{CYK}} \cup \mathcal{D}^{\mathrm{Adj}}_{\mathrm{CYK}} \end{array}$$

$$\mathcal{F}_{\mathrm{CYK}} = \left\{ \ [\mathbf{R}^\alpha, 0, n \mid -,- \mid adj] \mid \alpha \in \boldsymbol{I} \ \right\}$$

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. Por consiguiente, no nos volveremos a referir explícitamente a ellas.

Los pasos clave en el sistema de análisis $\mathbb{I}\mathrm{P}_{\mathrm{CYK}}$ son $\mathcal{D}^{\mathrm{Foot}}_{\mathrm{CYK}}$ y $\mathcal{D}^{\mathrm{Adj}}_{\mathrm{CYK}}$, 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 $\mathcal{D}^{\mathrm{Foot}}_{\mathrm{CYK}}$ 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. Como consecuencia, pueden existir 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. Sólo algunos 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 inicial 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.

3. Algoritmo de tipo Earley ascendente

Para superar la limitación de las gramáticas binarias definiremos un algoritmo de tipo Earley ascendente. Como primer paso precisamos introducir puntos en las producciones que representan los árboles elementales, por lo que los ítems serán de la forma

$$[A^\gamma \to \delta \bullet \nu, i, j \mid p, q]$$

tal que $\delta \overset{*}{\Rightarrow} a_{i+1} \dots a_p$ $\mathbf{F}^\gamma$ $a_{q+1} \dots a_j \overset{*}{\Rightarrow} a_{i+1} \dots a_j$ si y sólo si $(p,q) \neq (-,-)$ mientras que $\delta \overset{*}{\Rightarrow} a_{i+1} \dots a_j$ si y sólo si $(p,q) = (-,-)$.

Los ítems del nuevo esquema de análisis sintáctico, que denominaremos $\mathbf{P}_{\text{buE}_1}$, son por tanto un refinamiento de los ítems de $\mathbf{P}_{\text{CYK}}$. Al incluirse reglas con punto en la forma de los nuevos ítems ya no es necesario el elemento que indicaba si se había realizado una adjunción en el nodo situado en el lado izquierdo de la producción contenida en dicho ítem.

Esquema 2 *El sistema de análisis* $\mathbb{P}_{\text{buE}}$ *que se corresponde con el algoritmo de tipo Earley ascendente, dada una gramática de adjunción de árboles* $\mathcal{G}$ *y una cadena de entrada* $a_1 \dots a_n$ *se define como sigue:*

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

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

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

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

$$\mathcal{D}^{\text{Comp}}_{\text{buE}} = \frac{[A^\gamma \to \delta \bullet B^\gamma\nu, i, k \mid p, q], \ [B^\gamma \to \upsilon\bullet, k, j \mid p', q']}{[A^\gamma \to \delta B^\gamma \bullet \nu, i, j \mid p \cup p', q \cup q']}$$

$$\mathcal{D}^{\text{AdjComp}}_{\text{buE}} = \frac{[\top \to \mathbf{R}^\beta\bullet, k, j \mid l, m], \ [B^\gamma \to \upsilon\bullet, l, m \mid p', q'], \ [A^\gamma \to \delta \bullet B^\gamma\nu, i, k \mid p, q],}{[A^\gamma \to \delta B^\gamma \bullet \nu, i, j \mid p \cup p', q \cup q']}$$

tal que $\beta \in \boldsymbol{A}$ *y* $\beta \in \text{adj}(\gamma)$.

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

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

donde $p \cup q$ *se refiere a la operación de unión de índices:*

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

Se ha aplicado un *refinamiento* a los pasos deductivos puesto que los pasos de tipo LeftDom, RightDom y NoDom han sido separados en diferentes pasos de tipo Init y Comp, mientras que los pasos de tipo Unary y ε ya no son necesarios puesto que todas las producciones son tratadas uniformemente con independencia de su longitud. También se ha realizado una *extensión* del dominio de las producciones permitiendo árboles con un grado arbitrario de ramificación.

El algoritmo reconoce 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. Un conjunto de pasos deductivos $\mathcal{D}^{\text{Init}}_{\text{buE}}$ que expresan la expectativa de que una determinada regla pueda ser aplicada para reconocer una porción de la cadena de entrada son los encargados de lanzar el análisis ascendente.

Se ha aplicado un filtro al esquema de análisis sintáctico $\mathbf{P}_{\mathrm{CYK}}$, consistente en la contracción de pasos deductivos puesto que un ítem generado por un paso deductivo de tipo Adj sólo podría ser utilizado en un paso de tipo Comp para avanzar el punto en la regla que había predicho el no terminal del lado izquierdo de su producción. Por ello, hemos creado un nuevo paso deductivo AdjComp y eliminado los pasos Adj.

4. Algoritmo de tipo Earley

Incorporando predicción descendente podemos obtener un algoritmo de tipo Earley. Para ello debemos aplicar dos filtros dinámicos: el paso deductivo $\mathcal{D}_{\mathrm{buE}}^{\mathrm{Init}}$ sólo contendrá producciones cuyo lado izquierdo corresponda con la raíz de un árbol inicial mientras que un nuevo conjunto de pasos predictivos controlará la generación de nuevos ítems tratando de limitarla únicamente a aquellos que puedan resultar útiles en el proceso de análisis.

Esquema 3 *El sistema de análisis* $\mathbb{P}_{\mathrm{E}}$ *que se corresponde con el algoritmo de tipo Earley sin la propiedad del prefijo válido, dada una gramática de adjunción de árboles* $\mathcal{G}$ *y una cadena de entrada* $a_1 \ldots a_n$ *se define como sigue:*

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

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

$$\mathcal{D}_{\mathrm{E}}^{\mathrm{Pred}} = \frac{[A^{\gamma} \rightarrow \delta \bullet B^{\gamma}\nu, i, j \mid p, q]}{[B^{\gamma} \rightarrow \bullet \upsilon, j, j \mid -, -]}$$

$$\mathcal{D}_{\mathrm{E}}^{\mathrm{AdjPred}} = \frac{[A^{\gamma} \rightarrow \delta \bullet B^{\gamma}\nu, i, j \mid p, q]}{[\top \rightarrow \bullet \mathbf{R}^{\beta}, j, j \mid -, -]}$$

tal que $\beta \in \mathrm{a}dj(B^{\gamma})$.

$$\mathcal{D}_{\mathrm{E}}^{\mathrm{FootPred}} = \frac{\begin{array}{c}[\mathbf{F}^{\beta} \rightarrow \bullet\bot, k, k \mid -, -], \\ [A^{\gamma} \rightarrow \delta \bullet B^{\gamma}\nu, i, j \mid p, q]\end{array}}{[B^{\gamma} \rightarrow \bullet\delta, k, k \mid -, -]}$$

tal que $\beta \in \mathrm{a}dj(B^{\gamma})$.

$$\mathcal{D}_{\mathrm{E}}^{\mathrm{FootComp}} = \frac{\begin{array}{c}[B^{\gamma} \rightarrow \delta\bullet, k, l \mid p, q], \\ [\mathbf{F}^{\beta} \rightarrow \bullet\bot, k, k \mid -, -], \\ [A^{\gamma} \rightarrow \delta \bullet B^{\gamma}\nu, i, j \mid p', q']\end{array}}{[\mathbf{F}^{\beta} \rightarrow \bot\bullet, k, l \mid k, l]}$$

tal que $\beta \in \mathrm{a}dj(B^{\gamma})$ *y* $p \cup p'$ *así como* $q \cup q'$ *están definidos.*

$$\mathcal{D}_{\mathrm{E}}^{\mathrm{AdjComp^1}} = \frac{\begin{array}{c}[\top \rightarrow \mathbf{R}^{\beta}\bullet, j, m \mid k, l], \\ [B^{\gamma} \rightarrow \delta\bullet, k, l \mid p, q], \\ [\mathbf{F}^{\gamma} \rightarrow \bot\bullet, p, q \mid p, q], \\ [A^{\gamma} \rightarrow \delta \bullet B^{\gamma}\nu, i, j \mid -, -]\end{array}}{[A^{\gamma} \rightarrow \delta B^{\gamma} \bullet \nu, i, m \mid p, q]}$$

tal que $\beta \in \mathrm{a}dj(B^{\gamma})$.

$$\mathcal{D}_{\mathrm{E}}^{\mathrm{AdjComp^2}} = \frac{\begin{array}{c}[\top \rightarrow \mathbf{R}^{\beta}\bullet, j, m \mid k, l], \\ [B^{\gamma} \rightarrow \delta\bullet, k, l \mid -, -], \\ [A^{\gamma} \rightarrow \delta \bullet B^{\gamma}\nu, i, j \mid p, q]\end{array}}{[A^{\gamma} \rightarrow \delta B^{\gamma} \bullet \nu, i, m \mid p, q]}$$

tal que $\beta \in \mathrm{a}dj(B^{\gamma})$.

$$\begin{aligned}\mathcal{D}_{\mathrm{E}} = \quad & \mathcal{D}_{\mathrm{E}}^{\mathrm{Init}} \cup \mathcal{D}_{\mathrm{buE}}^{\mathrm{Scan}} \cup \mathcal{D}_{\mathrm{E}}^{\mathrm{Pred}} \cup \mathcal{D}_{\mathrm{buE}}^{\mathrm{Comp}} \cup \\ & \mathcal{D}_{\mathrm{E}}^{\mathrm{AdjPred}} \cup \mathcal{D}_{\mathrm{E}}^{\mathrm{FootPred}} \cup \mathcal{D}_{\mathrm{E}}^{\mathrm{FootComp}} \cup \\ & \mathcal{D}_{\mathrm{E}}^{\mathrm{AdjComp^1}} \cup \mathcal{D}_{\mathrm{E}}^{\mathrm{AdjComp^2}}\end{aligned}$$

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

El análisis comienza creando un ítem correspondiente a una producción del nodo raíz de un árbol inicial con el punto situado en el extremo izquierdo. Posteriormente, los pasos $\mathcal{D}_{\mathrm{E}}^{\mathrm{Pred}}$ y $\mathcal{D}_{\mathrm{E}}^{\mathrm{AdjComp}}$ se encargan de ir recorriendo cada árbol elemental. Los pasos $\mathcal{D}_{\mathrm{E}}^{\mathrm{AdjPred}}$ predicen la adjunción de un árbol β en un nodo de un árbol elemental γ, comenzando el análisis del árbol β. Una vez alcanzado el pie de dicho árbol auxiliar, deberemos retomar el análisis del subárbol de γ que pende del nodo de adjunción. Al no conocer en qué nodo de qué árbol elemental se ha producido la adjunción, los pasos $\mathcal{D}_{\mathrm{E}}^{\mathrm{FootPred}}$ predecirán todos los posibles nodos donde esté permitida la adjunción de β. Una vez terminado de analizar todo el subárbol predicho por $\mathcal{D}_{\mathrm{E}}^{\mathrm{FootPred}}$, los pasos $\mathcal{D}_{\mathrm{E}}^{\mathrm{FootComp}}$ retoman el análisis del árbol auxiliar β a partir del pie. Una vez terminado de analizar completamente β, la operación de adjunción concluye con la aplicación de uno de los pasos $\mathcal{D}_{\mathrm{E}}^{\mathrm{AdjComp^1}}$ o $\mathcal{D}_{\mathrm{E}}^{\mathrm{AdjComp^2}}$, que verifican que el subárbol escindido del nodo de adjunción

y el árbol auxiliar han sido correctamente reconstruidos. $\mathcal{D}_{\mathrm{E}}^{\mathrm{AdjComp}^1}$ se aplica en el caso de que el nodo de adjunción forme parte de la espina de un árbol auxiliar, en caso contrario se aplica $\mathcal{D}_{\mathrm{E}}^{\mathrm{AdjComp}^2}$.

5. La propiedad del prefijo válido

Los analizadores sintácticos que satisfacen la *propiedad del prefijo válido* 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. Más formalmente, un analizador sintáctico satisface la propiedad del prefijo válido si al leer la subcadena $a_1 \dots a_k$ de la cadena de entrada $a_1 \dots a_k a_{k+1} \dots a_n$ garantiza que hay una cadena $b_1 \dots b_m$, donde b_i no tiene porque formar parte de la cadena de entrada, tal que $a_1 \dots a_k b_1 \dots b_m$ 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 que deben actuar coordinadamente, 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. La fase descendente debe estar restringida para evitar expansiones que lleven al reconocimiento de prefijos no válidos [6].

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 sea un lenguaje independiente del contexto [11]. Un algoritmo básicamente ascendente (p.ej.: de tipo CYK o de tipo Earley ascendente) puede ir apilando 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 realizar apilamientos para satisfacer las restricciones impuestas por el lenguaje que define el conjunto de caminos. Al tener que coordinar los apilamientos de la fase ascendente y descendente, la transformación directa de un algoritmo sin la propiedad del prefijo válido en otro que sí la posea puede aumentar la complejidad del algoritmo resultante. Para evitarlo, será preciso refinar cuidadosamente dicho algoritmo [5].

Los algoritmos descritos hasta este punto no preservan la propiedad del prefijo válido puesto que al no ser la fase de predicción del pie lo suficientemente restrictiva no se puede verificar que el subárbol pegado al nodo pie se corresponda con la instancia del árbol que ha sido utilizada en la operación de adjunción.

Para obtener un esquema de análisis que se corresponda con un algoritmo del tipo Earley que posea la propiedad del prefijo válido es necesario refinar los ítems incluyendo un nuevo elemento que indique la posición del extremo izquierdo de la frontera del árbol al que se refieren los nodos de cada ítem:

$$[h, A^\gamma \to \delta \bullet \nu, i, j \mid p, q]$$

tal que $\mathbf{R}^\gamma \overset{*}{\Rightarrow} a_{h+1} \dots a_i \delta \nu v$ y además $\delta \overset{*}{\Rightarrow} a_i \dots a_p$ $\mathbf{F}^\gamma$ $a_{q+1} \dots a_j \overset{*}{\Rightarrow} a_i \dots a_j$ si y sólo si $(p, q) \neq (-, -)$ mientras que $\delta \overset{*}{\Rightarrow} a_i \dots a_j$ si y sólo si $(p, q) = (-, -)$.

En consecuencia, un ítem $[A^\gamma \to \delta \bullet \nu, i, j \mid p, q]$ de $\mathbf{P}_{\mathrm{E}_3}$ se corresponde ahora con el conjunto de ítems $[h, A^\gamma \to \delta \bullet \nu, i, j \mid p, q]\ \forall h \in [0, n]$.

Esquema 4 *El sistema de análisis* $\mathbb{IP}_{\mathrm{Earley}}$ *del algoritmo de tipo Earley con la propiedad del prefijo válido, dada una TAG y una cadena de entrada se define:*

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

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

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

$$\mathcal{D}_{\mathrm{Earley}}^{\mathrm{Pred}} = \frac{[h, A^\gamma \to \delta \bullet B^\gamma \nu, i, j \mid p, q]}{[h, B^\gamma \to \bullet v, j, j \mid -, -]}$$

$$\mathcal{D}^{\text{Comp}}_{\text{Earley}} = \frac{\begin{array}{c}[h, A^\gamma \to \delta \bullet B^\gamma \nu, i, k \mid p, q], \\ [h, B^\gamma \to v \bullet, k, j \mid p', q']\end{array}}{[h, A^\gamma \to \delta B^\gamma \bullet \nu, i, j \mid p \cup p', q \cup q']}$$

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

tal que $\beta \in \text{adj}(B^\gamma)$.

$$\mathcal{D}^{\text{FootPred}}_{\text{Earley}} = \frac{\begin{array}{c}[j, \mathbf{F}^\beta \to \bullet \bot, k, k \mid -, -], \\ [h, A^\gamma \to \delta \bullet B^\gamma \nu, i, j \mid p, q]\end{array}}{[h, B^\gamma \to \bullet \delta, k, k \mid -, -]}$$

tal que $\beta \in \text{adj}(B^\gamma)$.

$$\mathcal{D}^{\text{FootComp}}_{\text{Earley}} = \frac{\begin{array}{c}[h, B^\gamma \to \delta \bullet, k, l \mid p, q], \\ [j, \mathbf{F}^\beta \to \bullet \bot, k, k \mid -, -], \\ [h, A^\gamma \to \delta \bullet B^\gamma \nu, i, j \mid p', q']\end{array}}{[j, \mathbf{F}^\beta \to \bot \bullet, k, l \mid k, l]}$$

tal que $\beta \in \text{adj}(B^\gamma)$ *y tanto* $p \cup p'$ *como* $q \cup q'$ *están definidos.*

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

tal que $\beta \in \text{adj}(B^\gamma)$.

$$\mathcal{D}^{\text{AdjComp}^2}_{\text{Earley}} = \frac{\begin{array}{c}[j, \top \to \mathbf{R}^\beta \bullet, j, m \mid k, l], \\ [h, B^\gamma \to \delta \bullet, k, l \mid -, -], \\ [h, A^\gamma \to \delta \bullet B^\gamma \nu, i, j \mid p, q]\end{array}}{[h, A^\gamma \to \delta B^\gamma \bullet \nu, i, m \mid p, q]}$$

tal que $\beta \in \text{adj}(B^\gamma)$.

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

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

La complejidad del paso $\mathcal{D}^{\text{AdjComp}^1}_{\text{Earley}}$ haciendo uso de aplicación parcial es $\mathcal{O}(n^7)$ y coincide con la complejidad del algoritmo descrito por el esquema $\mathbf{P}_{\text{Earley}}$. Los algoritmos descritos anteriormente presentaban una complejidad $\mathcal{O}(n^6)$. El aumento de complejidad es debido al índice adicional incorporado a los ítems. Este índice es necesario para controlar que se están utilizando los árboles correctos al tratar el pie de un árbol auxiliar. En consecuencia, dicho índice sólo es útil en los pasos $\mathcal{D}^{\text{FootPred}}_{\text{Earley}}$ y $\mathcal{D}^{\text{FootComp}}_{\text{Earley}}$ y por tanto los pasos $\mathcal{D}^{\text{AdjComp}^1}_{\text{Earley}}$ y $\mathcal{D}^{\text{AdjComp}^2}_{\text{Earley}}$ pueden ser refinados, dividiéndolos en varios pasos con el fin de generar ítems intermedios carentes de dicho índice. La corrección de la transformación está garantizada por la *propiedad de independencia del contexto de TAG* [10] que establece la independencia de cada operación de adjunción con respecto a cualquier otra operación de adjunción.

Obtenemos de este modo el algoritmo de tipo Earley descrito por Nederhof en [5], que preserva la propiedad del prefijo válido con una complejidad $\mathcal{O}(n^6)$. Debemos definir un tipo adicional de ítem

$$[A^\gamma \to \delta \bullet \nu, i, j \mid p, q]$$

tal que $\delta \overset{*}{\Rightarrow} a_i \ldots a_p \; \mathbf{F}^\gamma \; a_{q+1} \ldots a_j \overset{*}{\Rightarrow} a_i \ldots a_j$ si y sólo si $(p, q) \neq (-, -)$ mientras que $\delta \overset{*}{\Rightarrow} a_i \ldots a_j$ si y sólo si $(p, q) = (-, -)$.

También debemos refinar los pasos que completan la adjunción, de modo que los pasos AdjComp^1 y AdjComp^2 serán equivalentes a la aplicación de un paso AdjComp^0 más un paso $\text{AdjComp}^{1'}$ o $\text{AdjComp}^{2'}$:

$$\mathcal{D}^{\text{AdjComp}^0}_{\text{Earley}} = \frac{\begin{array}{c}[j, \top \to \mathbf{R}^\beta \bullet, j, m \mid k, l], \\ [h, B^\gamma \to \delta \bullet, k, l \mid p, q],\end{array}}{[B^\gamma \to \delta \bullet, j, m \mid p, q]}$$

tal que $\beta \in \text{adj}(B^\gamma)$.

$$\mathcal{D}^{\text{AdjComp}^{1'}}_{\text{Earley}} = \frac{\begin{array}{c}[B^\gamma \to \delta \bullet, j, m \mid p, q], \\ [h, \mathbf{F}^\gamma \to \bot \bullet, p, q \mid p, q], \\ [h, A^\gamma \to \delta \bullet B^\gamma \nu, i, j \mid -, -]\end{array}}{[h, A^\gamma \to \delta B^\gamma \bullet \nu, i, m \mid p, q]}$$

tal que $\beta \in \text{adj}(B^\gamma)$.

$$\mathcal{D}^{\text{AdjComp}^{2'}}_{\text{Earley}} = \frac{\begin{array}{c}[B^\gamma \to \delta \bullet, j, m \mid -, -], \\ [h, A^\gamma \to \delta \bullet B^\gamma \nu, i, j \mid p, q]\end{array}}{[h, A^\gamma \to \delta B^\gamma \bullet \nu, i, m \mid p, q]}$$

tal que $\beta \in adj(B^{\gamma})$.

6. Conclusiones

Se ha descrito un conjunto de algoritmos de análisis sintáctico de TAG que constituyen una línea evolutiva continua que tiene como punto de partida el algoritmo de tipo CYK de Vijay-Shanker y Joshi [9] y como punto de llegada el algoritmo de tipo Earley de Nederhof [5] que preserva la propiedad del prefijo válido con complejidad $\mathcal{O}(n^6)$. Como algoritmos intermedios, se ha descubierto un algoritmo de tipo Earley ascendente y una nueva versión de un algoritmo de tipo Earley sin la propiedad del prefijo válido. Se ha mostrado cómo se pueden transformar unos algoritmos en otros mediante transformaciones simples. En esta línea evolutiva se podrían haber considerado más algoritmos intermedios, pero las restricciones en el espacio disponible han aconsejado considerar solamente los hitos más importantes en dicha evolución.

Actualmente estamos trabajando en la conversión de los algoritmos descritos a los modelos de interpretación tabular presentados en [2] y [3] para autómatas que manejan dos pilas [1].

Referencias

[1] Tilman Becker. A New Automaton Model for TAGs: 2-SA. *Computational Intelligence*, 10(4):422–430, 1994.

[2] Eric de la Clergerie, y Miguel A. Alonso Pardo. A Tabular Interpretation of a Class of 2-Stack Automata. In *Proc. of COLING/ACL'98*, Montreal, Canadá, agosto de 1998. ACL.

[3] Eric de la Clergerie, Miguel A. Alonso Pardo, y David Cabrero Souto. A tabular interpretation of bottom-up automata for TAG. In *Proc. of TAG+, Fourth Workshop on Tree-Adjoining Grammars and Related Frameworks*, Filadelfia, PA, Estados Unidos, agosto de 1998.

[4] Aravind K. Joshi, Leon S. Levy, y M. Takahashi. Tree adjunt grammars. *Journal of Computer and System Sciences*, 10(1):136–162, febrero de 1975.

[5] Mark-Jan Nederhof. Solving the correct-prefix property for TAGs. In T. Becker y H.-V. Krieger, editores, *Proc. of the Fifth Meeting on Mathematics of Language*, páginas 124–130, Schloss Dagstuhl, Saarbruecken, Alemania, agosto de 1997.

[6] Yves Schabes. The valid prefix property and left to right parsing of tree-adjoining grammar. In *Proc. of II International Workshop on Parsing Technologies, IWPT'91*, páginas 21–30, Cancún, Méjico, 1991.

[7] Yves Schabes y Stuart M. Shieber. An alternative conception of tree-adjoining derivation. *Computational Linguistics*, 20(1):91–124, 1994.

[8] Klaas Sikkel. *Parsing Schemata — A Framework for Specification and Analysis of Parsing Algorithms.* Texts in Theoretical Computer Science — An EATCS Series. Springer-Verlag, Berlín/Heidelberg/Nueva York, 1997.

[9] Krishnamurti Vijay-Shanker y Aravind K. Joshi. Some computational properties of tree adjoining grammars. In *23rd Annual Meeting of the Association for Computational Linguistics*, páginas 82–93, Chicago, IL, Estados Unidos, julio de 1985. ACL.

[10] Krishnamurti Vijay-Shanker y David J. Weir. Parsing some constrained grammar formalisms. *Computational Linguistics*, 19(4):591–636, 1994.

[11] Krishnamurti Vijay-Shanker, David J. Weir, y Aravind K. Joshi. Characterizing structural descriptions produced by various grammatical formalisms. In *Proc. of the 25th Annual Meeting of the Association for Computational Linguistics*, páginas 104–111, Buffalo, NY, Estados Unidos, junio de 1988. ACL.