Chapter 1

RELATING TABULAR PARSING ALGORITHMS FOR LIG AND TAG

Miguel A. Alonso
Departamento de Computación, Facultad de Informática, Universidad de La Coruña, Campus de Elviña s/n, 15071 La Coruña, Spain
alonso@udc.es

Éric de la Clergerie
INRIA, Domaine de Voluceau, Rocquecourt, B.P. 105, 78153 Le Chesnay, France
Eric.De_La_Clergerie@inria.fr

Víctor J. Díaz
Departamento de Lenguajes y Sistemas Informáticos, E.T.S. de Ingeniería Informática, Universidad de Sevilla, Av. Reina Mercedes s/n, 41012 Sevilla, Spain
vjdiaz@lsi.us.es

Manuel Vilares
Departamento de Informática, Escuela Superior de Ingeniería Informática, Universidad de Vigo, Campus As Lagoas s/n, 32004 Orense, Spain
vilares@ei.uvigo.es

Abstract Tree Adjoining Grammars (TAG) and Linear Indexed Grammars (LIG) are extensions of Context Free Grammars that generate the class of Tree Adjoining Languages. Taking advantage of this property, and providing a method for translating a TAG into a LIG, we define several parsing algorithms for TAG on the basis of their equivalent LIG parsers. We also explore why some practical optimizations for TAG parsing cannot be applied to the case of LIG.

1. Introduction

Tree Adjoining Grammars (TAG; Joshi and Schabes, 1997) and Linear Indexed Grammars (LIG; Gazdar, 1987) are extensions of Context Free Grammars (CFG). Tree adjoining grammars use trees instead of productions as the primary means of representing structure and seem to be adequate for describing syntactic phenomena occurring in natural language, due to their extended domain of locality and to their ability to factor recursion from the domain of dependencies. Linear indexed grammars associate a list of indices with each non-terminal symbol, with the restriction that the index list of the left hand side non-terminal of each production (the *mother*) can be inherited by at most one right hand side non-terminal (the *dependent child*) while the other lists must have a bounded size.

Several parsing algorithms have been proposed for TAG, ranging from simple bottom-up algorithms to sophisticated extensions of the Earley's algorithm (Vijay-Shanker and Weir, 1993; Joshi and Schabes, 1997; Nederhof, 1999). In (Alonso et al., 1999) we have shown the relationships among them, creating a continuum of parsing algorithms and showing what transformations must be applied to each one in order to obtain the next one in the continuum. For this purpose, we have selected Parsing Schemata (Sikkel, 1997) as the framework for describing parsing algorithms.

In order to improve efficiency, it is usual to translate the source tree adjoining grammar into a linear indexed grammar (Vijay-Shanker and Weir, 1991; Schabes and Shieber, 1994; Vijay-Shanker and Weir, 1993). We have presented in (Alonso et al., 2000a) a set of parsing algorithms for LIG that mimic the parsing strategies for TAG shown in (Alonso et al., 1999), including a strategy that preserves the correct prefix property. However, in (Alonso et al., 2000a) we did not present the relations, for each parsing strategy, between the algorithm for TAG and the algorithm for LIG implementing that strategy. In this chapter we study the relations among the way TAG parsers recognize adjunction and the way LIG parsers transmit information from one index list to another.

2. Tree Adjoining Grammars

Formally, a TAG is a tuple $\mathcal{G} = (V_N, V_T, S, \boldsymbol{I}, \boldsymbol{A})$, where V_N is a finite set of non-terminal symbols, V_T is a finite set of terminal symbols, S is the axiom of the grammar, $\boldsymbol{I}$ is a finite set of *initial trees* and $\boldsymbol{A}$ is a finite set of *auxiliary trees*. $\boldsymbol{I} \cup \boldsymbol{A}$ is the set of *elementary trees*. Internal nodes are labeled by non-terminals and leaf nodes by terminals or the

empty string ε, except for just one leaf per auxiliary tree (the *foot*) which is labeled by the same non-terminal used as the label of its root node. The path in an elementary tree from the root node to the foot node is called the *spine* of the tree. We write $N^\gamma \in \text{spine}(\gamma)$ to denote that a node N^γ belongs to the spine of γ.

New trees are derived by *adjunction*: let γ be a tree containing a node N^γ labeled by A and let β be an auxiliary tree whose root and foot nodes are also labeled by A. Then, the adjunction of β at the *adjunction node* N^γ is obtained by excising the subtree of γ with root N^γ, attaching β to N^γ and attaching the excised subtree to the foot of β. We write $\beta \in \text{adj}(N^\gamma)$ to denote that a tree β may be adjoined at node N^γ of the elementary tree γ. If adjunction is not mandatory at N^γ then $\mathbf{nil} \in \text{adj}(N^\gamma)$ where $\mathbf{nil} \notin \boldsymbol{I} \cup \boldsymbol{A}$ is a dummy symbol. If adjunction is not allowed at N^γ then $\{\mathbf{nil}\} = \text{adj}(N^\gamma)$.

In order to describe the parsing algorithms for TAG, we must be able to represent the partial recognition of elementary trees. Parsing algorithms for context-free grammars usually denote partial recognition of productions by dotted productions. We can extend this approach to the case of tree-based grammars by considering each elementary tree γ as formed by a set of context-free productions $\mathcal{P}(\gamma)$: a node N^γ and its children $N_1^\gamma, \ldots, N_g^\gamma$ are represented by a production $N^\gamma \rightarrow N_1^\gamma \cdots N_g^\gamma$. Thus, the position of the dot in the tree is indicated by the position of the dot in a production in $\mathcal{P}(\gamma)$. The elements of the productions are the nodes of the tree, with $\mathbf{R}^\gamma$ denoting its root and $\mathbf{F}^\gamma$ its possible foot. To simplify the description of parsing algorithms we add a production $\top \rightarrow \mathbf{R}^\gamma$ for each $\gamma \in \boldsymbol{I} \cup \boldsymbol{A}$, and the production $\mathbf{F}^\beta \rightarrow \bot$ for each $\beta \in \boldsymbol{A}$.

Let $\mathcal{P}(\mathcal{G})$ be the union of all $\mathcal{P}(\gamma)$, $\gamma \in \boldsymbol{I} \cup \boldsymbol{A}$. The relation $\overset{*}{\Rightarrow}$ of derivation on $\mathcal{P}(\mathcal{G})$ is defined as the smallest reflexive and transitive relation including the following base cases:

- $\delta' M^\gamma \delta'' \overset{*}{\Rightarrow} \delta' \upsilon \delta''$ if $M^\gamma \rightarrow \upsilon \in \mathcal{P}(\gamma)$ and $\mathbf{nil} \in \text{adj}(M^\gamma)$.
- $\delta' M^\gamma \delta'' \overset{*}{\Rightarrow} \delta' \upsilon_1 \upsilon \upsilon_2 \delta''$ if $M^\gamma \rightarrow \upsilon \in \mathcal{P}(\gamma)$, $\beta \in \text{adj}(M^\gamma)$, and $\mathbf{R}^\beta \overset{*}{\Rightarrow} \upsilon_1 \mathbf{F}^\beta \upsilon_2$.

The language defined by a TAG is the set of strings $w \in V_T$ such that $\mathbf{R}^\alpha \overset{*}{\Rightarrow} w$ with $S = \text{label}(\mathbf{R}^\alpha)$.

We also define additional forms of derivations for TAG. The first two, used to distinguish derivations with and without adjunction at a node M^γ such that $M^\gamma \rightarrow \upsilon$, are defined by $M^\gamma \overset{*}{\Rightarrow}_t \delta$ (resp. $M^\gamma \overset{*}{\Rightarrow}_b \delta$) if $M^\gamma \overset{*}{\Rightarrow} \delta$ (resp. $\upsilon \overset{*}{\Rightarrow} \delta$). The other two are used to denote derivations crossing root nodes of auxiliary trees and derivations crossing foot nodes. They are defined respectively as $\overset{*}{\Rightarrow}_{r=\text{def}} (\Rightarrow_r \cup \overset{*}{\Rightarrow})^*$ and $\overset{*}{\Rightarrow}_{f=\text{def}} (\Rightarrow_f$

$\cup \overset{*}{\Rightarrow})^*$ with the base cases $\delta_1 M^\gamma \delta_2 \Rightarrow_r \delta_1 \mathbf{R}^\beta \delta_2$ and $\delta_1 \mathbf{F}^\beta \delta_2 \Rightarrow_f \delta_1 \upsilon \delta_2$ if $\beta \in \text{adj}(M^\gamma)$ and $M^\gamma \rightarrow \upsilon \in \mathcal{P}(\gamma)$.

3. Linear Indexed Grammars

A linear indexed grammar is a tuple (V_T, V_N, V_I, P, S), where V_T is a finite set of terminals, V_N is a finite set of non-terminals, V_I is a finite set of indices, $S \in V_N$ is the start symbol and P is a finite set of productions. Following (Gazdar, 1987) we consider productions in which at most one index can be pushed on or popped from a list of indices:

$$A_0[\circ\circ\eta] \rightarrow A_1[\,] \cdots A_{d-1}[\,]\ A_d[\circ\circ\eta']\ A_{d+1}[\,]\ \cdots A_m[\,]$$

$$A_0[\,] \rightarrow a$$

where m is the length of the production, $A_j \in V_N$ for each $0 \leq j \leq m$, A_d is the dependent child, $\circ\circ$ is the part of the index list transmitted from the mother to the dependent child, $\eta, \eta' \in V_I \cup \{\varepsilon\}$ and for each production either η or η' or both must be ε, and $a \in V_T \cup \{\varepsilon\}$.

The derivation relation $\Rightarrow$ is defined for LIG by the following cases:

- $\Upsilon A[\zeta\eta]\Upsilon' \Rightarrow \Upsilon\Upsilon_1 A'[\zeta\eta']\Upsilon_2\Upsilon'$ if there exists a production $A[\circ\circ\eta] \rightarrow \Upsilon_1 A'[\circ\circ\eta']\ \Upsilon_2$.
- $\Upsilon A[\,]\Upsilon' \Rightarrow \Upsilon a \Upsilon'$ if there exists a production $A[\,] \rightarrow a$

where $A \in V_N$, $\zeta \in V_I^*$ and $\eta, \eta' \in V_I \cup \{\varepsilon\}$. The reflexive and transitive closure of $\Rightarrow$ is denoted by $\overset{*}{\Rightarrow}$. The language defined by a LIG is the set of strings $w \in V_T^*$ such that $S[\,] \overset{*}{\Rightarrow} w$.

In a derivation step $\Upsilon A[\zeta\eta]\Upsilon' \Rightarrow \Upsilon\Upsilon_1 A'[\zeta\eta']\Upsilon_2\Upsilon'$, we say that $A'[\zeta\eta']$ is the dependent successor of $A[\zeta\eta]$. We define the notion of dependent descendent as the reflexive and transitive closure of dependent successor.

To parse this type of grammars, tabulation techniques with polynomial complexity can be designed based on a property defined in (Vijay-Shanker and Weir, 1993), that we call the *context-freeness property of LIG*, such that if $A[\eta] \overset{*}{\Rightarrow} uB[\,]w$ where $u, w \in V_T^*$, $A, B \in V_N$, $\eta \in V_I \cup \{\varepsilon\}$ and $B[\,]$ is a dependent descendant of $A[\eta]$, then for each $\Upsilon_1, \Upsilon_2 \in (V_N[V_I^*] \cup V_T)^*$ and $\zeta \in V_I^*$ we have $\Upsilon_1 A[\zeta\eta]\Upsilon_2 \overset{*}{\Rightarrow} \Upsilon_1 uB[\zeta]w\Upsilon_2$ Conversely, if $B[\eta]$ is a dependent descendant of $A[\,]$ and $A[\,] \overset{*}{\Rightarrow} uB[\eta]w$ then $\Upsilon_1 A[\zeta]\Upsilon_2 \overset{*}{\Rightarrow} \Upsilon_1 uB[\zeta\eta]w\Upsilon_2$.

3.1 Compiling TAG to LIG

LIG is often used as an intermediate formalism for TAG parsing. Given a TAG $(V_T, V_N, S, \boldsymbol{I}, \boldsymbol{A})$ we can obtain a strongly equivalent LIG

(V_T, V'_N, V_I, S', P), where $V'_N = \{M^{\gamma\,\mathbf{t}} | M^\gamma \in \mathcal{N}\} \cup \{M^{\gamma\,\mathbf{b}} | M^\gamma \in \mathcal{N}\}$ and $V_I = \{M^\gamma | M^\gamma \in \mathcal{N}\}$, $\mathcal{N}$ denoting the set of every node M^γ of every elementary tree $\gamma \in \boldsymbol{I} \cup \boldsymbol{A}$ (Vijay-Shanker and Weir, 1991). The set P is constructed applying the following rules:

1 A production $S'[\circ\circ] \rightarrow \mathbf{R}^{\alpha\,\mathbf{t}}[\circ\circ]$ is generated for each initial tree α having its root labeled by S.

2 A production $M_0^{\gamma\,\mathbf{b}}[\circ\circ] \rightarrow M_1^{\gamma\,\mathbf{t}}[\circ\circ]\ M_2^{\gamma\,\mathbf{t}}[\,] \cdots M_m^{\gamma\,\mathbf{t}}[\,]$ is generated for each $M_0^\gamma \rightarrow M_1^\gamma \cdots M_m^\gamma \in \mathcal{P}(\gamma)$ such that $\gamma \in \boldsymbol{I}$ or $\gamma \in \boldsymbol{A}$ and $\forall_{1 \leq i \leq m} M_i \notin \text{spine}(\gamma)$.

3 A production $M_0^{\beta\,\mathbf{b}}[\circ\circ] \rightarrow M_1^{\beta\,\mathbf{t}}[\,] \cdots M_d^{\beta\,\mathbf{t}}[\circ\circ] \cdots M_m^{\beta\,\mathbf{t}}[\,]$ is generated for each $M_0^\beta \rightarrow M_1^\beta \cdots M_d^\beta \cdots M_m^\beta \in \mathcal{P}(\beta)$ such that $\beta \in \boldsymbol{A}$ and $M_0^\beta, M_d^\beta \in \text{spine}(\beta)$.

4 A production $M^{\gamma\,\mathbf{t}}[\circ\circ] \rightarrow M^{\gamma\,\mathbf{b}}[\circ\circ]$ is generated for each node M^γ such that $\mathbf{nil} \in \text{adj}(M^\gamma)$.

5 A production $M^{\gamma\,\mathbf{t}}[\circ\circ] \rightarrow \mathbf{R}^{\beta\,\mathbf{t}}[\circ\circ M^\gamma]$ is generated for each adjunction node M^γ such that $\beta \in \boldsymbol{A}$ and $\beta \in \text{adj}(M^\gamma)$.

6 A production $\mathbf{F}^{\beta\,\mathbf{b}}[\circ\circ M^\gamma] \rightarrow M^{\gamma\,\mathbf{b}}[\circ\circ]$ is generated for each node M^γ such that $\beta \in \boldsymbol{A}$ and $\beta \in \text{adj}(M^\gamma)$.

7 A production $M^{\gamma\,\mathbf{t}}[\,] \rightarrow a$ is generated for each node M^γ labeled by $a \in V_T \cup \{\varepsilon\}$.

Considering a top-down traversal of trees and adjunctions, the productions generated by rule 1 start the traversal from the root node of initial trees labeled by the axiom of the grammar. Rule 2 productions are slightly artificial productions used to traverse nodes not on the spine of auxiliary trees; indeed, to respect the kind of expected LIG productions, we have chosen the first child as dependent child when there is actually none, as every indices list should be empty. A more natural LIG production (if allowed) would be (2') $M_0^{\gamma\,\mathbf{b}}[\,] \rightarrow M_1^{\gamma\,\mathbf{t}}[\,]\ M_2^{\gamma\,\mathbf{t}}[\,] \cdots M_m^{\gamma\,\mathbf{t}}[\,]$. Rule 3 productions are used to traverse nodes on the spine of auxiliary trees. By means of the set of productions generated by rule 4, we can continue regular traversal when adjunction is not mandatory at a given node, while rule 5 productions suspend the traversal of an elementary tree γ to start the traversal of the auxiliary tree that can be adjoined at node M^γ; the fact that we will have to return to M^γ is recorded by pushing this node on the index list. When the auxiliary tree has been

completely traversed, a rule 6 production pops M^γ from the index list to resume the traversal of γ at this node. Productions generated by rule 7 are in charge of recognizing terminal symbols and the empty string.

Superscripts $\mathbf{t}$ and $\mathbf{b}$ are used to guarantee that at most one auxiliary tree is adjoined at each node. Given an adjunction node M^γ, in a top-down view of adjunction, $M^{\gamma\mathbf{t}}$ corresponds to reaching the node M^γ before adjunction and $M^{\gamma\mathbf{b}}$ corresponds to reaching this node after adjunction. In a bottom-up view of adjunction, $M^{\gamma\mathbf{b}}$ corresponds to reaching the node M^γ before adjunction and $M^{\gamma\mathbf{t}}$ corresponds to reaching this node after adjunction.

4. Bottom-up Parsing Algorithms

The basic bottom-up parsing algorithm for context-free grammars is the one defined by Cocke, Younger and Kasami (Kasami, 1965; Younger, 1967). Extensions have been defined for TAG (Vijay-Shanker and Weir, 1991; Alonso et al., 1999) and LIG (Vijay-Shanker and Weir, 1991; Alonso et al., 2000a). In the case of LIG, grammars are restricted to have at most two non-terminal elements, or one element which must be a terminal, in the right-hand side of each production. This restriction could be considered as the transposition of Chomsky Normal Form to linear indexed grammars. In the case of TAG, nodes in elementary trees can have at most two children.

We will describe parsing algorithms using *Parsing Schemata*, a framework for high-level descriptions of parsers (Sikkel, 1997). A *parsing system* for a grammar G and string $a_1 \cdots a_n$ is a triple $\langle \mathcal{I}, \mathcal{H}, \mathcal{D} \rangle$, with $\mathcal{I}$ a set of *items* which represent intermediate parse results, $\mathcal{H}$ an initial set of items $[a_{j+1}, j, j+1]$, with $0 \leq j < n$, called the *hypothesis* that encodes the sentence $a_1 \cdots a_n$ to be parsed, and $\mathcal{D}$ a set of *deduction steps* that allow new items to be derived from already known items. Deduction steps are of the form $\frac{\eta_1, \ldots, \eta_k}{\xi}\ _{cond}$, meaning that if all antecedents η_i of a deduction step are present and the conditions *cond* are satisfied, then the consequent ξ should be generated by the parser. A set $\mathcal{F} \subseteq \mathcal{I}$ of *final items* represents the recognition of a sentence. A *parsing schema* is a parsing system parameterized by a grammar and a sentence.

4.1 Items

Items used in the tabular interpretation of the CYK-like algorithm for LIG are of the form

$$[A, \eta, i, j \mid B, p, q]$$

where, in general, $A, B \in V_N$, $\eta \in V_I$, $0 \leq i \leq j$ and $i \leq p \leq q \leq j$ denoted $(p, q) \leq (i, j)$. Elements B, η, p and q may be unbound in some cases and represented by $-$.

Each item represents one of the following kinds of derivations:

- $A[\eta] \stackrel{*}{\Rightarrow} a_{i+1} \cdots a_p \; B[\,] \; a_{q+1} \cdots a_j$ iff $(B, p, q) \neq (-, -, -)$, $B[\,]$ is a dependent descendent of $A[\eta]$ and $(p, q) \leq (i, j)$.
- $A[\,] \stackrel{*}{\Rightarrow} a_{i+1} \cdots a_j$ iff $\eta = -$ and $(B, p, q) = (-, -, -)$.

If the index list associated with A is empty then $\eta = -$ and $(B, p, q) = (-, -, -)$, otherwise η is the topmost element of the index list and the part (B, p, q) acts as a logical pointer to other items of the form $[B, \eta', p, q \mid B', p', q']$ from which we can retrieve the second element η' of that list. By following the chains of pointers, we can retrieve the entire index list associated to A. The string $a_1 \cdots a_n$ to be parsed has been successfully recognized if a final item in the set $\mathcal{F} = \{\, [S, -, 0, n \mid -, -, -] \,\}$ has been generated.

These items are like those proposed for the tabulation of right-oriented linear indexed automata (Alonso et al., 2000b; Nederhof, 1998) and for the tabulation of bottom-up 2–stack automata (De la Clergerie et al., 1998). They are slightly different from the items of the form $[A, \eta, i, j \mid B, \eta', p, q]$ proposed by (Vijay-Shanker and Weir, 1991) for their CYK-like algorithm, where the element $\eta' \in V_I$ is useless: the context-freeness property of LIG implies that if $A[\eta] \stackrel{*}{\Rightarrow} a_{i+1} \cdots a_p \; B[\,] \; a_{q+1} \cdots a_j$ then for any η' we have that $A[\eta'\eta] \stackrel{*}{\Rightarrow} a_{i+1} \cdots a_p \; B[\eta'] \; a_{q+1} \cdots a_j$.

In the case of TAG parsing, if we translate the grammar to LIG following the mechanism shown in section 3.1, we obtain items of the form:

$$[N^{\gamma\mathbf{x}}, M^{\gamma'}, i, j \mid M^{\gamma'}, p, q]$$

where $\mathbf{x} \in \{\mathbf{t}, \mathbf{b}\}$, representing one of the following two situations:

- $N^{\gamma} \stackrel{*}{\Rightarrow}_{\mathbf{x}} a_{i+1} \cdots a_p \; \mathbf{F}^{\gamma} \; a_{q+1} \cdots a_j \Rightarrow_f a_{i+1} \cdots a_p \; \upsilon \; a_{q+1} \cdots a_j \stackrel{*}{\Rightarrow}_f a_{i+1} \cdots a_j$, $M^{\gamma'} \rightarrow \upsilon$, and $\gamma \in \mathrm{adj}(M^{\gamma'})$ iff $(M^{\gamma'}, p, q) \neq (-, -, -)$.
- $N^{\gamma} \stackrel{*}{\Rightarrow}_{\mathbf{x}} a_{i+1} \cdots a_j$ iff $(M^{\gamma'}, p, q) = (-, -, -)$.

We can observe that each item represents a state in the parsing process, storing information about the node that we are currently visiting, the part of the input string spanned by this node, and the list of nested adjunctions that have been started but not yet finished. The topmost element $M^{\gamma'}$ of this list is explicitly stored in the item. However, the element $M^{\gamma'}$ is redundant, as an item is valid for any node $M^{\gamma'}$ of an

elementary tree such that γ can be adjoined at $M^{\gamma'}$. Therefore, by discarding redundant information, we can obtain a more compact form of items for TAG parsing:

$$[N^{\gamma\mathbf{x}}, i, j \mid p, q]$$

Final items are those belonging to the set $\mathcal{F} = \{\, [\mathbf{R}^{\alpha\mathbf{t}}, 0, n \mid -, -] \,\}$, with $\alpha \in \boldsymbol{I}$ and $S = \text{label}(\mathbf{R}^{\alpha})$.

4.2 Deduction Steps

The bottom-up parsing process is started by steps recognizing terminal symbols and the empty string:

$$\mathcal{D}^{\text{Scan}}_{\text{CYK-LIG}} = \frac{[a, j, j+1]}{[A, -, j, j+1 \mid -, -, -]} \quad A[\,] \to a \in P$$

$$\mathcal{D}^{\varepsilon}_{\text{CYK-LIG}} = \frac{}{[A, -, j, j \mid -, -, -]} \quad A[\,] \to \varepsilon \in P$$

The other steps are in charge of combining the items corresponding to the body elements in the right-hand side of a production in order to generate the item corresponding to the left-hand side element, propagating bottom-up the information about the index list:

$$\mathcal{D}^{[\circ\circ][\,][\circ\circ]}_{\text{CYK-LIG}} = \frac{\begin{array}{l}[B, -, i, k \mid -, -, -],\\ [C, \eta, k, j \mid D, p, q]\end{array}}{[A, \eta, i, j \mid D, p, q]} \quad A[\circ\circ] \to B[\,]\, C[\circ\circ] \in P$$

$$\mathcal{D}^{[\circ\circ][\circ\circ][\,]}_{\text{CYK-LIG}} = \frac{\begin{array}{l}[B, \eta, i, k \mid D, p, q],\\ [C, -, k, j \mid -, -, -]\end{array}}{[A, \eta, i, j \mid D, p, q]} \quad A[\circ\circ] \to B[\circ\circ]\, C[\,] \in P$$

$$\mathcal{D}^{[\circ\circ][\circ\circ]}_{\text{CYK-LIG}} = \frac{[B, \eta, i, j \mid D, p, q],}{[A, \eta, i, j \mid D, p, q]} \quad A[\circ\circ] \to B[\circ\circ] \in P$$

$$\mathcal{D}^{[\circ\circ\eta][\,][\circ\circ]}_{\text{CYK-LIG}} = \frac{\begin{array}{l}[B, -, i, k \mid -, -, -],\\ [C, \eta', k, j \mid D, p, q]\end{array}}{[A, \eta, i, j \mid C, k, j]} \quad A[\circ\circ\eta] \to B[\,]\, C[\circ\circ] \in P$$

$$\mathcal{D}^{[\circ\circ\eta][\circ\circ][\,]}_{\text{CYK-LIG}} = \frac{\begin{array}{l}[B, \eta', i, k \mid D, p, q],\\ [C, -, k, j \mid -, -, -]\end{array}}{[A, \eta, i, j \mid B, i, k]} \quad A[\circ\circ\eta] \to B[\circ\circ]\, C[\,] \in P$$

$$\mathcal{D}^{[\circ\circ\eta][\circ\circ]}_{\text{CYK-LIG}} = \frac{[B, \eta', i, j \mid D, p, q],}{[A, \eta, i, j \mid B, i, j]} \quad A[\circ\circ\eta] \to B[\circ\circ] \in P$$

$$\mathcal{D}^{[\circ\circ][\ ][\circ\circ\eta]}_{\text{CYK-LIG}} = \frac{\begin{array}{l}[B,-,i,k \mid -,-,-],\\ [C,\eta,k,j \mid D,p,q],\\ [D,\eta',p,q \mid E,r,s]\end{array}}{[A,\eta',i,j \mid E,r,s]} \quad A[\circ\circ] \rightarrow B[\ ]\ C[\circ\circ\eta] \in P$$

$$\mathcal{D}^{[\circ\circ][\circ\circ\eta][\ ]}_{\text{CYK-LIG}} = \frac{\begin{array}{l}[B,\eta,i,k \mid D,p,q],\\ [C,-,k,j \mid -,-,-],\\ [D,\eta',p,q \mid E,r,s]\end{array}}{[A,\eta',i,j \mid E,r,s]} \quad A[\circ\circ] \rightarrow B[\circ\circ\eta]\ C[\ ] \in P$$

$$\mathcal{D}^{[\circ\circ][\circ\circ\eta]}_{\text{CYK-LIG}} = \frac{\begin{array}{l}[B,\eta,i,j \mid D,p,q],\\ [D,\eta',p,q \mid E,r,s]\end{array}}{[A,\eta',i,j \mid E,r,s]} \quad A[\circ\circ] \rightarrow B[\circ\circ\eta] \in P$$

These steps have counterparts in TAG parsing. The steps in charge of starting the TAG parser are the following:

$$\mathcal{D}^{\text{Scan}}_{\text{CYK-TAG}} = \frac{[a,j,j+1]}{[N^{\gamma}\mathbf{b},j,j+1 \mid -,-]} \quad a = \text{label}(N^{\gamma})$$

$$\mathcal{D}^{\varepsilon}_{\text{CYK-TAG}} = \frac{}{[N^{\gamma}\mathbf{b},j,j \mid -,-]} \quad \varepsilon = \text{label}(N^{\gamma})$$

The sets $\mathcal{D}^{[\circ\circ][\ ][\circ\circ]}_{\text{CYK-LIG}}$ and $\mathcal{D}^{[\circ\circ][\circ\circ][\ ]}_{\text{CYK-LIG}}$ correspond to the bottom-up propagation of information through the spine of an auxiliary tree when the right child and the left child, respectively, is placed on the spine:

$$\mathcal{D}^{\text{RightDom}}_{\text{CYK-TAG}} = \frac{\begin{array}{l}[M^{\gamma}\mathbf{t},i,k \mid -,-],\\ [P^{\gamma}\mathbf{t},k,j \mid p,q]\end{array}}{[N^{\gamma}\mathbf{b},i,j \mid p,q]} \quad \begin{array}{l}N^{\gamma} \rightarrow M^{\gamma}P^{\gamma} \in \mathcal{P}(\gamma),\\ P^{\gamma} \in \text{spine}(\gamma)\end{array}$$

$$\mathcal{D}^{\text{LeftDom}}_{\text{CYK-TAG}} = \frac{\begin{array}{l}[M^{\gamma}\mathbf{t},i,k \mid p,q],\\ [P^{\gamma}\mathbf{t},k,j \mid -,-]\end{array}}{[N^{\gamma}\mathbf{b},i,j \mid p,q]} \quad \begin{array}{l}N^{\gamma} \rightarrow M^{\gamma}P^{\gamma} \in \mathcal{P}(\gamma),\\ M^{\gamma} \in \text{spine}(\gamma)\end{array}$$

The set $\mathcal{D}^{[\circ\circ][\circ\circ][\ ]}_{\text{CYK-LIG}}$ also corresponds to the bottom-up propagation of information, in this case for productions not covering the spine of an auxiliary tree:

$$\mathcal{D}^{\text{NoDom}}_{\text{CYK-TAG}} = \frac{\begin{array}{l}[M^{\gamma}\mathbf{t},i,k \mid -,-],\\ [P^{\gamma}\mathbf{t},k,j \mid -,-]\end{array}}{[N^{\gamma}\mathbf{b},i,j \mid -,-]} \quad \begin{array}{l}N^{\gamma} \rightarrow M^{\gamma}P^{\gamma} \in \mathcal{P}(\gamma),\\ N^{\gamma} \notin \text{spine}(\gamma)\end{array}$$

The set $\mathcal{D}^{[\circ\circ][\circ\circ]}_{\text{CYK-LIG}}$ corresponds to the bottom-up propagation of information for nodes with only one child and to the bottom-up traversal

of nodes at which adjunction is not mandatory:

$$\mathcal{D}^{\text{Unary}}_{\text{CYK}-\text{TAG}} = \frac{[M^{\gamma}\mathbf{t}, i, j \mid p, q]}{[N^{\gamma}\mathbf{b}, i, j \mid p, q]} \quad N^{\gamma} \rightarrow M^{\gamma} \in \mathcal{P}(\gamma)$$

$$\mathcal{D}^{\text{NoAdj}}_{\text{CYK}-\text{TAG}} = \frac{[N^{\gamma}\mathbf{b}, i, j \mid p, q]}{[N^{\gamma}\mathbf{t}, i, j \mid p, q]} \ \ \mathbf{nil} \in \text{adj}(N^{\gamma})$$

The set $\mathcal{D}^{[\circ\circ\eta][\circ\circ]}_{\text{CYK}-\text{LIG}}$ corresponds to the bottom-up starting of an adjunction operation from the foot node of an auxiliary tree:

$$\mathcal{D}^{\text{Foot}}_{\text{CYK}-\text{TAG}} = \frac{[N^{\gamma}\mathbf{b}, i, j \mid p, q]}{[\mathbf{F}^{\beta}\mathbf{b}, i, j \mid i, j]} \ \ \beta \in \text{adj}(N^{\gamma})$$

The set $\mathcal{D}^{[\circ\circ][\circ\circ\eta]}_{\text{CYK}-\text{LIG}}$ corresponds to the bottom-up ending of an adjunction operation when the root node of an auxiliary tree is reached:

$$\mathcal{D}^{\text{Adj}}_{\text{CYK}-\text{TAG}} = \frac{[\mathbf{R}^{\beta}\mathbf{t}, i, j \mid p, q],\ [N^{\gamma}\mathbf{b}, p, q \mid r, s]}{[N^{\gamma}\mathbf{t}, i, j \mid r, s]} \ \ \beta \in \text{adj}(N^{\gamma})$$

The space complexity of these bottom-up parsers with respect to the length n of the input string is $\mathcal{O}(n^4)$, as each item stores four positions of the input string. The time complexity is $\mathcal{O}(n^6)$ and is given by the deduction steps in $\mathcal{D}^{[\circ\circ][\ ][\circ\circ\eta]}_{\text{CYK}-\text{LIG}}$ and $\mathcal{D}^{[\circ\circ][\circ\circ\eta][\ ]}_{\text{CYK}-\text{LIG}}$. Although these steps involve 7 positions of the input string, each step can be decomposed, by partial application, into a set of deduction steps involving at most 6 positions. As an example, the application of a step in $\mathcal{D}^{[\circ\circ][\ ][\circ\circ\eta]}_{\text{CYK}-\text{LIG}}$ can be performed by combining its second and third items in $\mathcal{O}(n^6)$ complexity:

$$\frac{\begin{array}{c}[C, \eta, k, j \mid D, p, q], \\ [D, \eta', p, q \mid E, r, s]\end{array}}{\langle C, \eta', k, j, E, r, s\rangle}$$

At this point, positions p and q are discarded, being only useful when combining the second and third items. The resulting element $\langle C, \eta', k, j, E, r, s\rangle$ may be combined with the first item, in $\mathcal{O}(n^5)$ complexity, in order to obtain the consequent item of the original step:

$$\frac{\begin{array}{c}\langle C, \eta', k, j, E, r, s\rangle, \\ [B, -, i, k \mid -, -, -]\end{array}}{[A, \eta', i, j \mid E, r, s]} \ \ A[\circ\circ] \rightarrow B[\ ]\ C[\circ\circ\eta] \in P$$

It is worth noting that this rule transformation may be related to the grammar transformation of production $A[\circ\circ] \rightarrow B[\ ]\ C[\circ\circ\eta]$ into the two equivalent productions $A[\circ\circ] \rightarrow B[\ ]\ C'[\circ\circ]$ and $C'[\circ\circ] \rightarrow C[\circ\circ\eta]$, where C' is some fresh non-terminal.

5. Earley-like Parsing Algorithms

Earley-like parsers overcome the limitation of binary branching imposed by CYK-like parsing algorithms and incorporate top-down prediction in order to reduce the number of deduced items.

5.1 Items

To overcome the limitation imposed by CYK-like algorithms, we introduce dotted productions into items. Thus, we can distinguish the part of a production already processed from the unprocessed part. In the case of LIG, we write $\underline{A}$ as a shorthand for $A[\circ\circ\eta]$, $A[\circ\circ]$ and $A[\,]$ when the specification of the list of indices is not relevant in the context. Therefore, LIG items are now of the form:

$$[\underline{A} \rightarrow \Upsilon_1 \bullet \Upsilon_2, \eta, i, j \mid B, p, q]$$

and they represent one of the following two cases:

- $A[\eta] \Rightarrow \Upsilon_1\Upsilon_2 \stackrel{*}{\Rightarrow} a_{i+1}\cdots a_p\ B[\,]\ a_{q+1}\cdots a_j\ \Upsilon_2$ iff $(B,p,q) \neq (-,-,-)$, $B[\,]$ is a dependent descendant of $A[\eta]$ and $(p,q) \leq (i,j)$.
- $\Upsilon_1 \stackrel{*}{\Rightarrow} a_{i+1}\cdots a_j$ iff $\eta = -$ and $(B,p,q) = (-,-,-)$. If the dependent child is in Υ_1 then the list of indices associated with $\underline{A}$ and with the dependent child must be empty.

The recognition of the input string is indicated by the presence of final items in the set $\mathcal{F} = \{\, [\underline{S} \rightarrow \Upsilon\bullet, -, 0, n \mid -,-,-] \,\}$.

In the case of TAG parsing, if we translate the grammar to LIG following the mechanism shown in section 3.1, we obtain items of the form:

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

representing the following cases:

- $\delta \stackrel{*}{\Rightarrow} a_{i+1}\cdots a_p\ \mathbf{F}^\gamma\ a_{q+1}\cdots a_j \Rightarrow_f a_{i+1}\cdots a_p\ \upsilon\ a_{q+1}\cdots a_j \stackrel{*}{\Rightarrow}_f a_{i+1}\cdots a_j$, $M^{\gamma'} \rightarrow \upsilon$, and $\gamma \in \mathrm{adj}(M^{\gamma'})$ iff $(M^{\gamma'},p,q) \neq (-,-,-)$.
- $\delta \stackrel{*}{\Rightarrow} a_{i+1}\cdots a_j$ iff $(M^{\gamma'},p,q) = (-,-,-)$.

As mentioned for the CYK-like algorithm, the element $M^{\gamma'}$ is redundant, allowing a more compact form of items for TAG parsing:

$$[N^\gamma \rightarrow \delta \bullet \nu, i, j \mid p, q]$$

The set of final items is $\mathcal{F} = \{\, [\mathbf{R}^\alpha \rightarrow \delta\bullet, 0, n \mid -,-] \,\}$, with $\alpha \in \boldsymbol{I}$ and $S = \mathrm{label}(\mathbf{R}^\alpha)$.

5.2 Deduction Steps

The CYK-like parsing algorithm for LIG does not take into account whether the part of the input string recognized by each grammar element can be derived from S, the axiom of the grammar. Earley-like algorithms limit the number of deduction steps that can be applied at each point by predicting productions which are candidates to be part of a derivation starting from the grammar axiom. As a first approach, we consider that prediction is performed by taking into account only the context-free skeleton. The resulting parsing system has the following characteristics:

- A set of Init deduction steps is in charge of starting the top-down prediction by considering only the productions with S as left-hand side.
- In any state of the parsing process, a set of Pred deduction steps generates only those items involving productions with a relevant context-free skeleton.

The following list specifies the sets of deduction steps for an Earley-like parsing algorithm for linear indexed grammars:

$$\mathcal{D}^{\text{Init}}_{\text{E-LIG}} = \frac{}{[\underline{S} \rightarrow \bullet \Upsilon, -, 0, 0 \mid -, -, -]}$$

$$\mathcal{D}^{\text{Scan}}_{\text{E-LIG}} = \frac{[A[\,] \rightarrow \bullet a, -, j, j \mid -, -, -], \; [a, j, j+1]}{[A[\,] \rightarrow a \bullet, -, j, j+1 \mid -, -, -]}$$

$$\mathcal{D}^{\varepsilon}_{\text{E-LIG}} = \frac{[A[\,] \rightarrow \bullet \varepsilon, -, j, j \mid -, -, -]}{[A[\,] \rightarrow \varepsilon \bullet, -, j, j \mid -, -, -]}$$

$$\mathcal{D}^{\text{Pred}}_{\text{E-LIG}} = \frac{[\underline{A} \rightarrow \Upsilon_1 \bullet \underline{B}\ \Upsilon_2, \gamma, i, j \mid C, p, q]}{[\underline{B} \rightarrow \bullet \Upsilon_3, -, j, j \mid -, -, -]}$$

$$\mathcal{D}^{\text{Comp}[\,]}_{\text{E-LIG}} = \frac{[\underline{A} \rightarrow \Upsilon_1 \bullet B[\,]\ \Upsilon_2, \gamma, i, k \mid C, p, q], \; [\underline{B} \rightarrow \Upsilon_3 \bullet, -, k, j \mid -, -, -]}{[\underline{A} \rightarrow \Upsilon_1\ B[\,] \bullet \Upsilon_2, \gamma, i, j \mid C, p, q]}$$

$$\mathcal{D}^{\text{Comp}[\circ\circ\eta][\circ\circ]}_{\text{E-LIG}} = \frac{[A[\circ\circ\eta] \rightarrow \Upsilon_1 \bullet B[\circ\circ]\ \Upsilon_2, -, i, k \mid -, -, -], \; [\underline{B} \rightarrow \Upsilon_3 \bullet, \eta', k, j \mid C, p, q]}{[A[\circ\circ\eta] \rightarrow \Upsilon_1\ B[\circ\circ] \bullet \Upsilon_2, \eta, i, j \mid B, k, j]}$$

$$\mathcal{D}^{\text{Comp}[\circ\circ][\circ\circ]}_{\text{E-LIG}} = \frac{[A[\circ\circ] \rightarrow \Upsilon_1 \bullet B[\circ\circ]\ \Upsilon_2, -, i, k \mid -, -, -], \; [\underline{B} \rightarrow \Upsilon_3 \bullet, \eta', k, j \mid C, p, q]}{[A[\circ\circ] \rightarrow \Upsilon_1\ B[\circ\circ] \bullet \Upsilon_2, \eta', i, j \mid C, p, q]}$$

Table 1.1. Productions proposed by Schabes and Shieber

Schabes & Shieber Production	*Equivalent Production*
$\mathbf{b}[\circ\circ N^\gamma] \rightarrow \mathbf{t}[N_1^\gamma]\cdots\mathbf{t}[\circ\circ N_s^\gamma]\cdots\mathbf{t}[N_m^\gamma]$	$N^{\gamma\,\mathbf{b}}[\circ\circ] \rightarrow N_1^{\gamma\,\mathbf{t}}[\,]\cdots N_s^{\gamma\,\mathbf{t}}[\circ\circ]\cdots N_m^{\gamma\,\mathbf{t}}[\,]$
$\mathbf{b}[N^\gamma] \rightarrow \mathbf{t}[N_1^\gamma]\cdots\mathbf{t}[N_m^\gamma]$	$N^{\gamma\,\mathbf{b}}[\,] \rightarrow N_1^{\gamma\,\mathbf{t}}[\,]\cdots N_m^{\gamma\,\mathbf{t}}[\,]$
$\mathbf{t}[\circ\circ N^\gamma] \rightarrow \mathbf{b}[\circ\circ N^\gamma]$	$N^{\gamma\,\mathbf{t}}[\circ\circ] \rightarrow N^{\gamma\,\mathbf{b}}[\circ\circ]$
$\mathbf{t}[\circ\circ N^\gamma] \rightarrow \mathbf{t}[\circ\circ N^\gamma \mathbf{R}^\beta]$	$N^{\gamma\,\mathbf{t}}[\circ\circ] \rightarrow \mathbf{R}^{\beta\,\mathbf{t}}[\circ\circ N^\gamma]$
$\mathbf{b}[\circ\circ N^\gamma] \rightarrow \mathbf{t}[\circ\circ N^\gamma \mathbf{R}^\beta]$	$N^{\gamma\,\mathbf{b}}[\circ\circ] \rightarrow \mathbf{R}^{\beta\,\mathbf{t}}[\circ\circ N^\gamma]$
$\mathbf{b}[\circ\circ N^\gamma \mathbf{F}^\beta] \rightarrow \mathbf{b}[\circ\circ N^\gamma]$	$\mathbf{F}^{\beta\,\mathbf{b}}[\circ\circ N^\gamma] \rightarrow N^{\gamma\,\mathbf{b}}[\circ\circ]$
$\mathbf{t}[N^\gamma] \rightarrow \mathbf{t}[\mathbf{R}^\alpha]$	$N^{\gamma\,\mathbf{t}}[\,] \rightarrow \mathbf{R}^{\alpha\,\mathbf{t}}[\,]$

$$\mathcal{D}^{\text{Comp}[\circ\circ][\circ\circ\eta]}_{\text{E}-\text{LIG}} = \frac{\begin{array}{l}[A[\circ\circ] \rightarrow \Upsilon_1 \bullet B[\circ\circ\eta]\ \Upsilon_2\ , -, i, k \mid -, -, -],\\ [\underline{B} \rightarrow \Upsilon_3 \bullet\ , \eta, k, j \mid C, p, q],\\ [\underline{C} \rightarrow \Upsilon_4 \bullet, \eta', p, q \mid D, r, s]\end{array}}{[A[\circ\circ] \rightarrow \Upsilon_1\ B[\circ\circ\eta] \bullet \Upsilon_2, \eta', i, j \mid D, r, s]}$$

The resulting algorithm, which has a space complexity $\mathcal{O}(n^4)$ and a time complexity $\mathcal{O}(n^6)$, is very close to the Earley-like algorithm described in (Schabes and Shieber, 1994) although the latter can only be applied to a specific class of linear indexed grammars obtained from tree adjoining grammars. However, both algorithms share an important feature: they are weakly predictive as they do not consider the contents of the index lists when predictive steps are applied. At first glance, the algorithm proposed in (Schabes and Shieber, 1994) consults the element on the top of the index list during prediction, but a deeper study of the behavior of the algorithm makes it clear that this is not true, due to the fact that the context-free skeleton of the elementary trees of a TAG is stored in the index lists, reducing the non-terminal set of the resulting LIG to $\{\mathbf{t}, \mathbf{b}\}$. In table 1.1, an equivalent set of productions with a richer non-terminal set is given. If we consider these productions, there is no consultation of the top of the index lists in the application of predictive steps.

From this parsing algorithm for LIG, we can derive an Earley-like parsing algorithm for TAG, similar to the one described in (Schabes, 1991), that applies prediction with respect to the structure of elementary trees, but not with respect to the lists of pending adjunctions. In the case of TAG, parsing begins by creating the item corresponding to a production that has the root of an initial tree as left-hand side and the dot in the leftmost position of the right-hand side:

$$\mathcal{D}^{\text{Init}}_{\text{E}-\text{TAG}} = \frac{}{[\top \rightarrow \bullet \mathbf{R}^\alpha, 0, 0 \mid -, -]} \quad \alpha \in \boldsymbol{I},\ S = \text{label}(\gamma)$$

Terminal symbols and the empty string are recognized by the following deduction steps:

$$\mathcal{D}^{\text{Scan}}_{\text{E}-\text{TAG}} = \frac{\begin{array}{c}[N^\gamma \to \delta \bullet M^\gamma \nu, i, j \mid p, q],\\ [a, j, j+1]\end{array}}{[N^\gamma \to \delta M^\gamma \bullet \nu, i, j+1 \mid p, q]} \quad a = \text{label}(M^\gamma)$$

$$\mathcal{D}^{\varepsilon}_{\text{E}-\text{TAG}} = \frac{[N^\gamma \to \delta \bullet M^\gamma \nu, i, j \mid p, q],}{[N^\gamma \to \delta M^\gamma \bullet \nu, i, j \mid p, q]} \quad \varepsilon = \text{label}(M^\gamma)$$

The set $\mathcal{D}^{\text{Pred}}_{\text{E}-\text{LIG}}$ of deduction steps has its counterpart in several different steps, one for each kind of predictions that we can perform on TAG parsing, namely:

- Prediction of a child node:

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

- Prediction of the adjunction on a node of an elementary tree γ, starting the traversal of an auxiliary tree β:

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

- Prediction, when the foot node of β is reached, of some subtree of γ to be attached to this foot node and traversed (after suspension of the traversal of β). At this point, no information is available about the adjunction node, so all possible nodes are predicted:

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

The counterpart of $\mathcal{D}^{\text{Comp}[\,]}_{\text{E}-\text{LIG}}$ corresponds to a bottom-up traversal of nodes not on the spine of an auxiliary tree:

$$\mathcal{D}^{\text{CompNoSpine}}_{\text{E}-\text{TAG}} = \frac{\begin{array}{c}[N^\gamma \to \delta \bullet M^\gamma \nu, i, k \mid p, q],\\ [M^\gamma \to \upsilon \bullet, k, j \mid -, -]\end{array}}{[N^\gamma \to \delta M^\gamma \bullet \nu, i, j \mid p, q]} \quad \begin{array}{l} M^\gamma \notin \text{spine}(\gamma)\\ \textbf{nil} \in \text{adj}(M^\gamma)\end{array}$$

whereas the counterpart of $\mathcal{D}^{\text{Comp}[\circ\circ][\circ\circ]}_{\text{E}-\text{LIG}}$ is in charge of the bottom-up traversal of nodes on the spine of an auxiliary tree:

$$\mathcal{D}^{\text{CompSpine}}_{\text{E}-\text{TAG}} = \frac{\begin{array}{c}[N^\gamma \to \delta \bullet M^\gamma \nu, i, k \mid -, -],\\ [M^\gamma \to \upsilon \bullet, k, j \mid p, q]\end{array}}{[N^\gamma \to \delta M^\gamma \bullet \nu, i, j \mid p, q]} \quad \begin{array}{l} M^\gamma \in \text{spine}(\gamma)\\ \textbf{nil} \in \text{adj}(M^\gamma)\end{array}$$

Upon completion of the traversal of a subtree predicted at the foot node $\mathbf{F}^\beta$, a step in the counterpart of $\mathcal{D}_{\text{E}-\text{LIG}}^{\text{Comp}[\circ\circ\eta][\circ\circ]}$ resumes the traversal of auxiliary tree β at $\mathbf{F}^\beta$:

$$\mathcal{D}_{\text{E}-\text{TAG}}^{\text{FootComp}} = \frac{\begin{array}{l}[\mathbf{F}^\beta \rightarrow \bullet\bot, k, k \mid -, -],\\ [M^\gamma \rightarrow \upsilon\bullet, k, j \mid p, q]\end{array}}{[\mathbf{F}^\beta \rightarrow \bot\bullet, k, j \mid k, j]} \quad \beta \in \text{adj}(M^\gamma)$$

Upon completion of the traversal of β, a deduction step in the counterpart of $\mathcal{D}_{\text{E}-\text{LIG}}^{\text{Comp}[\circ\circ][\circ\circ\eta]}$ checks if the subtree attached to the foot of β corresponds to the one rooted at the adjunction node M^γ, when M^γ is placed on a spine:

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

If the adjunction node M^γ is not placed on a spine, then the completion of an adjunction is equivalent to the consecutive application of the counterparts of $\mathcal{D}_{\text{E}-\text{LIG}}^{\text{Comp}[\circ\circ][\circ\circ\eta]}$ and $\mathcal{D}_{\text{E}-\text{LIG}}^{\text{Comp}[\,]}$:

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

The space complexity of these parsing algorithms for LIG and TAG remains $\mathcal{O}(n^4)$ and the time complexity remains $\mathcal{O}(n^6)$.

6. Earley-like Parsing Algorithms Preserving the Correct Prefix Property

Parsers satisfying the *correct prefix property* (CPP) guarantee that, as they read the input string from left to right, the substrings read so far are valid prefixes of the language defined by the grammar. More formally, a parser satisfies the correct prefix property if, for any substring $a_1 \cdots a_k$ read from the input string $a_1 \cdots a_k a_{k+1} \cdots a_n$, it guarantees that there exists a string of tokens $b_1 \cdots b_m$, where b_i need not be part of the input string, such that $a_1 \cdots a_k b_1 \cdots b_m$ is a valid string of the language.

To maintain the correct prefix property, a parser must recognize all possible derived trees in prefix form. In order to do that, two different phases must work coordinately: a top-down phase that expands the children of each node visited and a bottom-up phase grouping the children nodes to indicate the recognition of the parent node (Schabes, 1991).

6.1 Items

A parser for LIG that preserves the correct prefix property must check that each predicted element $A[\zeta]$ satisfies $S[\,] \stackrel{*}{\Rightarrow} w'\, A[\zeta]\, \Upsilon$ where w' is a prefix of the input string w. To obtain a CPP Earley-like parser, we need to modify the Pred steps in order to predict information about the index lists. As a consequence, items must be also modified, introducing a new element that allows us to track the contents of the predicted index lists. The items are now of the form

$$[E, h \mid \underline{A} \rightarrow \Upsilon_1 \bullet \Upsilon_2, \eta, i, j \mid B, p, q]$$

and they represent one of the following kinds of derivations:

- $S[\,] \stackrel{*}{\Rightarrow} a_1 \cdots a_h\, E[\zeta]\, \Upsilon_4 \stackrel{*}{\Rightarrow} a_1 \cdots a_h \cdots a_i\, A[\zeta\eta]\, \Upsilon_3 \Upsilon_4 \stackrel{*}{\Rightarrow} a_1 \cdots a_h \cdots a_i \cdots a_p\, B[\zeta]\, a_{q+1} \cdots a_j\, \Upsilon_2 \Upsilon_3 \Upsilon_4$ if and only if $(B, p, q) \neq (-, -, -)$, $A[\zeta\eta]$ is a dependent descendent of $E[\zeta]$, and $B[\zeta]$ is a dependent descendent of $A[\zeta\eta]$. Such a derivation corresponds to the completion of the dependent child of a production having the non-terminal A with a non empty index list as left-hand side.
- $S[\,] \stackrel{*}{\Rightarrow} a_1 \cdots a_h\, E[\zeta]\, \Upsilon_4 \stackrel{*}{\Rightarrow} a_1 \cdots a_h \cdots a_i\ \ A[\zeta\eta]\ \ \Upsilon_3 \Upsilon_4 \stackrel{*}{\Rightarrow} a_1 \cdots a_h \cdots a_i \cdots a_j\ \ \Upsilon_2 \Upsilon_3 \Upsilon_4$ iff $(E, h) \neq (-, -)$, $(B, p, q) = (-, -, -)$, $A[\zeta\eta]$ is a dependent descendant of $E[\zeta]$, Υ_1 does not contain the descendent child of $A[\zeta\eta]$, and $(p, q) \leq (i, j)$. Such a derivation corresponds to prediction of the non-terminal A with a non-empty index list.
- $S[\,] \stackrel{*}{\Rightarrow} a_1 \cdots a_i A[\,]\Upsilon_4 \stackrel{*}{\Rightarrow} a_1 \cdots a_i \cdots a_j\, \Upsilon_2 \Upsilon_4$ iff $(E, h) = (-, -)$, $\eta = -$, and $(B, p, q) = (-, -, -)$. If Υ_1 includes the dependent child of $A[\,]$ then the index list associated with that dependent child is empty. Such a derivation corresponds to the prediction or completion of an element $A[\,]$.

The new items are refinements of the items in the Earley-like parser without the CPP: the element η is used to store the top of the predicted index list and the pair (E, h) allows us to track the item involved in the prediction. At first glance, we could suppose that, in order to track this item, it would be necessary to store (E, η', h, k). However, due to the context-freeness property of LIG, the index η' may be discarded as the derivation must be valid independently of the rest of the index list. The position k may be discarded because every predicted item in the parsing system is of the form $[\underline{A} \rightarrow \bullet\Upsilon, h, h \mid B, p, q]$ and therefore $h = k$.

The recognition of the input string is indicated by the generation of items in the set of final items $\mathcal{F} = \{\, [-, - \mid \underline{S} \rightarrow \Upsilon\bullet, -, 0, n \mid -, -, -] \,\}$.

In the case of TAG parsing, if we translate the grammar to LIG as shown in section 3.1, we obtain items of the form:

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

representing the following cases, where α denotes an initial tree with $\text{label}(\mathbf{R}^{\alpha}) = S$:

- $\mathbf{R}^{\alpha} \overset{*}{\Rightarrow}_r a_1 \cdots a_h M^{\gamma'} \nu'$, $\mathbf{R}^{\gamma} \overset{*}{\Rightarrow} a_{h+1} \cdots a_i \delta \nu \upsilon'$, $\delta \overset{*}{\Rightarrow} a_{i+1} \cdots a_p$ $\mathbf{F}^{\gamma}$ $a_{q+1} \cdots a_j \Rightarrow_f a_{i+1} \cdots a_p$ υ $a_{q+1} \cdots a_j \overset{*}{\Rightarrow}_f a_{i+1} \cdots a_j$, $M^{\gamma'} \rightarrow \upsilon$, and $\gamma \in \text{adj}(M^{\gamma'})$ iff $(M^{\gamma'}, h) \neq (-,-)$ and $(p,q) \neq (-,-)$.
- $\mathbf{R}^{\alpha} \overset{*}{\Rightarrow}_r a_1 \cdots a_h M^{\gamma'} \nu'$, $\mathbf{R}^{\gamma} \overset{*}{\Rightarrow} a_{h+1} \cdots a_i \delta \nu \upsilon$, $\delta \overset{*}{\Rightarrow} a_{i+1} \cdots a_j$, and $\gamma \in \text{adj}(M^{\gamma'})$ iff if $(M^{\gamma'}, h) \neq (-,-)$ and $(p,q) = (-,-)$.
- $\mathbf{R}^{\alpha} \overset{*}{\Rightarrow}_r a_1 \cdots a_i \delta \nu \nu'$, $\delta \overset{*}{\Rightarrow} a_{i+1} \cdots a_j$, and N^{γ} not on a spine iff $(M^{\gamma'}, h) = (-,-)$ and $(p,q) = (-,-)$.

When bound, $M^{\gamma'}$ and h indicate, respectively, the node at which γ has been adjoined and the position of the input string where this adjunction was started. Note that h is also the left-most position of the frontier of γ. Once again, the element $M^{\gamma'}$ is redundant, allowing a more compact form of items for TAG parsing:

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

The set of final items is $\mathcal{F} = \{\, [- \mid \mathbf{R}^{\alpha} \rightarrow \delta\bullet, 0, n \mid -,-] \,\}$, with $\alpha \in \boldsymbol{I}$ and $S = \text{label}(\mathbf{R}^{\alpha})$.

6.2 Deduction Steps

With respect to deduction steps for LIG, the completion steps must be adapted to the new form of the items in order to manipulate the new components E and h, and the prediction steps must be refined to take into account the different kinds of productions:

$$\mathcal{D}^{\text{Init}}_{\text{Earley-LIG}} = \frac{}{[-,- \mid \underline{S} \rightarrow \bullet\Upsilon, -, 0, 0 \mid -,-,-]}$$

$$\mathcal{D}^{\text{Scan}}_{\text{Earley-LIG}} = \frac{[-,- \mid A[\,] \rightarrow \bullet a, -, j, j \mid -,-,-], \; [a, j, j+1]}{[-,- \mid A[\,] \rightarrow a\bullet, -, j, j+1 \mid -,-,-]}$$

$$\mathcal{D}^{\varepsilon}_{\text{Earley-LIG}} = \frac{[-,- \mid A[\,] \rightarrow \bullet\varepsilon, -, j, j \mid -,-,-]}{[-,- \mid A[\,] \rightarrow \varepsilon\bullet, -, j, j \mid -,-,-]}$$

$$\mathcal{D}^{\text{Pred}[\,]}_{\text{Earley}-\text{LIG}} = \frac{[E,h \mid \underline{A} \rightarrow \Upsilon_1 \bullet B[\,]\ \Upsilon_2, \eta, i, j \mid C, p, q]}{[-,- \mid \underline{B} \rightarrow \bullet \Upsilon_3, -, j, j \mid -, -, -]} \quad \begin{array}{l} \underline{B} = B[\circ\circ] \\ \text{or } \underline{B} = B[\,] \end{array}$$

$$\mathcal{D}^{\text{Pred}[\circ\circ\eta][\circ\circ]}_{\text{Earley}-\text{LIG}} = \frac{\begin{array}{l}[E,h \mid A[\circ\circ\eta] \rightarrow \Upsilon_1 \bullet B[\circ\circ]\ \Upsilon_2, \eta, i, j \mid -, -, -], \\ [M,m \mid \underline{E} \rightarrow \bullet \Upsilon_3, \eta', h, h \mid -, -, -]\end{array}}{[M,m \mid \underline{B} \rightarrow \bullet \Upsilon_4, \eta', j, j \mid -, -, -]}$$

such that $\underline{B} = B[\,]$ iff $\eta' = -$, while $\underline{B} = B[\circ\circ]$ or $\underline{B} = B[\circ\circ\eta']$ iff $\eta' \neq -$.

$$\mathcal{D}^{\text{Pred}[\circ\circ][\circ\circ]}_{\text{Earley}-\text{LIG}} = \frac{[E,h \mid A[\circ\circ] \rightarrow \Upsilon_1 \bullet B[\circ\circ]\ \Upsilon_2, \eta, i, j \mid -, -, -]}{[E,h \mid \underline{B} \rightarrow \bullet \Upsilon_3, \eta, j, j \mid -, -, -]}$$

such that $\underline{B} = B[\,]$ iff $\eta = -$, while $\underline{B} = B[\circ\circ]$ or $\underline{B} = B[\circ\circ\eta]$ iff $\eta \neq -$.

$$\mathcal{D}^{\text{Pred}[\circ\circ][\circ\circ\eta]}_{\text{Earley}-\text{LIG}} = \frac{[E,h \mid A[\circ\circ] \rightarrow \Upsilon_1 \bullet B[\circ\circ\eta]\ \Upsilon_2, \eta', i, j \mid -, -, -]}{[A,i \mid \underline{B} \rightarrow \bullet \Upsilon_3, \eta, j, j \mid -, -, -]}$$

such that $\underline{B} = B[\circ\circ]$ or $\underline{B} = B[\circ\circ\eta]$.

$$\mathcal{D}^{\text{Comp}[\,]}_{\text{Earley}-\text{LIG}} = \frac{\begin{array}{l}[E,h \mid \underline{A} \rightarrow \Upsilon_1 \bullet B[\,]\ \Upsilon_2, \eta, i, j \mid C, p, q], \\ [-,- \mid \underline{B} \rightarrow \Upsilon_3 \bullet, -, j, k \mid -, -, -]\end{array}}{[E,h \mid \underline{A} \rightarrow \Upsilon_1\ B[\,] \bullet \Upsilon_2, \eta, i, k \mid C, p, q]}$$

$$\mathcal{D}^{\text{Comp}[\circ\circ\eta][\circ\circ]}_{\text{Earley}-\text{LIG}} = \frac{\begin{array}{l}[E,h \mid A[\circ\circ\eta] \rightarrow \Upsilon_1 \bullet B[\circ\circ]\ \Upsilon_2, \eta, i, j \mid -, -, -], \\ [M,m \mid \underline{E} \rightarrow \bullet \Upsilon_3, \eta', h, h \mid -, -, -], \\ [M,m \mid \underline{B} \rightarrow \Upsilon_4 \bullet, \eta', j, k \mid C, p, q]\end{array}}{[E,h \mid A[\circ\circ\eta] \rightarrow \Upsilon_1\ B[\circ\circ] \bullet \Upsilon_2, \eta, i, k \mid B, j, k]}$$

$$\mathcal{D}^{\text{Comp}[\circ\circ][\circ\circ]}_{\text{Earley}-\text{LIG}} = \frac{\begin{array}{l}[E,h \mid A[\circ\circ] \rightarrow \Upsilon_1 \bullet B[\circ\circ]\ \Upsilon_2, \eta, i, j \mid -, -, -], \\ [E,h \mid \underline{B} \rightarrow \Upsilon_3 \bullet, \eta, j, k \mid C, p, q]\end{array}}{[E,h \mid A[\circ\circ] \rightarrow \Upsilon_1\ B[\circ\circ] \bullet \Upsilon_2, \eta, i, k \mid C, p, q]}$$

$$\mathcal{D}^{\text{Comp}[\circ\circ][\circ\circ\eta]}_{\text{Earley}-\text{LIG}} = \frac{\begin{array}{l}[E,h \mid A[\circ\circ] \rightarrow \Upsilon_1 \bullet B[\circ\circ\eta]\ \Upsilon_2\ , \eta', i, j \mid -, -, -], \\ [A,i \mid \underline{B} \rightarrow \Upsilon_3 \bullet\ , \eta, j, k \mid C, p, q], \\ [E,h \mid \underline{C} \rightarrow \Upsilon_4 \bullet, \eta', p, q \mid D, r, s]\end{array}}{[E,h \mid A[\circ\circ] \rightarrow \Upsilon_1\ B[\circ\circ\eta] \bullet \Upsilon_2, \eta', i, k \mid D, r, s]}$$

The space complexity of this algorithm with respect to the length n of the input string is $\mathcal{O}(n^5)$, as each item stores five positions of the input string. The time complexity is $\mathcal{O}(n^8)$ due to steps in the set $\mathcal{D}^{\text{Comp}[\circ\circ][\circ\circ\eta]}_{\text{Earley}}$. To reduce the time complexity, we use a technique, inspired by the work of (Nederhof, 1999), and similar to the one used in (De la Clergerie and Alonso, 1998; Alonso et al., 2000b) to reduce the

complexity of the tabular interpretations of automata for tree adjoining languages. The idea is to split each deduction step in $\mathcal{D}_{\text{Earley}}^{\text{Comp}[\circ\circ][\circ\circ\eta]}$ into two different steps such that their final complexity is at most $\mathcal{O}(n^6)$:

$$\mathcal{D}_{\text{Earley}-\text{LIG}}^{\text{Comp}[\circ\circ][\circ\circ\eta]^0} = \frac{\begin{array}{c}[A, j \mid \underline{B} \rightarrow \Upsilon_3 \bullet\, , \eta, j, k \mid C, p, q],\\ [E, h \mid \underline{C} \rightarrow \Upsilon_4 \bullet, \eta', p, q \mid D, r, s]\end{array}}{[[\underline{B} \rightarrow \Upsilon_3 \bullet\, , \eta, j, k \mid D, r, s]]}$$

$$\mathcal{D}_{\text{Earley}-\text{LIG}}^{\text{Comp}[\circ\circ][\circ\circ\eta]^1} = \frac{\begin{array}{l}[[\underline{B} \rightarrow \Upsilon_3 \bullet\, , \eta, j, k \mid D, r, s]],\\ [E, h \mid A[\circ\circ] \rightarrow \Upsilon_1 \bullet B[\circ\circ\eta]\ \Upsilon_2\ , \eta', i, j \mid -, -, -],\\ [E, h \mid \underline{C} \rightarrow \Upsilon_4 \bullet, \eta', p, q \mid D, r, s]\end{array}}{[E, h \mid A[\circ\circ] \rightarrow \Upsilon_1\ B[\circ\circ\eta] \bullet \Upsilon_2, \eta', i, k \mid D, r, s]}$$

Deduction steps in $\mathcal{D}_{\text{Earley}}^{\text{Comp}[\circ\circ][\circ\circ\eta]^0}$ generate an intermediate item of the form $[[\underline{B} \rightarrow \Upsilon_3 \bullet\, , \eta, j, k \mid D, r, s]]$ that will be taken as antecedent for steps in $\mathcal{D}_{\text{Earley}}^{\text{Comp}[\circ\circ][\circ\circ\eta]^1}$ and that represents a derivation

$$\begin{array}{rcl} B[\eta'\eta] & \stackrel{*}{\Rightarrow} & a_{j+1} \cdots a_p\, C[\eta']\, a_{q+1} \cdots a_k \\ & \stackrel{*}{\Rightarrow} & a_{j+1} \cdots a_p \cdots a_r\ D[\,]\ a_{s+1} \cdots a_q \cdots a_k \end{array}$$

for some η', p and q. Deduction steps $\text{Comp}[\circ\circ][\circ\circ\eta]^1$ combine this pseudo-item with an item $[E, h \mid A[\circ\circ] \rightarrow \Upsilon_1 \bullet B[\circ\circ\eta]\ \Upsilon_2\ , \eta', i, j \mid -, -, -]$ that represents a derivation

$$\begin{array}{rcl} S[\,] & \stackrel{*}{\Rightarrow} & a_1 \cdots a_h\, E[\zeta]\, \Upsilon_5 \\ & \stackrel{*}{\Rightarrow} & a_1 \cdots a_h \cdots a_i\ A[\zeta\eta']\ \Upsilon_3\Upsilon_5 \\ & \stackrel{*}{\Rightarrow} & a_1 \cdots a_h \cdots a_i \cdots a_j\ B[\zeta\eta'\eta]\ \Upsilon_2\Upsilon_3\Upsilon_5 \end{array}$$

and with an item $[E, h \mid \underline{C} \rightarrow \Upsilon_4 \bullet, \gamma', p, q \mid D, r, s]$ representing a derivation

$$\begin{array}{rcl} S[\,] & \stackrel{*}{\Rightarrow} & a_1 \cdots a_h\, E[\zeta]\, \Upsilon_5 \\ & \stackrel{*}{\Rightarrow} & a_1 \cdots a_h \cdots a_p\ C[\zeta\eta'']\ \Upsilon_4\Upsilon_5 \\ & \stackrel{*}{\Rightarrow} & a_1 \cdots a_h \cdots a_p \cdots a_r\ D[\zeta]\ a_{s+1} \cdots a_q\ \Upsilon_4\Upsilon_5 \end{array}$$

As a result, a new item of the form $[E, h \mid A[\circ\circ] \rightarrow \Upsilon_1\ B[\circ\circ\eta] \bullet \Upsilon_2, \eta', i, k \mid D, r, s]$ is generated. This last item represents the existence of a derivation

$$\begin{array}{rcl} S[\,] & \stackrel{*}{\Rightarrow} & a_1 \cdots a_h\, E[\zeta]\, \Upsilon_5 \\ & \stackrel{*}{\Rightarrow} & a_1 \cdots a_h \cdots a_i\ A[\zeta\eta']\ \Upsilon_3\Upsilon_5 \\ & \stackrel{*}{\Rightarrow} & a_1 \cdots a_h \cdots a_i \cdots a_j\ B[\zeta\eta'\eta]\ \Upsilon_2\Upsilon_3\Upsilon_5 \\ & \stackrel{*}{\Rightarrow} & a_1 \cdots a_h \cdots a_i \cdots a_j \cdots a_p\ C[\zeta\eta']\ a_{q+1} \cdots a_k\ \Upsilon_2\Upsilon_3\Upsilon_5 \\ & \stackrel{*}{\Rightarrow} & a_1 \cdots a_h \cdots a_i \cdots a_j \cdots a_p \cdots a_r\ D[\zeta]\ a_{s+1} \cdots a_{q+1} \cdots a_k\ \Upsilon_2\Upsilon_3\Upsilon_5 \end{array}$$

In the case of TAG, the CYK-like and Earley-like parsing algorithms described in the previous sections do not preserve the correct prefix property because foot-prediction (a top-down operation) is not restrictive enough to guarantee that the subtree attached to the foot node is really a subtree of the tree involved in the adjunction. The following is the list of deduction steps for TAG obtained from the list of deduction steps for LIG:

$$\mathcal{D}^{\text{Init}}_{\text{Earley}-\text{TAG}} = \frac{}{[- \mid \top \rightarrow \bullet \mathbf{R}^{\alpha}, 0, 0 \mid -, -]} \quad \alpha \in \boldsymbol{I},\ S = \text{label}(\alpha)$$

$$\mathcal{D}^{\text{Scan}}_{\text{Earley}-\text{TAG}} = \frac{[h \mid N^{\gamma} \rightarrow \delta \bullet M^{\gamma}\nu, i, j \mid p, q], \quad [a, j, j+1]}{[h, N^{\gamma} \rightarrow \delta M^{\gamma} \bullet \nu, i, j+1 \mid p, q]} \quad a = \text{label}(M^{\gamma})$$

$$\mathcal{D}^{\varepsilon}_{\text{Earley}-\text{TAG}} = \frac{[h \mid N^{\gamma} \rightarrow \delta \bullet M^{\gamma}\nu, i, j \mid p, q],}{[h \mid N^{\gamma} \rightarrow \delta M^{\gamma} \bullet \nu, i, j \mid p, q]} \quad \varepsilon = \text{label}(M^{\gamma})$$

The set $\mathcal{D}^{\text{Pred}[\,]}_{\text{E}-\text{LIG}}$ has its counterparts in the set of steps in charge of predicting a node not placed on the spine:

$$\mathcal{D}^{\text{PredNoSpine}}_{\text{Earley}-\text{TAG}} = \frac{[h \mid N^{\gamma} \rightarrow \delta \bullet M^{\gamma}\nu, i, j \mid p, q]}{[- \mid M^{\gamma} \rightarrow \bullet \upsilon, j, j \mid -, -]} \quad \begin{array}{l} M^{\gamma} \notin \text{spine}(\gamma), \\ \mathbf{nil} \in \text{adj}(M^{\gamma}) \end{array}$$

Items related to a node M^{γ} not on a spine need not keep trace of h (corresponding to the input position when starting the traversal of γ), as done for LIG for non descendant children.

The counterpart of $\mathcal{D}^{\text{Pred}[\circ\circ][\circ\circ]}_{\text{Earley}-\text{LIG}}$ is the set of steps in charge of predicting children nodes of a node placed on the spine:

$$\mathcal{D}^{\text{PredSpine}}_{\text{Earley}-\text{TAG}} = \frac{[h \mid N^{\gamma} \rightarrow \delta \bullet M^{\gamma}\nu, i, j \mid -, -]}{[h \mid M^{\gamma} \rightarrow \bullet \upsilon, j, j \mid -, -]} \quad \begin{array}{l} M^{\gamma} \in \text{spine}(\gamma), \\ \mathbf{nil} \in \text{adj}(M^{\gamma}) \end{array}$$

Prediction of adjunction is performed by steps equivalent to the consecutive application of two steps in $\mathcal{D}^{\text{Pred}[\circ\circ][\circ\circ]}_{\text{Earley}-\text{LIG}}$ and $\mathcal{D}^{\text{Pred}[\circ\circ][\circ\circ\eta]}_{\text{Earley}-\text{LIG}}$:

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

Each step performing the prediction at a foot node of the excised subtree rooted by M^{γ} corresponds to the consecutive application of the counterparts of a step in $\mathcal{D}^{\text{Pred}[\circ\circ\eta][\circ\circ]}_{\text{Earley}-\text{LIG}}$ and a step in $\mathcal{D}^{\text{Pred}[\,]}_{\text{Earley}-\text{LIG}}$ if M^{γ} is not on a spine:

$$\mathcal{D}^{\text{FootPredNoSpine}}_{\text{Earley}-\text{TAG}} = \frac{[h \mid \mathbf{F}^{\beta} \rightarrow \bullet \bot, j, j \mid -, -], \quad [m \mid N^{\gamma} \rightarrow \delta \bullet M^{\gamma}\nu, i, h \mid p, q]}{[- \mid M^{\gamma} \rightarrow \bullet \upsilon, j, j \mid -, -]} \quad \begin{array}{l} \beta \in \text{adj}(M^{\gamma}), \\ M^{\gamma} \notin \text{spine}(\gamma) \end{array}$$

but if M^γ is placed on the spine of γ then the resulting step corresponds to the consecutive application of the counterparts of a step in $\mathcal{D}^{\text{Pred}[\circ\circ\eta][\circ\circ]}_{\text{Earley}-\text{LIG}}$ and a step in $\mathcal{D}^{\text{Pred}[\circ\circ][\circ\circ]}_{\text{Earley}-\text{LIG}}$:

$$\mathcal{D}^{\text{FootPredSpine}}_{\text{Earley}-\text{TAG}} = \frac{\begin{array}{l}[h \mid \mathbf{F}^\beta \rightarrow \bullet\bot, j, j \mid -, -],\\ [m \mid N^\gamma \rightarrow \delta \bullet M^\gamma\nu, i, h \mid -, -]\end{array}}{[m \mid M^\gamma \rightarrow \bullet\upsilon, j, j \mid -, -]} \quad \begin{array}{l}\beta \in \text{adj}(M^\gamma),\\ M^\gamma \in \text{spine}(\gamma)\end{array}$$

The counterparts of $\mathcal{D}^{\text{Comp}[\,]}_{\text{E}-\text{LIG}}$ and $\mathcal{D}^{\text{Comp}[\circ\circ][\circ\circ]}_{\text{E}-\text{TAG}}$ corresponds to steps traversing elementary trees bottom-up:

$$\mathcal{D}^{\text{CompNoSpine}}_{\text{Earley}-\text{TAG}} = \frac{\begin{array}{l}[h \mid N^\gamma \rightarrow \delta \bullet M^\gamma\nu, i, j \mid p, q],\\ [- \mid M^\gamma \rightarrow \upsilon\bullet, j, k \mid -, -]\end{array}}{[h \mid N^\gamma \rightarrow \delta M^\gamma \bullet \nu, i, k \mid p, q]} \quad \begin{array}{l}M^\gamma \notin \text{spine}(\gamma)\\ \textbf{nil} \in \text{adj}(M^\gamma)\end{array}$$

$$\mathcal{D}^{\text{CompSpine}}_{\text{Earley}-\text{TAG}} = \frac{\begin{array}{l}[h \mid N^\gamma \rightarrow \delta \bullet M^\gamma\nu, i, j \mid -, -],\\ [h \mid M^\gamma \rightarrow \upsilon\bullet, j, k \mid p, q]\end{array}}{[h \mid N^\gamma \rightarrow \delta M^\gamma \bullet \nu, i, k \mid p, q]} \quad \begin{array}{l}M^\gamma \in \text{spine}(\gamma)\\ \textbf{nil} \in \text{adj}(M^\gamma)\end{array}$$

Upon completion of the traversal of the subtree rooted at a node M^γ, not on the spine, and predicted at a foot node $\mathbf{F}^\beta$, the traversal of β is resumed by the following step which is the result of the consecutive application of the counterparts of $\mathcal{D}^{\text{Comp}[\circ\circ\eta][\circ\circ]}_{\text{Earley}-\text{LIG}}$ and $\mathcal{D}^{\text{Comp}[\,]}_{\text{Earley}-\text{LIG}}$:

$$\mathcal{D}^{\text{FootCompNoSpine}}_{\text{Earley}-\text{TAG}} = \frac{\begin{array}{l}[h \mid \mathbf{F}^\beta \rightarrow \bullet\bot, j, j \mid -, -],\\ [m \mid N^\gamma \rightarrow \delta \bullet M^\gamma\nu, i, h \mid p, q],\\ [- \mid M^\gamma \rightarrow \upsilon\bullet, j, k \mid -, -]\end{array}}{[h \mid \mathbf{F}^\beta \rightarrow \bot\bullet, j, k \mid j, k]} \quad \begin{array}{l}\beta \in \text{adj}(M^\gamma),\\ M^\gamma \notin \text{spine}(\gamma)\end{array}$$

If M^γ is not in the spine of γ then the set of deduction steps for the completion of foot is the counterpart of $\mathcal{D}^{\text{Comp}[\circ\circ\eta][\circ\circ]}_{\text{E}-\text{LIG}}$:

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

Upon completion of the traversal of β, a deduction step in the counterpart of $\mathcal{D}^{\text{Comp}[\circ\circ][\circ\circ\eta]}_{\text{Earley}-\text{LIG}}$ checks if the subtree attached to the foot of β corresponds to the one rooted at the adjunction node M^γ. When M^γ is on a spine, each step works in coordination with a $\mathcal{D}^{\text{Comp}[\circ\circ][\circ\circ]}_{\text{Earley}-\text{LIG}}$ step:

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

If the adjunction node is not on a spine, then deduction steps for the completion of adjunction are equivalent to the consecutive application of two steps in the counterparts of $\mathcal{D}^{\text{Comp}[\circ\circ][\circ\circ\eta]}_{\text{Earley}-\text{LIG}}$ and $\mathcal{D}^{\text{Comp}[\,]}_{\text{Earley}-\text{LIG}}$:

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

The time complexity of this algorithm, with respect to the length n of the input string, is $\mathcal{O}(n^7)$, and is given by the adjunction completion steps. To reduce it to the standard $\mathcal{O}(n^6)$ time complexity, we must apply the same technique used in the case of LIG parsing, splitting $\mathcal{D}^{\text{AdjCompSpine}}_{\text{Earley}-\text{TAG}}$ into $\mathcal{D}^{\text{AdjComp}^0}_{\text{Earley}-\text{TAG}}$ and $\mathcal{D}^{\text{AdjCompSpine}^1}_{\text{Earley}-\text{TAG}}$, whereas $\mathcal{D}^{\text{AdjCompNOSpine}}_{\text{Earley}-\text{TAG}}$ is split into $\mathcal{D}^{\text{AdjComp}^0}_{\text{Earley}-\text{TAG}}$ and $\mathcal{D}^{\text{AdjCompNoSpine}^1}_{\text{Earley}-\text{TAG}}$:

$$\mathcal{D}^{\text{AdjComp}^0}_{\text{Earley}-\text{TAG}} = \frac{\begin{array}{l}[j \mid \top \rightarrow \mathbf{R}^\beta \bullet, j, k \mid p, q],\\ [h \mid M^\gamma \rightarrow \delta \bullet, p, q \mid r, s]\end{array}}{[[M^\gamma \rightarrow \delta \bullet, j, k \mid r, s]]} \quad \beta \in \text{adj}(M^\gamma)$$

$$\mathcal{D}^{\text{AdjCompSpine}^1}_{\text{Earley}-\text{TAG}} = \frac{\begin{array}{l}[[M^\gamma \rightarrow \upsilon \bullet, j, k \mid r, s]],\\ [h \mid N^\gamma \rightarrow \delta \bullet M^\gamma \nu, i, j \mid -, -],\\ [h \mid M^\gamma \rightarrow \upsilon \bullet, p, q \mid r, s]\end{array}}{[h \mid N^\gamma \rightarrow \delta M^\gamma \bullet \nu, i, k \mid r, s]} \quad \begin{array}{l}\beta \in \text{adj}(M^\gamma),\\ M^\gamma \in \text{spine}(\gamma)\end{array}$$

$$\mathcal{D}^{\text{AdjCompNoSpine}^1}_{\text{Earley}-\text{TAG}} = \frac{\begin{array}{l}[[M^\gamma \rightarrow \upsilon \bullet, j, k \mid r, s]],\\ [h \mid N^\gamma \rightarrow \delta \bullet M^\gamma \nu, i, j \mid p', q'],\\ [- \mid M^\gamma \rightarrow \upsilon \bullet, p, q \mid -, -]\end{array}}{[h \mid N^\gamma \rightarrow \delta M^\gamma \bullet \nu, i, k \mid p', q']} \quad \begin{array}{l}\beta \in \text{adj}(M^\gamma),\\ M^\gamma \notin \text{spine}(\gamma)\end{array}$$

7. Bidirectional Parsing

Bidirectional parsing strategies can start computations at any position of the input string and can span to the right and to the left to include substrings which were scanned in a bidirectional manner by subcomputations. Although these kinds of strategies seems to be naturally adapted for TAG, only a few bidirectional parsing algorithms have been proposed (Lavelli and Satta, 1991; van Noord, 1994; Díaz et al., 2000). In the case of LIG, to the best of our knowledge, only a bottom-up head-corner parser has been defined (Schneider, 2000).

In this section we propose a new bidirectional bottom-up parser for LIG derived from the context-free parser defined by De Vreught and

Honig (Sikkel, 1997). Intermediate results of the parsing process are stored as items of the form

$$[\underline{A} \to \Upsilon_1 \bullet \Upsilon_2 \bullet \Upsilon_3, \eta, i, j \mid B, p, q]$$

with double-dotted productions. Each item represents one of the following kinds of derivation:

- $\Upsilon_2 \stackrel{*}{\Rightarrow} \Upsilon_{2'}\ E[\eta]\ \Upsilon_{2''} \stackrel{*}{\Rightarrow} a_{i+1} \cdots a_p\ B[\,]\ a_{q+1} \cdots a_j$ iff $(B, p, q) \neq (-, -, -)$, where $B[\,]$ is a dependent descendent of $E[\eta]$ and $(p, q) \leq (i, j)$.
- $\Upsilon_2 \stackrel{*}{\Rightarrow} a_{i+1} \cdots a_j$ iff $\eta = -$ and $(B, p, q) = (-, -, -)$.

The input string $a_1 \cdots a_n$ is recognized if an item in $\mathcal{F} = \{\, [\underline{S} \to \bullet\Upsilon\bullet\,, -, 0, n \mid -, -, -]\,\}$ is generated.

The parsing process starts by recognizing a terminal symbol or the empty string:

$$\mathcal{D}^{\text{Scan}}_{\text{dVH-LIG}} = \frac{[a, j, j+1]}{[A[\,] \to \bullet a\bullet, -, j, j+1 \mid -, -, -]}$$

$$\mathcal{D}^{\varepsilon}_{\text{dVH-LIG}} = \frac{}{[A[\,] \to \bullet\varepsilon\bullet, -, j, j \mid -, -, -]}$$

Upon recognition of a production defining non-terminal B, the bottom-up recognition of productions having B in their body is triggered:

$$\mathcal{D}^{\text{Inc}[\,]}_{\text{dVH-LIG}} = \frac{[\underline{B} \to \bullet\Upsilon_3\bullet, -, i, j \mid -, -, -]}{[\underline{A} \to \Upsilon_1 \bullet\ B[\,] \bullet \Upsilon_2, -, i, j \mid -, -, -]}$$

$$\mathcal{D}^{\text{Inc}[\circ\circ][\circ\circ]}_{\text{dVH-LIG}} = \frac{[\underline{B} \to \bullet\Upsilon_3\bullet, \eta, i, j \mid C, p, q]}{[A[\circ\circ] \to \Upsilon_1 \bullet\ B[\circ\circ] \bullet \Upsilon_2, \eta, i, j \mid C, p, q]}$$

$$\mathcal{D}^{\text{Inc}[\circ\circ\eta][\circ\circ]}_{\text{dVH-LIG}} = \frac{[\underline{B} \to \bullet\Upsilon_3\bullet, \eta', i, j \mid C, p, q]}{[A[\circ\circ\eta] \to \Upsilon_1 \bullet\ B[\circ\circ] \bullet \Upsilon_2, \eta, i, j \mid B, i, j]}$$

$$\mathcal{D}^{\text{Inc}[\circ\circ][\circ\circ\eta]}_{\text{dVH-LIG}} = \frac{\begin{array}{c}[\underline{B} \to \bullet\Upsilon_3\bullet, \eta, i, j \mid C, p, q],\\ [\underline{C} \to \bullet\Upsilon_4\bullet, \eta', p, q \mid D, r, s]\end{array}}{[A[\circ\circ\eta] \to \Upsilon_1 \bullet\ B[\circ\circ] \bullet \Upsilon_2, \eta', i, j \mid D, r, s]}$$

When two consecutive parts of the right-hand side of a rule have been recognized, they are concatenated:

$$\mathcal{D}^{\text{Conc}[\,]}_{\text{dVH-LIG}} = \frac{\begin{array}{c}[\underline{A} \to \Upsilon_1 \bullet \Upsilon_2 \bullet \Upsilon_3\Upsilon_4, \eta, i, j \mid C, p, q],\\ [\underline{A} \to \Upsilon_1\Upsilon_2 \bullet \Upsilon_3 \bullet \Upsilon_4, \eta', j, k \mid C', p', q'],\end{array}}{[\underline{A} \to \Upsilon_1 \bullet \Upsilon_2\Upsilon_3 \bullet \Upsilon_4, \eta \cup \eta', i, k \mid C \cup C', p \cup p', q \cup q']}$$

where, given X and Y, we define $X \cup Y$ as X if Y is unbound, as Y if X is unbound, and being undefined otherwise.

The translation of this algorithm to TAG uses items of the form

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

representing one of the following two situations:

- $\delta \stackrel{*}{\Rightarrow} a_{i+1} \cdots a_p \ \mathbf{F}^\gamma \ a_{q+1} \cdots a_j \Rightarrow_f a_{i+1} \cdots a_p \ \upsilon \ a_{q+1} \cdots a_j \stackrel{*}{\Rightarrow}_f a_{i+1} \cdots a_j$, $M^{\gamma'} \rightarrow \upsilon$, and $\gamma \in \text{adj}(M^{\gamma'})$ iff $(p, q) \neq (-, -)$.
- $\delta \stackrel{*}{\Rightarrow} a_{i+1} \cdots a_j$ iff $(p, q) = (-, -)$.

As already mentioned for other TAG parsing algorithms, the element M^γ is redundant, allowing more compact items (Díaz et al., 2000) of the form:

$$[N^\gamma \rightarrow \nu \bullet \delta \bullet \omega, i, j \mid p, q]$$

As before, the parsing process is started bottom-up by means of the recognition of terminal symbols or the empty string:

$$\mathcal{D}^{\text{Scan}}_{\text{dVH-TAG}} = \frac{[a, j, j+1]}{[N^\gamma \rightarrow \nu \bullet M^\gamma \bullet \omega, j, j+1 \mid -, -]} \quad a = \text{label}(M^\gamma)$$

$$\mathcal{D}^{\varepsilon}_{\text{dVH-TAG}} = \frac{}{[N^\gamma \rightarrow \nu \bullet M^\gamma \bullet \omega, j, j \mid -, -]} \quad \varepsilon = \text{label}(M^\gamma)$$

Steps in $\mathcal{D}^{\text{Inc}[\,]}_{\text{dVH-LIG}}$ correspond to the bottom-up traversal of nodes not placed on the spine:

$$\mathcal{D}^{\text{IncNoSpine}}_{\text{dVH-TAG}} = \frac{[M^\gamma \rightarrow \bullet\delta\bullet, i, j \mid -, -]}{[N^\gamma \rightarrow \nu \bullet M^\gamma \bullet \omega, i, j \mid -, -]} \quad M^\gamma \notin \text{spine}(\gamma), \ \mathbf{nil} \in \text{adj}(M^\gamma)$$

whereas steps in $\mathcal{D}^{\text{Inc}[\circ\circ][\circ\circ]}_{\text{dVH-LIG}}$ correspond to the bottom-up traversal of nodes in the spine of an auxiliary tree, propagating the list of pending adjunctions:

$$\mathcal{D}^{\text{IncNoSpine}}_{\text{dVH-TAG}} = \frac{[M^\gamma \rightarrow \bullet\delta\bullet, i, j \mid p, q]}{[N^\gamma \rightarrow \nu \bullet M^\gamma \bullet \omega, i, j \mid p, q]} \quad M^\gamma \in \text{spine}(\gamma), \ \mathbf{nil} \in \text{adj}(M^\gamma)$$

Steps in $\mathcal{D}^{\text{Inc}[\circ\circ\eta][\circ\circ]}_{\text{dVH-LIG}}$ correspond to the bottom-up starting of an a adjunction operation at the foot node of an auxiliary tree:

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

whereas steps in $\mathcal{D}^{\text{Inc}[\circ\circ][\circ\circ\eta]}_{\text{dVH}-\text{LIG}}$ correspond to finishing the adjunction operation once the auxiliary tree has been completely traversed:

$$\mathcal{D}^{\text{Adj}}_{\text{dVH}-\text{TAG}} = \frac{[\top \rightarrow \bullet \mathbf{R}^{\beta} \bullet, i, j \mid p, q] \quad [M^{\gamma} \rightarrow \bullet \delta \bullet, p, q \mid r, s]}{[N^{\gamma} \rightarrow \nu \bullet M^{\gamma} \bullet \omega, i, j \mid r, s]} \quad \beta \in \text{adj}(M^{\gamma})$$

Steps in $\mathcal{D}^{\text{Conc}}_{\text{dVH}-\text{LIG}}$ correspond to the combination of two partial analyses spanning consecutive parts of the input string:

$$\mathcal{D}^{\text{Conc}}_{\text{dVH}-\text{TAG}} = \frac{[N^{\gamma} \rightarrow \nu \bullet \delta_1 \bullet \delta_2 \omega, i, j \mid p, q] \quad [N^{\gamma} \rightarrow \nu \delta_1 \bullet \delta_2 \bullet \omega, j, k \mid p', q']}{[N^{\gamma} \rightarrow \nu \bullet \delta_1 \delta_2 \bullet \omega, i, j \mid p \cup p', q \cup q']}$$

The same relations between dVH-LIG and dVH-TAG can be found between the bottom-up head-corner parser defined by (Schneider, 2000) and a bottom-up head-corner parser for TAG. It is also possible to define bidirectional parsing algorithms for LIG that mimic the head-corner algorithm for TAG defined by (van Noord, 1994) or the algorithm proposed by (Lavelli and Satta, 1991).

8. Specialized TAG parsers

We have shown in the previous sections that TAG and LIG are very closely related and that a parsing schema for one formalism may easily be transposed to the other formalism. However, LIG is more generic than TAG (with an easy encoding of TAG as LIG) which means that more specialized and efficient schemata are possible for TAG. Actually, this fact has been used in previous schemata to simplify the items for TAG by deleting redundant or useless information. These simplifications exploited the facts that (a) the traversal of an auxiliary tree does not need any information about the adjunction node, and that (b) the foot node of an auxiliary tree resumes the traversal of the subtree rooted at the adjunction node and attached to the foot node.

There is another particularity that may help us to design specialized parsing algorithms for TAG, namely that elementary trees represent domains of locality that are lost when converting to LIG clauses. A simple algorithm based on this notion of domain of locality has been presented in (De la Clergerie, 2001). An item of the form $[h \mid N^{\gamma} \rightarrow \delta \bullet M^{\gamma} \nu, j \mid p, q \mid \mathcal{S}]$ retraces the history of the traversal of some elementary tree γ, from the point h where it was started up to the current point j, with (possibly) some hole p, q covered by a foot node, and a (possibly empty) stack $\mathcal{S}$ of pairs (u, v) denoting adjunctions which have been predicted but not yet completed. In such a pair, u denotes

the point where the traversal was suspended because of an adjunction and v denotes the point where the traversal was resumed after leaving the foot node of an auxiliary tree. We note $(u, v) = (-, -)$ in case of null adjunction. Parsing succeeds if a final item belonging to $\mathcal{F} = \{\, [0 \mid \mathbf{R}^\alpha \rightarrow v\bullet, n \mid -, - \mid \emptyset] \mid \alpha \in \boldsymbol{I} \text{ and } S = \text{label}(\mathbf{R}^\alpha)\}$ is derivable.

$$\mathcal{D}^{\text{Init}}_{\text{Weak-TAG}} = \frac{}{[0 \mid \top \rightarrow \bullet\mathbf{R}^\alpha, 0 \mid -, - \mid \emptyset]} \quad \alpha \in \boldsymbol{I},\ S = \text{label}(\alpha)$$

$$\mathcal{D}^{\text{Scan}}_{\text{Weak-TAG}} = \frac{\begin{array}{c}[h \mid N^\gamma \rightarrow \delta \bullet M^\gamma\nu, j \mid p, q \mid \mathcal{S}], \\ [a, j, j+1]\end{array}}{[h, N^\gamma \rightarrow \delta M^\gamma \bullet \nu, j+1 \mid p, q \mid \mathcal{S}]} \quad a = \text{label}(M^\gamma)$$

$$\mathcal{D}^{\varepsilon}_{\text{Weak-TAG}} = \frac{[h \mid N^\gamma \rightarrow \delta \bullet M^\gamma\nu, j \mid p, q \mid \mathcal{S}]}{[h, N^\gamma \rightarrow \delta M^\gamma \bullet \nu, j \mid p, q \mid \mathcal{S}]} \quad \varepsilon = \text{label}(M^\gamma)$$

$$\mathcal{D}^{\text{Pred}}_{\text{Weak-TAG}} = \frac{[h \mid N^\gamma \rightarrow \delta \bullet M^\gamma\nu, j \mid p, q \mid \mathcal{S}]}{[h \mid M^\gamma \rightarrow \bullet\upsilon, j \mid p, q \mid \mathcal{S}, (-, -)]} \quad \textbf{nil} \in \text{adj}(M^\gamma)$$

$$\mathcal{D}^{\text{Comp}}_{\text{Weak-TAG}} = \frac{[h \mid M^\gamma \rightarrow \upsilon\bullet, j \mid p, q \mid \mathcal{S}, (-, -)]}{[h \mid N^\gamma \rightarrow \delta M^\gamma \bullet \nu, j \mid p, q \mid \mathcal{S}]} \quad \textbf{nil} \in \text{adj}(M^\gamma)$$

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

$$\mathcal{D}^{\text{FootPred}}_{\text{Weak-TAG}} = \frac{\begin{array}{c}[h \mid \mathbf{F}^\beta \rightarrow \bullet\bot, i \mid -, - \mid \mathcal{S}'], \\ [m \mid N^\gamma \rightarrow \upsilon \bullet M^\gamma\nu, h \mid p, q \mid \mathcal{S}]\end{array}}{[m \mid M^\gamma \rightarrow \bullet\upsilon, i \mid p, q \mid \mathcal{S}, (h, i)]} \quad \beta \in \text{adj}(M^\gamma)$$

$$\mathcal{D}^{\text{FootComp}}_{\text{Weak-TAG}} = \frac{\begin{array}{c}[h \mid \mathbf{F}^\beta \rightarrow \bullet\bot, i \mid -, - \mid \mathcal{S}'], \\ [m \mid M^\gamma \rightarrow \upsilon\bullet, j \mid p, q \mid \mathcal{S}, (h, i)]\end{array}}{[h \mid \mathbf{F}^\beta \rightarrow \bot\bullet, j \mid i, j \mid \mathcal{S}']} \quad \beta \in \text{adj}(M^\gamma)$$

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

The resulting algorithm corresponds to a left-to-right top-down parsing strategy preserving the correct prefix property and is simpler than many other algorithms. Its worst-case complexity is $\mathcal{O}(n^{4+2d})$ in space and $\mathcal{O}(n^{5+2d})$ in time where d denotes the maximal depth of elementary trees, and hence the maximal number of uncompleted adjunctions at some point in an elementary tree. These complexities are not optimal but experiments performed with linguistic grammars have nevertheless shown that this algorithm can be efficient (De la Clergerie, 2001).

9. Conclusion

We have presented a set of algorithms for LIG and TAG parsing, including pure bottom-up algorithms, Earley-like algorithms with weak prediction, Earley-like algorithms with strong prediction that preserves the correct prefix property and bidirectional algorithms. Other algorithms could also have been included, but for reasons of space we have chosen to show only the algorithms we consider milestones in the development of parsers for LIG and TAG.

We have also studied the relations among the steps in charge of recognizing the adjunction operation in a TAG and the steps in charge of transmitting information through the spine in a LIG, obtaining fruitful results. For example, the TAG formalism restricts combination of items by means of (explicit or implicit) adjoining constraints. These constraints are not present in LIG, and so LIG parsing algorithms must take into account phenomena that never occur in TAG parsing. As a result, items in TAG parsers are more compact than their counterparts in LIG parsers. In addition, some practical optimizations for TAG parsing algorithms (e.g., the *weak tabular interpretation* presented by De la Clergerie, 2001) are not valid for LIG parsers.

Acknowledgments

The research reported in this chapter has been partially supported by Plan Nacional de Investigación Científica, Desarrollo e Innovación Tecnológica (TIC2000-0370-C02-01), by Ministerio de Ciencia y Tecnología (HP2001-0044), by Xunta de Galicia (PGIDT01PXI10506PN, PGIDIT02PXIB30501PR, PGIDIT02SIN01E), and by Universidade da Coruña.

References

Alonso, M. A., Cabrero, D., De la Clergerie, É., and Vilares, M. (1999). Tabular algorithms for TAG parsing. In *Proc. of EACL'99, Ninth Conference of the European Chapter of the Association for Computational Linguistics*, pages 150–157, Bergen, Norway. ACL.

Alonso, M. A., De la Clergerie, É., Graña, J., and Vilares, M. (2000a). New tabular algorithms for LIG parsing. In *Proc. of the Sixth International Workshop on Parsing Technologies (IWPT 2000)*, pages 29–40, Trento, Italy.

Alonso, M. A., Nederhof, M.-J., and De la Clergerie, É. (2000b). Tabulation of automata for tree adjoining languages. *Grammars*, 3(2/3):89–110.

De la Clergerie, É. (2001). Refining tabular parsers for TAGs. In *Proceedings of Language Technologies 2001: The Second Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL'01)*, pages 167–174, CMU, Pittsburgh, PA, USA.

De la Clergerie, É. and Alonso, M. A. (1998). A tabular interpretation of a class of 2-Stack Automata. In *COLING-ACL'98, 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Proceedings of the Conference*, volume II, pages 1333–1339, Montreal, Quebec, Canada. ACL.

De la Clergerie, É., Alonso, M. A., and Cabrero, D. (1998). A tabular interpretation of bottom-up automata for TAG. In *Proc. of Fourth International Workshop on Tree-Adjoining Grammars and Related Frameworks (TAG+4)*, pages 42–45, Philadelphia, PA, USA.

Díaz, V. J., Carrillo, V. and Alonso, M. A. (2000). A bidirectional bottom-up parser for TAG. In *Proc. of the Sixth International Workshop on Parsing Technologies (IWPT 2000)*, pages 299–300, Trento, Italy.

Gazdar, G. (1987). Applicability of indexed grammars to natural languages. In Reyle, U. and Rohrer, C., editors, *Natural Language Parsing and Linguistic Theories*, pages 69–94. D. Reidel Publishing Company.

Joshi, A. K. and Schabes, Y. (1997). Tree-adjoining grammars. In Rozenberg, G. and Salomaa, A., editors, *Handbook of Formal Languages. Vol 3: Beyond Words*, chapter 2, pages 69–123. Springer-Verlag, Berlin/Heidelberg/New York.

Kasami, T. (1965). An efficient recognition and syntax algorithm for context-free languages. Scientific Report AFCRL-65-758, Air Force Cambridge Research Lab., Bedford, Massachussetts.

Lavelli, A. and Satta, G. (1991) Bidirectional parsing of lexicalized tree adjoining grammars. In *Proceedings of the 5th Conference of the European Chapter of the Association for Computational Linguistics (EACL'91)*, pages 27–32, Berlin, Germany.

Nederhof, M.-J. (1998). Linear indexed automata and tabulation of TAG parsing. In *Proc. of First Workshop on Tabulation in Parsing and Deduction (TAPD'98)*, pages 1–9, Paris, France.

Nederhof, M.-J. (1999). The computational complexity of the correct-prefix property for TAGs. *Computational Linguistics*, 25(3):345–360.

Van Noord, G. (1994). Head-corner parsing for TAG. *Computational Intelligence*, 10(4):525–534, 1994.

Schabes, Y. (1991). The valid prefix property and left to right parsing of tree-adjoining grammar. In *Proc. of II International Workshop on Parsing Technologies, IWPT'91*, pages 21–30, Cancún, Mexico.

Schabes, Y. and Shieber, S. M. (1994). An alternative conception of tree-adjoining derivation. *Computational Linguistics*, 20(1):91–124.

Schneider, K.-M. (2000). Algebraic construction of parsing schemata. In *Proc. of the Sixth International Workshop on Parsing Technologies (IWPT 2000)*, pages 242–253, Trento, Italy.

Sikkel, K. (1997). *Parsing Schemata — A Framework for Specification and Analysis of Parsing Algorithms*. Texts in Theoretical Computer Science — An EATCS Series. Springer-Verlag, Berlin/Heidelberg/New York.

Vijay-Shanker, K. and Weir, D. J. (1991). Polynomial parsing of extensions of context-free grammars. In Tomita, M., editor, *Current Issues in Parsing Technology*, chapter 13, pages 191–206. Kluwer Academic Publishers, Norwell, MA, USA.

Vijay-Shanker, K. and Weir, D. J. (1993). Parsing some constrained grammar formalisms. *Computational Linguistics*, 19(4):591–636.

Younger, D. H. (1967). Recognition and parsing of context-free languages in time n^3. *Information and Control*, 10(2):189–208.