Variants of mixed parsing of TAG and TIG

Miguel A. Alonso* — **Víctor J. Díaz****

[*] *Departamento de Computación, Universidade da Coruña*
Campus de Elviña s/n, 15071 La Coruña (Spain)
alonso@udc.es

[**] *Departamento de Lenguajes y Sistemas Informáticos, Universidad de Sevilla*
Avda. Reina Mercedes s/n, 41012 Sevilla (Spain)
vjdiaz@lsi.us.es

ABSTRACT. *Tree Adjoining Grammar (TAG) is a useful formalism for describing the syntactic structure of natural languages. In practice, a large part of wide coverage TAGs is formed by trees that satisfy the restrictions imposed by Tree Insertion Grammar (TIG), a simpler formalism. This characteristic can be used to reduce the practical complexity of TAG parsing, applying the standard adjunction operation only in those cases in which the simpler cubic-time TIG adjunction cannot be applied. A major obstacle to this task is posed by the fact that simultaneous adjunctions are forbidden in TAG but they are allowed in TIG. In this article, we describe several algorithms for mixed parsing of TAG and TIG: a first one forbidding simultaneous adjunctions, a second one allowing this kind of adjunctions, and a third one which extends the second one to preserve the correct prefix property.*

RÉSUMÉ. *La Grammaire d'Arbres Adjoints (TAG) est un formalisme utile pour décrire la structure syntaxique des langues naturelles. En pratique, la plupart des TAG à large couverture contiennent des arbres qui satisfont les restrictions imposées par la Grammaire d'Insertion d'Arbres (TIG), qui est un formalisme plus simple. Cette caractéristique peut être employée pour réduire la complexité pratique de l'analyse TAG, en appliquant l'opération d'adjonction standard seulement dans les cas où l'adjonction TIG, plus simple, ne peut pas être appliquée. L'un des plus grands obstacles à cette tâche réside dans le fait que les adjonctions simultanées sont interdites en TAG mais elles sont permises en TIG. Dans cet article, nous décrivons plusieurs algorithmes pour l'analyse mixte de TAG et de TIG : 1) celui qui interdit les adjonctions simultanées ; 2) celui qui permet ce type d'adjonction ; et 3) celui qui étend le deuxième afin de préserver la propriété du préfixe correcte.*

KEYWORDS: *parsing, tree adjoining grammar, tree insertion grammar.*

MOTS-CLÉS : *analyse syntaxique, grammaires d'arbres adjoints, grammaires d'insertion d'arbres.*

1. Introduction

Tree Adjoining Grammar (TAG) [JOS 75, JOS 97, ABE 00] and Tree Insertion Grammar (TIG) [SCH 95] are grammatical formalisms that make use of a tree-based operation called adjunction. However, adjunctions are more restricted in the case of TIG than in the case of TAG, which has important consequences with respect to the set of languages generated and the worst-case complexity of parsing algorithms:

– TAG generates tree adjoining languages, a strict superset of context-free languages, and the complexity of parsing algorithms is in $\mathcal{O}(n^6)$ for time and in $\mathcal{O}(n^4)$ for space with respect to the length n of the input string.

– TIG generates context-free languages and can be parsed in $\mathcal{O}(n^3)$ for time and in $\mathcal{O}(n^2)$ for space.

– The correct prefix property [SCH 91] is preserved by TIG parsers without increasing their computational cost. In the case of TAG, preserving this property involves an increase in the space complexity from $\mathcal{O}(n^4)$ to $\mathcal{O}(n^5)$ [NED 99].

Although the powerful adjunction provided by TAG makes it useful for describing the syntax of natural languages, most of the trees involved in wide coverage grammars like XTAG [DOR 94] do not make use of such operation, and so a large portion of XTAG is in fact a TIG [SCH 95]. As the full power of a TAG parser is only put into practice in adjunctions involving a given set of trees, to apply a parser working in $\mathcal{O}(n^6)$ time complexity when most of the work can be done by a $\mathcal{O}(n^3)$ parser seems to be a waste of computing resources. In this article, we propose to improve the practical efficiency of TAG parsers by applying mixed parser strategies that takes the best of both worlds: those parts of the grammar that correspond to a TIG are managed in $\mathcal{O}(n^3)$ time and $\mathcal{O}(n^2)$ space complexity, and only those parts of the grammar involving the full kind of adjunction present in TAG are managed in $\mathcal{O}(n^6)$ time and $\mathcal{O}(n^4)$ space complexity ($\mathcal{O}(n^5)$ space complexity in the case of parsers satisfying the correct prefix property).

This article may be outlined as follows. The remainder of this section is devoted to describe the notation used in the article. In section 2 we present a mixed parsing algorithm in which at most one auxiliary tree is allowed to be adjoined at a given node. This algorithm is modified in section 3 to allow simultaneous adjunctions. New modifications are considered in section 4 in order to preserve the correct prefix property. The computational complexity of these algorithms is analyzed in section 5 and their practical efficiency is studied in section 6. Section 7 presents final conclusions.

1.1. *Tree Adjoining Grammars*

Formally, a TAG is a 5-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 a finite set of terminal symbols, $S \in V_N$ the axiom of the grammar, $\boldsymbol{I}$ a finite set of *initial trees* and $\boldsymbol{A}$ a finite set of *auxiliary trees*. $\boldsymbol{I} \cup \boldsymbol{A}$ is the set of *elementary trees*. Internal nodes of elementary trees are labeled by non-

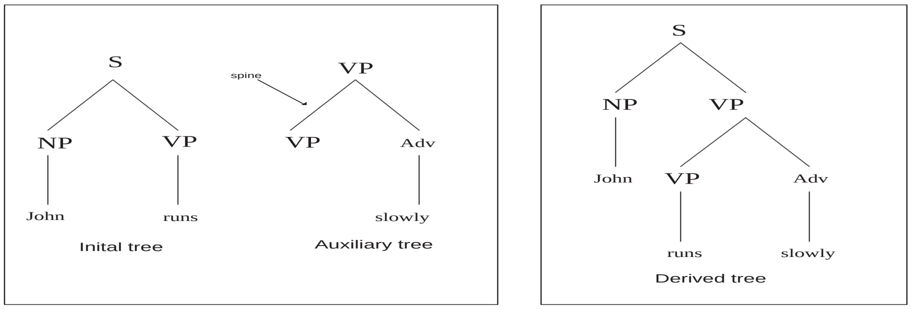

Figure 1. *Adjunction operation*

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 auxiliary tree from the root node to the foot node is called the *spine* of the tree. New trees are derived by *adjunction*: let γ be an elementary or derived 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 can add constraints on the nodes of elementary trees so the adjunction on a node can be mandatory, optional or forbidden. The string language of a TAG $\mathcal{G}$ is defined as the set of yields of all the trees derived from initial trees rooted by the axiom of the grammar [JOS 97]. We illustrate the adjunction operation in Fig. 1, where we show a simple TAG with two elementary trees: an initial tree rooted S and an auxiliary tree rooted VP. The derived tree obtained after adjoining the VP auxiliary tree at the node labeled by VP located in the initial tree is also shown.[1]

1.2. *Tree Insertion Grammars*

We can consider the set $\boldsymbol{A}$ as formed by the union of the sets $\boldsymbol{A}_L$, containing *left auxiliary trees* in which every nonempty frontier node[2] is to the left of the foot node, $\boldsymbol{A}_R$, containing *right auxiliary trees* in which every nonempty frontier node is to the right of the foot node, and $\boldsymbol{A}_W$, containing *wrapping auxiliary trees* in which nonempty frontier nodes are placed both to the left and to the right of the foot node. Figure 2 shows three derived trees resulting from the adjunction of a wrapping, left and right auxiliary tree, respectively. We can note from that picture that the trees derived by the adjunction of left an right auxiliary trees are simpler than those derived

1. The operation of *substitution* can also be defined for TAG, but it does not increase the generative power of the formalism. The incorporation of substitution to the parsing algorithms defined in this article is straightforward and does not modify their complexity.
2. An empty frontier node is a leaf node labeled by the empty string ε.

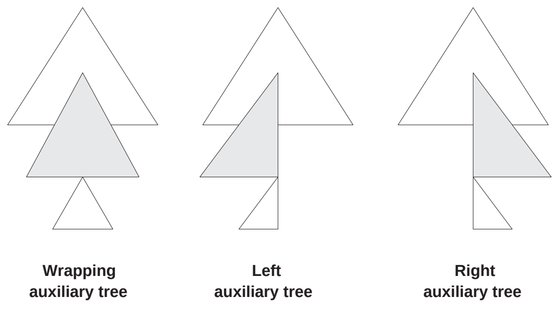

Figure 2. *Adjunction of left, right and wrapping auxiliary trees*

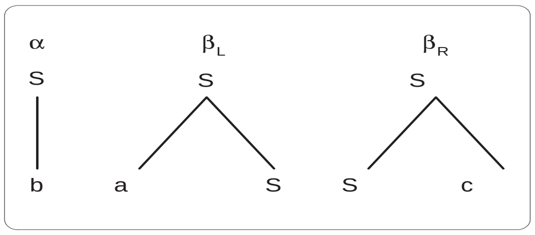

Figure 3. *Example of TIG grammar*

by wrapping auxiliary trees. It is just this evidence which introduces the notion of Tree Insertion Grammars.

Given an auxiliary tree, those nodes placed on the spine are called *spine nodes* and those nodes placed to the left (resp. right) of the spine are called *left nodes* (resp. *right nodes*). The set $\boldsymbol{A}_{SL} \subseteq \boldsymbol{A}_L$ (resp. $\boldsymbol{A}_{SR} \subseteq \boldsymbol{A}_R$) of *strongly left* (resp. *strongly right*) auxiliary trees is formed by trees in which no adjunction is permitted on right (resp. left) nodes and only strongly left (resp. right) auxiliary trees are allowed to adjoin on spine nodes. We denote by $\boldsymbol{A'}$ the set $\boldsymbol{A} - (\boldsymbol{A}_{SL} \cup \boldsymbol{A}_{SR})$. Given the set $\boldsymbol{A}$ of a TAG, we can determine the set $\boldsymbol{A}_{SL}$ as follows: firstly, we determine the set $\boldsymbol{A}_L$ examining the frontier of the trees in $\boldsymbol{A}$ and we set $\boldsymbol{A}_{SL} := \boldsymbol{A}_L$; secondly, we eliminate from $\boldsymbol{A}_{SL}$ those trees that permit adjunctions on nodes to the right of their spine; and thirdly, we iteratively eliminate from $\boldsymbol{A}_{SL}$ those trees that allow adjoining trees in $\boldsymbol{A} - \boldsymbol{A}_{SL}$ on nodes of their spine. $\boldsymbol{A}_{SR}$ is determined in an analogous way.

In essence, a TIG is a restricted TAG where auxiliary trees must be either strongly left or strongly right and adjunctions are not allowed in root and foot nodes of auxiliary trees. There is also a different approach between both formalism with respect to the way adjunctions are performed. In contrast with TAG, where only an auxiliary tree can be adjoined at a node, TIG enables simultaneous adjunctions, i.e., the adjunction of several auxiliary trees on a node of a tree. We illustrate this point in Figure 3 where a TIG grammar with an initial tree α, a left auxiliary tree β_L and a right auxiliary tree

β_R is depicted. When simultaneous adjunction of β_L and β_R is allowed at the root node of α, the TIG language is a^*bc^*, i.e., an optional sequence of a's followed by a b and followed by an optional sequence of c's. In contrast, if simultaneous adjunction was not allowed, we could not combine the left and right auxiliary trees, and the language generated would be the union of the strings ab and bc.[3]

1.3. *Notation for parsing algorithms*

We will describe parsing algorithms using *Parsing Schemata*, a framework for high-level descriptions of parsing algorithms [SIK 97]. A *parsing system* for a grammar G and string $a_1 \dots a_n$ 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 called *hypothesis* that encodes the sentence 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 $\frac{\eta_1, \dots, \eta_k}{\xi}$*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* represent the recognition of a sentence. A *parsing schema* is a parsing system parameterized by a grammar and a sentence.

Given an input string $a_1 \dots a_n$, the hypothesis of all parsing systems described in this article will be defined in the standard way:

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

In order to describe the parsing algorithms for tree-based formalisms, 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 $N_1^\gamma \dots N_g^\gamma$ are represented by a production $N^\gamma \rightarrow N_1^\gamma \dots N_g^\gamma$. 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.

To simplify the description of parsing algorithms we consider an additional production $\top \rightarrow \mathbf{R}^\alpha$ for each $\alpha \in \boldsymbol{I}$ and the two additional productions $\top \rightarrow \mathbf{R}^\beta$ and $\mathbf{F}^\beta \rightarrow \bot$ for each $\beta \in \boldsymbol{A}$, where $\mathbf{R}^\beta$ and $\mathbf{F}^\beta$ correspond to the root node and the foot node of β, respectively. After disabling $\top$ and $\bot$ as adjunction nodes the generative capability of the grammars remains intact. We introduce also the following notation: given two pairs (p,q) and (i,j) of integers, $(p,q) \leq (i,j)$ is satisfied if $i \leq p$ and $q \leq j$ and given two integers p and q we define $p \cup q$ as p if q is undefined and as q if p is undefined, being undefined in other case.

We use $\beta \in \mathrm{adj}(N^\gamma)$ to denote that an auxiliary tree β may be adjoined at node N^γ of the tree γ. If adjunction is not mandatory at N^γ then $\mathbf{nil} \in \mathrm{adj}(N^\gamma)$ where

3. We remind you that TIG forbids adjunction at the root nodes of auxiliary trees.

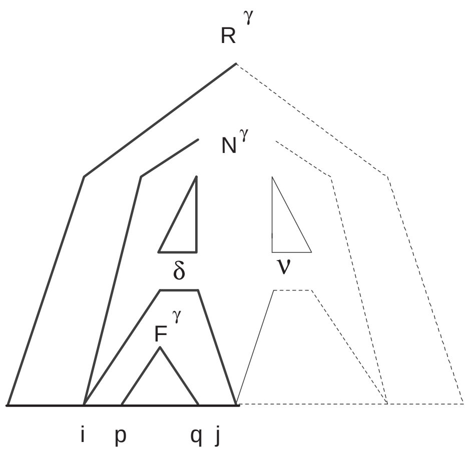

Figure 4. *Graphical representation of the items in the set* $\mathcal{I}_{\mathrm{Mix}_1}$

$\mathbf{nil} \notin \boldsymbol{I} \cup \boldsymbol{A}$ is a dummy symbol. If adjunction is not allowed at N^γ then $\{\mathbf{nil}\} = \mathrm{adj}(N^\gamma)$. We also use $\mathrm{label}(N^\gamma)$ to denote the label of a node N^γ belonging to an elementary tree γ.

2. Mixed parsing without simultaneous adjunctions

In this section we define a parsing system $\mathbb{P}_{\mathrm{Mix}_1} = \langle \mathcal{I}_{\mathrm{Mix}_1}, \mathcal{H}, \mathcal{D}_{\mathrm{Mix}_1} \rangle$ corresponding to an Earley-like TAG parser merged with an Earley-like TIG parser, in which the adjunction of strongly left and strongly right auxiliary trees will be managed by specialized deduction steps, the rest of adjunctions will be managed with the classical deduction steps included in most TAG parsers [ALO 99]. In this parsing algorithm, simultaneous adjunctions are not allowed. Thus, we follow the standard TAG definition of adjunction. With slight modifications, this parsing system corresponds to the parsing algorithm shown in [ALO 02].

2.1. *Items*

The items in the set $\mathcal{I}_{\mathrm{Mix}_1}$ are of the form $[N^\gamma \rightarrow \delta \bullet \nu, i, j \mid p, q \mid adj]$ such that $N^\gamma \rightarrow \delta\nu \in \mathcal{P}(\gamma)$, $\gamma \in \boldsymbol{I} \cup \boldsymbol{A}$, $0 \leq i \leq j \leq n$, $(p, q) = (-, -)$ or $(p, q) \leq (i, j)$, and $adj \in \{\mathrm{true}, \mathrm{false}\}$. The two indices with respect to the input string i and j indicate the portion of the input string that has been spanned from δ (see figure 4). If $\gamma \in \boldsymbol{A}$, p and q are two indices with respect to the input string that indicate that part of the input string recognized by the foot node of γ if it is a descendant of δ. In other case they are undefined, which is denoted by $p = q = -$. The last boolean component adj is needed to manage mandatory adjunction: $adj = \mathrm{true}$ if and only if an adjunction has taken place at N^γ, otherwise $adj = \text{false}$. Therefore, this kind of items satisfy one of the following conditions:

1) $\gamma \in \boldsymbol{A'}$, $(p,q) \neq (-,-)$ and $\delta \neq \varepsilon$ spans $a_{i+1} \dots a_p\ \mathbf{F}^\gamma\ a_{q+1} \dots a_j$

2) $\delta \neq \varepsilon$, $(p,q) = (-,-)$ and δ spans the string $a_{i+1} \dots a_j$.

3) $\delta = \varepsilon$, $(p,q) = (-,-)$, $i = j$, $adj = \text{false}$. The last boolean component indicates that any tree has been adjoined at N^γ.

4) $\delta = \varepsilon$, $(p,q) = (-,-)$, $adj = \text{true}$ and there exists a $\beta \in \boldsymbol{A}_{SL}$ such that $\beta \in \text{adj}(N^\gamma)$ and $\mathbf{R}^\beta$ spans $a_{i+1} \dots a_j$ (i.e., β has been adjoined at N^γ). In this case, i and j indicate the portion of the input string spanned by the left auxiliary tree adjoined at N^γ.

In this algorithm, the last boolean component of items is also used to control that at most one adjunction has been performed on a node. A value of true indicates that an adjunction has taken place on the node N^γ and therefore further adjunctions on the same node will be forbidden. A value of false indicates that no adjunction was performed on that node. In this case, during future processing this item can play the role of the item recognizing the excised part of an elementary tree to be attached to the foot node of a right auxiliary tree. As a consequence, only one adjunction can take place on a node, as is usual for TAG parsers.

2.2. *Deduction steps*

The set of deduction steps is formed by the following subsets:

$$\begin{aligned}\mathcal{D}_{\text{Mix}_1} = \ & \mathcal{D}^{\text{Init}}_{\text{Mix}_1} \cup \mathcal{D}^{\text{Scan}}_{\text{Mix}_1} \cup \mathcal{D}^{\varepsilon}_{\text{Mix}_1} \cup \mathcal{D}^{\text{Pred}}_{\text{Mix}_1} \cup \mathcal{D}^{\text{Comp}}_{\text{Mix}_1} \cup \\ & \mathcal{D}^{\text{AdjPred}}_{\text{Mix}_1} \cup \mathcal{D}^{\text{FootPred}}_{\text{Mix}_1} \cup \mathcal{D}^{\text{FootComp}}_{\text{Mix}_1} \cup \mathcal{D}^{\text{AdjComp}}_{\text{Mix}_1} \cup \\ & \mathcal{D}^{\text{LAdjPred}}_{\text{Mix}_1} \cup \mathcal{D}^{\text{LAdjComp}}_{\text{Mix}_1} \cup \mathcal{D}^{\text{RAdjPred}}_{\text{Mix}_1} \cup \mathcal{D}^{\text{RAdjComp}}_{\text{Mix}_1} \cup \mathcal{D}^{\text{LRFoot}}_{\text{Mix}_1}\end{aligned}$$

The parsing process starts by creating the items corresponding to productions having the root of an initial tree as right-hand side and the dot in the leftmost position of the right-hand side:

$$\mathcal{D}^{\text{Init}}_{\text{Mix}_1} = \frac{}{[\top \rightarrow \bullet \mathbf{R}^\alpha, 0, 0 \mid -, - \mid \text{false}]} \quad \alpha \in \boldsymbol{I} \ \wedge \ S = \text{label}(\mathbf{R}^\alpha)$$

A set of deductive steps in $\mathcal{D}^{\text{Pred}}_{\text{Mix}_1}$ and $\mathcal{D}^{\text{Comp}}_{\text{Mix}_1}$ traverse each elementary tree while steps in $\mathcal{D}^{\text{Scan}}_{\text{Mix}_1}$ and $\mathcal{D}^{\varepsilon}_{\text{Mix}_1}$ scan input symbols and the empty symbol, respectively:

$$\mathcal{D}^{\text{Pred}}_{\text{Mix}_1} = \frac{[N^\gamma \rightarrow \delta \bullet M^\gamma \nu, i, j \mid p, q \mid adj]}{[M^\gamma \rightarrow \bullet \upsilon, j, j \mid -, - \mid \text{false}]} \quad \begin{array}{l}\textbf{nil} \in \text{adj}(M^\gamma) \ \vee \\ (\exists \beta \in \boldsymbol{A}_{SL} \cup \boldsymbol{A}_{SR},\ \beta \in \text{adj}(M^\gamma))\end{array}$$

$$\mathcal{D}^{\text{Comp}}_{\text{Mix}_1} = \frac{\begin{array}{c}[N^\gamma \rightarrow \delta \bullet M^\gamma \nu, i, j \mid p, q \mid adj],\\ [M^\gamma \rightarrow \upsilon \bullet, j, k \mid p', q' \mid adj']\end{array}}{[N^\gamma \rightarrow \delta M^\gamma \bullet \nu, i, k \mid p \cup p', q \cup q' \mid adj]}$$

$$\begin{array}{l}\text{with } (\mathbf{nil} \in \text{adj}(M^\gamma) \ \wedge \ adj' = \text{false}) \ \vee \\ \quad (\exists \beta \in \boldsymbol{A},\ \beta \in \text{adj}(M^\gamma) \ \wedge \ adj' = \text{true})\end{array}$$

$$\mathcal{D}^{\text{Scan}}_{\text{Mix}_1} = \frac{\begin{array}{l}[N^\gamma \rightarrow \delta \bullet M^\gamma \nu, i, j \mid p, q \mid adj],\\ [a, j, j+1]\end{array}}{[N^\gamma \rightarrow \delta M^\gamma \bullet \nu, i, j+1 \mid p, q \mid adj]} \ \ a = \text{label}(M^\gamma)$$

$$\mathcal{D}^{\varepsilon}_{\text{Mix}_1} = \frac{[N^\gamma \rightarrow \delta \bullet M^\gamma \nu, i, j \mid p, q \mid adj]}{[N^\gamma \rightarrow \delta M^\gamma \bullet \nu, i, j \mid p, q \mid adj]} \ \ \varepsilon = \text{label}(M^\gamma)$$

The rest of steps are in charge of managing adjunction operations. If a strongly left auxiliary tree $\beta \in \boldsymbol{A}_{SL}$ can be adjoined at a given node M^γ, a step in $\mathcal{D}^{\text{LAdjPred}}_{\text{Mix}_1}$ starts the traversal of β. When β has been completely traversed, a step in $\mathcal{D}^{\text{LAdjComp}}_{\text{Mix}_1}$ starts the traversal of the subtree corresponding to M^γ and sets the last element of the item to true in order to forbid further adjunctions on this node.

$$\mathcal{D}^{\text{LAdjPred}}_{\text{Mix}_1} = \frac{[M^\gamma \rightarrow \bullet \upsilon, i, i \mid -, - \mid \text{false}]}{[\top \rightarrow \bullet \mathbf{R}^\beta, i, i \mid -, - \mid \text{false}]} \ \ \beta \in \text{adj}(M^\gamma) \ \wedge \ \beta \in \boldsymbol{A}_{SL}$$

$$\mathcal{D}^{\text{LAdjComp}}_{\text{Mix}_1} = \frac{\begin{array}{l}[M^\gamma \rightarrow \bullet \upsilon, i, i \mid -, - \mid \text{false}],\\ [\top \rightarrow \mathbf{R}^\beta \bullet, i, j \mid -, - \mid \text{false}]\end{array}}{[M^\gamma \rightarrow \bullet \upsilon, i, j \mid -, - \mid \text{true}]} \ \ \beta \in \boldsymbol{A}_{SL} \ \wedge \ \beta \in \text{adj}(M^\gamma)$$

If a strongly right auxiliary tree $\beta \in \boldsymbol{A}_{SR}$ can be adjoined at a given node M^γ, when the subtree corresponding to this node has been completely traversed, a step in $\mathcal{D}^{\text{RAdjPred}}_{\text{Mix}_1}$ starts the traversal of the tree β. When β has been completely traversed, a step in $\mathcal{D}^{\text{RAdjComp}}_{\text{Mix}_1}$ updates the input positions spanned by M^γ taking into account the part of the input string spanned by β, and sets the last element of the item to true in order to forbid further adjunctions on this node.

$$\mathcal{D}^{\text{RAdjPred}}_{\text{Mix}_1} = \frac{[M^\gamma \rightarrow \upsilon \bullet, i, j \mid p, q \mid \text{false}]}{[\top \rightarrow \bullet \mathbf{R}^\beta, j, j \mid -, - \mid \text{false}]} \ \ \beta \in \boldsymbol{A}_{SR} \ \wedge \ \beta \in \text{adj}(M^\gamma)$$

$$\mathcal{D}^{\text{RAdjComp}}_{\text{Mix}_1} = \frac{\begin{array}{l}[M^\gamma \rightarrow \upsilon \bullet, i, j \mid p, q \mid \text{false}],\\ [\top \rightarrow \mathbf{R}^\beta \bullet, j, k \mid -, - \mid \text{false}]\end{array}}{[M^\gamma \rightarrow \upsilon \bullet, i, k \mid p, q \mid \text{true}]} \ \ \beta \in \boldsymbol{A}_{SR} \ \wedge \ \beta \in \text{adj}(M^\gamma)$$

No special treatment is given to the foot node of strongly left and right auxiliary trees and so, it is simply skipped by a step in the set $\mathcal{D}^{\text{LRFoot}}_{\text{Mix}_1}$.

$$\mathcal{D}^{\text{LRFoot}}_{\text{Mix}_1} = \frac{[\mathbf{F}^\beta \to \bullet\bot, j, j, adj]}{[\mathbf{F}^\beta \to \bot\bullet, j, j, adj]} \;\; \beta \in \boldsymbol{A}_{SL} \cup \boldsymbol{A}_{SR}$$

A step in $\mathcal{D}^{\text{AdjPred}}_{\text{Mix}_1}$ predicts the adjunction of an auxiliary tree $\beta \in \boldsymbol{A'}$ at a node of an elementary tree γ and starts the traversal of β. Once the foot of β has been reached, the traversal of β is momentary suspended by a step in $\mathcal{D}^{\text{FootPred}}_{\text{Mix}_1}$, which re-takes the subtree of γ which must be attached to the foot of β. At this moment, there is no information available about the node in which the adjunction of β has been performed, so all possible nodes are predicted. When the traversal of a predicted subtree has finished, a step in $\mathcal{D}^{\text{FootComp}}_{\text{Mix}_1}$ re-takes the traversal of β continuing at the foot node. When the traversal of β is completely finished, a deduction step in $\mathcal{D}^{\text{AdjComp}}_{\text{Mix}_1}$ checks if the subtree attached to the foot of β corresponds with the adjunction node. The traversal of M^γ (and therefore the adjunction of β at M^γ) is finished by a step in $\mathcal{D}^{\text{Comp}}_{\text{Mix}_1}$, taking into account that p' and q' are instantiated if and only if the adjunction node is on the spine of γ. It is interesting to remark that we follow the approach of [NED 99], splitting the completion of an adjunction between $\mathcal{D}^{\text{AdjComp}}_{\text{Mix}_1}$ and $\mathcal{D}^{\text{Comp}}_{\text{Mix}_1}$.

$$\mathcal{D}^{\text{AdjPred}}_{\text{Mix}_1} = \frac{[N^\gamma \to \delta \bullet M^\gamma \nu, i, j \mid p, q \mid adj]}{[\top \to \bullet\mathbf{R}^\beta, j, j \mid -, - \mid \text{false}]} \;\; \beta \in \boldsymbol{A'} \;\wedge\; \beta \in \text{adj}(M^\gamma)$$

$$\mathcal{D}^{\text{FootPred}}_{\text{Mix}_1} = \frac{[\mathbf{F}^\beta \to \bullet\bot, k, k \mid -, - \mid \text{false}]}{[M^\gamma \to \bullet\upsilon, k, k \mid -, - \mid \text{false}]} \;\; \beta \in \boldsymbol{A'} \;\wedge\; \beta \in \text{adj}(M^\gamma)$$

$$\mathcal{D}^{\text{FootComp}}_{\text{Mix}_1} = \frac{\begin{array}{c}[\mathbf{F}^\beta \to \bullet\bot, k, k \mid -, - \mid \text{false}],\\ [M^\gamma \to \upsilon\bullet, k, l \mid p', q' \mid \text{false}]\end{array}}{[\mathbf{F}^\beta \to \bot\bullet, k, l \mid k, l \mid \text{false}]} \;\; \beta \in \boldsymbol{A'} \;\wedge\; \beta \in \text{adj}(M^\gamma)$$

$$\mathcal{D}^{\text{AdjComp}}_{\text{Mix}_1} = \frac{\begin{array}{c}[\top \to \mathbf{R}^\beta\bullet, j, m \mid k, l \mid \text{false}],\\ [M^\gamma \to \upsilon\bullet, k, l \mid p', q' \mid \text{false}]\end{array}}{[M^\gamma \to \upsilon\bullet, j, m \mid p', q' \mid \text{true}]} \;\; \beta \in \boldsymbol{A'} \;\wedge\; \beta \in \text{adj}(M^\gamma)$$

The input string belongs to the language defined by the grammar if a final item in the set $\mathcal{F} = \left\{ \; [\top \to \mathbf{R}^\alpha\bullet, 0, n \mid -, - \mid \text{false}] \mid \alpha \in \boldsymbol{I} \;\wedge\; S = \text{label}(\mathbf{R}^\alpha) \; \right\}$ is generated.

3. Mixed parsing with simultaneous adjunctions

Let us consider now that the trees in Figure 3 define a TAG. In this case, to generate the language a^*bc^* we need to perform several adjunctions of β_L and β_R at their root nodes. When parsing a sentence, derived trees are obtained using the expensive TAG adjunction operation although we know the similarities of this TAG grammar with a TIG grammar. In fact, allowing simultaneous adjunction and disabling adjunction at the root nodes of β_L and β_R, the same set of derived trees would be produced using the cheaper TIG adjunction operation. Whenever we have a sufficient number of auxiliary trees with this property in a TAG grammar, we can exploit the benefits of TIG adjunction allowing simultaneous adjunction and disabling the adjunctions at the root nodes of auxiliary trees.[4] For example, we can note that determiners or adjectives are usually modeled in XTAG with left auxiliary trees but relative clauses are modeled with right auxiliary trees. Whenever we need to modify a noun with both determiners and relative clauses we can combine left and right auxiliary trees in this way. The interesting point is that 93% of the spines of the auxiliary trees in XTAG contain only the root and the foot node, so this modification can help to improve parsing performance.

In this section we define a parsing system $\mathbb{P}_{\mathrm{Mix}_2} = \langle \mathcal{I}_{\mathrm{Mix}_2}, \mathcal{H}, \mathcal{D}_{\mathrm{Mix}_2} \rangle$ corresponding to a mixed parsing algorithm for TAG and TIG in which simultaneous adjunctions are allowed on any node, with the following ordering: the adjunction of strongly left auxiliary trees will take place before the adjunction of other types of trees. This ordering has been established for compatibility with the definition of simultaneous adjunctions in TIG [SCH 95]. With slightly modifications, this parsing system corresponds to the parsing algorithm shown in [ALO 03].

3.1. *Items and deduction steps*

Items in the set $\mathcal{I}_{\mathrm{Mix}_2}$ have the same form than items in the set $\mathcal{I}_{\mathrm{Mix}_1}$. However, given that more than one tree is allowed to be adjoined at a given node, the last boolean component adj has true as value if and only if one or more adjunctions have taken place at N^γ, otherwise $adj = \mathrm{false}$. In particular, i and j will indicate the portion of the input string spanned by the strongly left auxiliary trees adjoined at N^γ if there exists a sequence of strongly auxiliary trees that have been adjoined at N^γ, $\delta = \varepsilon$, $(p, q) = (-, -)$ and $adj = \mathrm{true}$.

The set of deduction steps is formed by the following subsets:

$$\begin{aligned}
\mathcal{D}_{\mathrm{Mix}_2} = \quad & \mathcal{D}^{\mathrm{Init}}_{\mathrm{Mix}_1} \cup \mathcal{D}^{\mathrm{Scan}}_{\mathrm{Mix}_1} \cup \mathcal{D}^{\varepsilon}_{\mathrm{Mix}_1} \cup \mathcal{D}^{\mathrm{Pred}}_{\mathrm{Mix}_2} \cup \mathcal{D}^{\mathrm{Comp}}_{\mathrm{Mix}_2} \cup \\
& \mathcal{D}^{\mathrm{LAdjPred}}_{\mathrm{Mix}_2} \cup \mathcal{D}^{\mathrm{LAdjComp}}_{\mathrm{Mix}_2} \cup \mathcal{D}^{\mathrm{RAdjPred}}_{\mathrm{Mix}_2} \cup \mathcal{D}^{\mathrm{RAdjComp}}_{\mathrm{Mix}_2} \cup \mathcal{D}^{\mathrm{LRFoot}}_{\mathrm{Mix}_1} \cup \\
& \mathcal{D}^{\mathrm{AdjPred}}_{\mathrm{Mix}_2} \cup \mathcal{D}^{\mathrm{FootPred}}_{\mathrm{Mix}_2} \cup \mathcal{D}^{\mathrm{FootComp}}_{\mathrm{Mix}_2} \cup \mathcal{D}^{\mathrm{AdjComp}}_{\mathrm{Mix}_2} \cup \mathcal{D}^{\mathrm{Comb}}_{\mathrm{Mix}_2}
\end{aligned}$$

4. Simultaneous adjuntion does not increase the generative capability of TAG due to the simultaneous adjunction of a set of auxiliary trees on a given node can be simulated by an adjunction at that node followed by a sequence of (traditional, non simultaneous) adjunctions at the root nodes of the auxiliary trees.

The starting of the parsing process and the scanning of terminal symbols and the empty string is performed as in $\mathbb{P}_{\mathrm{Mix}_1}$. In contrast, prediction and completion are performed differently: steps in $\mathcal{D}^{\mathrm{Pred}}_{\mathrm{Mix}_2}$ do not need to check any condition, while steps in $\mathcal{D}^{\mathrm{Comp}}_{\mathrm{Mix}_2}$ must ensure that mandatory adjunction and forbidden adjunction constraints ($\mathbf{nil} \notin \mathrm{adj}(M^\gamma)$ and $\{\mathbf{nil}\} = \mathrm{adj}(M^\gamma)$, respectively) are satisfied.

$$\mathcal{D}^{\mathrm{Pred}}_{\mathrm{Mix}_2} = \frac{[N^\gamma \to \delta \bullet M^\gamma \nu, i, j \mid p, q \mid adj]}{[M^\gamma \to \bullet \upsilon, j, j \mid -, - \mid \mathrm{false}]}$$

$$\mathcal{D}^{\mathrm{Comp}}_{\mathrm{Mix}_2} = \frac{\begin{array}{c}[N^\gamma \to \delta \bullet M^\gamma \nu, i, j \mid p, q \mid adj],\\ [M^\gamma \to \upsilon \bullet, j, k \mid p', q' \mid adj']\end{array}}{[N^\gamma \to \delta M^\gamma \bullet \nu, i, k \mid p \cup p', q \cup q' \mid adj]} \quad \begin{array}{l} adj' = \mathrm{true} \text{ if } \mathbf{nil} \notin \mathrm{adj}(M^\gamma) \\ adj' = \mathrm{false} \text{ if } \{\mathbf{nil}\} = \mathrm{adj}(M^\gamma)\end{array}$$

In left adjunctions, the value of the boolean component in the first antecedent item of steps in $\mathcal{D}^{\mathrm{LAdjPred}}_{\mathrm{Mix}_2}$ and $\mathcal{D}^{\mathrm{LAdjComp}}_{\mathrm{Mix}_2}$ is not relevant. Simultaneous adjunctions of several strongly left auxiliary trees on a node M^γ is achieved by applying a pair of steps $\mathcal{D}^{\mathrm{LAdjPred}}_{\mathrm{Mix}_2}$ and $\mathcal{D}^{\mathrm{LAdjComp}}_{\mathrm{Mix}_2}$ for each auxiliary tree.

$$\mathcal{D}^{\mathrm{LAdjPred}}_{\mathrm{Mix}_2} = \frac{[M^\gamma \to \bullet \upsilon, i, j \mid -, - \mid adj]}{[\top \to \bullet \mathbf{R}^\beta, j, j \mid -, - \mid \mathrm{false}]} \quad \beta \in \mathrm{adj}(M^\gamma) \ \wedge \ \beta \in \boldsymbol{A}_{SL}$$

$$\mathcal{D}^{\mathrm{LAdjComp}}_{\mathrm{Mix}_2} = \frac{\begin{array}{c}[M^\gamma \to \bullet \upsilon, i, j \mid -, - \mid adj],\\ [\top \to \mathbf{R}^\beta \bullet, j, k \mid -, - \mid \mathrm{false}]\end{array}}{[M^\gamma \to \bullet \upsilon, i, k \mid -, - \mid \mathrm{true}]} \quad \beta \in \boldsymbol{A}_{SL} \ \wedge \ \beta \in \mathrm{adj}(M^\gamma)$$

A similar modification must be performed for deduction steps in charge of dealing with right adjunctions. Simultaneous adjunctions of several strongly right auxiliary trees on a node M^γ is achieved by applying a pair of steps $\mathcal{D}^{\mathrm{RAdjPred}}_{\mathrm{Mix}_2}$ and $\mathcal{D}^{\mathrm{RAdjComp}}_{\mathrm{Mix}_2}$ for each auxiliary tree.

$$\mathcal{D}^{\mathrm{RAdjPred}}_{\mathrm{Mix}_2} = \frac{[M^\gamma \to \upsilon \bullet, i, j \mid p, q \mid adj]}{[\top \to \bullet \mathbf{R}^\beta, j, j \mid -, - \mid \mathrm{false}]} \quad \beta \in \boldsymbol{A}_{SR} \ \wedge \ \beta \in \mathrm{adj}(M^\gamma)$$

$$\mathcal{D}^{\mathrm{RAdjComp}}_{\mathrm{Mix}_2} = \frac{\begin{array}{c}[M^\gamma \to \upsilon \bullet, i, j \mid p, q \mid adj],\\ [\top \to \mathbf{R}^\beta \bullet, j, k \mid -, - \mid \mathrm{false}]\end{array}}{[M^\gamma \to \upsilon \bullet, i, k \mid p, q \mid \mathrm{true}]} \quad \beta \in \boldsymbol{A}_{SR} \ \wedge \ \beta \in \mathrm{adj}(M^\gamma)$$

The traversal of an auxiliary tree $\beta \in \boldsymbol{A'}$ that can be adjoined at a node M^γ is started once the traversal of the production $M^\gamma \to \bullet \upsilon$ has been started by a $\mathcal{D}^{\mathrm{Pred}}_{\mathrm{Mix}_2}$ step. This way we make possible to adjoin at M^γ several strongly left auxiliary trees prior to β. Deduction steps in $\mathcal{D}^{\text{FootPred}}_{\text{Mix}_2}$, $\mathcal{D}^{\text{FootComp}}_{\text{Mix}_2}$ and $\mathcal{D}^{\text{AdjComp}}_{\text{Mix}_2}$ perform tasks analogous to those of $\mathcal{D}^{\text{FootPred}}_{\text{Mix}_1}$, $\mathcal{D}^{\text{FootComp}}_{\text{Mix}_1}$ and $\mathcal{D}^{\text{AdjComp}}_{\text{Mix}_1}$, respectively.

$$\mathcal{D}^{\text{AdjPred}}_{\text{Mix}_2} = \frac{[M^\gamma \rightarrow \bullet\upsilon, i, j \mid -, - \mid \mathit{adj}]}{[\top \rightarrow \bullet\mathbf{R}^\beta, j, j \mid -, - \mid \text{false}]} \;\; \beta \in \boldsymbol{A'} \;\wedge\; \beta \in \text{adj}(M^\gamma)$$

$$\mathcal{D}^{\text{FootPred}}_{\text{Mix}_2} = \frac{[\mathbf{F}^\beta \rightarrow \bullet\bot, k, k \mid -, - \mid \mathit{adj}]}{[M^\gamma \rightarrow \bullet\upsilon, k, k \mid -, - \mid \text{false}]} \;\; \beta \in \boldsymbol{A'} \;\wedge\; \beta \in \text{adj}(M^\gamma)$$

$$\mathcal{D}^{\text{FootComp}}_{\text{Mix}_2} = \frac{\begin{array}{c}[\mathbf{F}^\beta \rightarrow \bullet\bot, l, l \mid -, - \mid \mathit{adj}],\\ [M^\gamma \rightarrow \upsilon\bullet, l, m \mid p', q' \mid \mathit{adj}']\end{array}}{[\mathbf{F}^\beta \rightarrow \bot\bullet, l, m \mid l, m \mid \mathit{adj}]} \;\; \beta \in \boldsymbol{A'} \;\wedge\; \beta \in \text{adj}(M^\gamma)$$

$$\mathcal{D}^{\text{AdjComp}}_{\text{Mix}_2} = \frac{\begin{array}{c}[\top \rightarrow \mathbf{R}^\beta\bullet, k, r \mid l, m \mid \text{false}],\\ [M^\gamma \rightarrow \upsilon\bullet, l, m \mid p', q' \mid \mathit{adj}]\end{array}}{[M^\gamma \rightarrow \upsilon\bullet, k, r \mid p', q' \mid \text{true}]} \;\; \beta \in \boldsymbol{A'} \;\wedge\; \beta \in \text{adj}(M^\gamma)$$

Simultaneous adjunctions of several auxiliary trees in $\boldsymbol{A'}$ is achieved by using the consequent item generated by a deduction step in $\mathcal{D}^{\text{FootPred}}_{\text{Mix}_2}$ as antecedent of a deduction step in $\mathcal{D}^{\text{AdjPred}}_{\text{Mix}_2}$ to start the adjunction of an auxiliary tree $\beta' \in \boldsymbol{A'}$. When the traversal of β' has finished, a step in $\mathcal{D}^{\text{FootComp}}_{\text{Mix}_2}$ re-takes the traversal of β at the foot node. The process is repeated for each auxiliary tree which is to be simultaneously adjoined.

The subset $\mathcal{D}^{\text{Comb}}_{\text{Mix}_2}$ is needed to put together the results corresponding to the simultaneous adjunctions of strongly left and wrapping auxiliary trees:

$$\mathcal{D}^{\text{Comb}}_{\text{Mix}_2} = \frac{\begin{array}{c}[M^\gamma \rightarrow \bullet\upsilon, i, j \mid -, - \mid \text{true}],\\ [M^\gamma \rightarrow \upsilon\bullet, j, k \mid p, q \mid \text{true}]\end{array}}{[M^\gamma \rightarrow \upsilon\bullet, i, k \mid p, q \mid \text{true}]}$$

The input string belongs to the language defined by the grammar if a final item in the set $\mathcal{F} = \left\{ \; [\top \rightarrow \mathbf{R}^\alpha\bullet, 0, n \mid -, - \mid \text{false}] \mid \alpha \in \boldsymbol{I} \;\wedge\; S = \text{label}(\mathbf{R}^\alpha) \; \right\}$ is generated.

3.2. *An example of parsing*

The behavior of this algorithm is illustrated by means of an example. Figure 5 shows the adjunction of a strongly-left auxiliary tree β_{l1}, a strongly right auxiliary tree β_{r1}, two wrapping trees β_{w1} and β_{w2}, a strongly-left auxiliary tree β_{l2} and a strongly right auxiliary tree β_{r2}, enumerated in a top-down view of the resulting derived tree, which is obtained as follows:

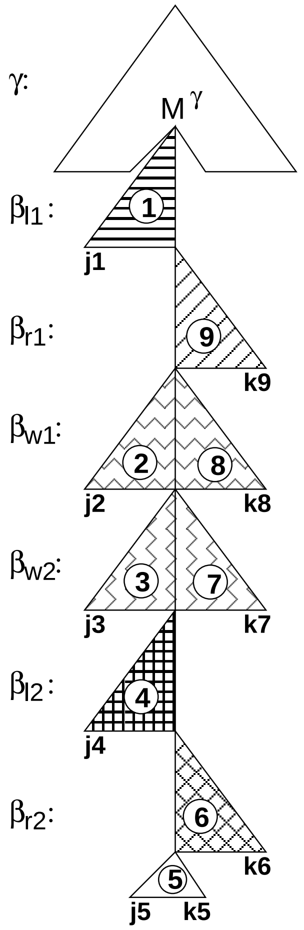

Figure 5. *An example of simultaneous adjunctions*

1) Once the adjunction node M^γ is reached at position j_1, a step in $\mathcal{D}^{\text{Pred}}_{\text{Mix}_2}$ generates the item $[M^\gamma \rightarrow \bullet\upsilon, j_1, j_1 \mid -, - \mid \text{false}]$. Then, a step in $\mathcal{D}^{\text{LAdjPred}}_{\text{Mix}_2}$ is applied in order to start the adjunction of β_{l1}, which is finished by a step in $\mathcal{D}^{\text{LAdjComp}}_{\text{Mix}_2}$ that generates the item $[M^\gamma \rightarrow \bullet\upsilon, j_1, j_2 \mid -, - \mid \text{true}]$.

2) Strongly right auxiliary trees do not span anything to the left of their spine, therefore no action is performed with respect to β_{r1} at this moment. Instead, a step in $\mathcal{D}^{\text{AdjPred}}_{\text{Mix}_2}$ predicts the adjunction of the auxiliary tree β_{w1}, generating the item $[\top \rightarrow \bullet\mathbf{R}^{\beta_{w1}}, j_2, j_2 \mid -, - \mid \text{false}]$.

3) When the foot node of β_{w1} is reached at position j_3, a step in $\mathcal{D}^{\text{FootPred}}_{\text{Mix}_2}$ generates the item $[M^\gamma \rightarrow \bullet\upsilon, j_3, j_3 \mid -, - \mid \text{false}]$. A deduction step in $\mathcal{D}^{\text{AdjPred}}_{\text{Mix}_2}$ takes this item as antecedent and starts the adjunction of β_{w2}, generating the item

$[\top \rightarrow \bullet\mathbf{R}^{\beta_{w2}}, j_3, j_3 \mid -, - \mid \text{false}]$. When the foot node of β_{w2} is reached at position j_4, the traversal of γ is re-taken at M^γ by means of the application of a step in $\mathcal{D}^{\text{FootPred}}_{\text{Mix}_2}$, generating the consequent item $[M^\gamma \rightarrow \bullet v, j_4, j_4 \mid -, - \mid \text{false}]$.

4) The adjunction of β_{l2} is then predicted by a deduction step in $\mathcal{D}^{\text{LAdjPred}}_{\text{Mix}_2}$. The completion of this adjunction by a step in $\mathcal{D}^{\text{LAdjComp}}_{\text{Mix}_2}$ gives as a result the generation of the item $[M^\gamma \rightarrow \bullet v, j_4, j_5 \mid -, - \mid \text{true}]$. It is interesting to remark that the ordering imposed on the trees involved in simultaneous adjunctions has been preserved due to the adjunctions of β_{l1} and β_{l2} have been completely performed before the adjunction of other types of auxiliary trees.

5) β_{r2} is not considered at this moment. Once the subtree rooted by M^γ has been completely traversed, we get the item $[M^\gamma \rightarrow v\bullet, j_4, k_5 \mid -, - \mid \text{true}]$.

6) At this moment, a step in $\mathcal{D}^{\text{RAdjPred}}_{\text{Mix}_2}$ starts the adjunction of β_{r2} by generating the item $[\top \rightarrow \bullet\mathbf{R}^{\beta_{r2}}, k_5, k_5 \mid -, - \mid \text{false}]$. When a step in $\mathcal{D}^{\text{RAdjComp}}_{\text{Mix}_2}$ performs the completion of this adjunction, the item $[M^\gamma \rightarrow \bullet v, j_4, k_6 \mid -, - \mid \text{true}]$ is generated.

7) At this point, a step in $\mathcal{D}^{\text{FootComp}}_{\text{Mix}_2}$ re-takes the traversal of β_{w2}, generating the item $[\mathbf{F}^{\beta_{w2}} \rightarrow \perp\bullet, j_4, k_6 \mid j_4, k_6 \mid \text{false}]$ which means that the subtree corresponding to the adjunction node of this auxiliary tree is expected to span the substring $a_{j_4+1} \ldots a_{k_6}$. The complete traversal of β_{w2} is indicated by the item $[\top \rightarrow \mathbf{R}^{\beta_{w2}}\bullet, j_3, k_7 \mid j_4, k_6 \mid \text{false}]$, which is used by a step in $\mathcal{D}^{\text{AdjComp}}_{\text{Mix}_2}$ to generate the item $[M^\gamma \rightarrow v\bullet, j_3, k_7 \mid -, - \mid \text{true}]$ indicating that the adjunction corresponding to β_{w2} has been completed.

8) Then, a step in $\mathcal{D}^{\text{FootComp}}_{\text{Mix}_2}$ is in charge of re-taking the traversal of β_{w1}, generating the item $[\mathbf{F}^{\beta_{w1}} \rightarrow \perp\bullet, j_3, k_7 \mid j_3, k_7 \mid \text{false}]$ which means that the subtree corresponding to the adjunction node of this auxiliary tree is expected to span the substring $a_{j_3+1} \ldots a_{k_7}$. The adjunction of β_{w1} is finished by a step in $\mathcal{D}^{\text{AdjComp}}_{\text{Mix}_2}$, yielding the item $[M^\gamma \rightarrow v\bullet, j_2, k_8 \mid -, - \mid \text{true}]$.

9) At this moment we have two possibilities in order to adjoin β_{r1}:

a) A step in $\mathcal{D}^{\text{Comb}}_{\text{Mix}_2}$ combines the item $[M^\gamma \rightarrow \bullet v, j_1, j_2 \mid -, - \mid \text{true}]$ and the item $[M^\gamma \rightarrow v\bullet, j_2, k_8 \mid -, - \mid \text{true}]$ in order to obtain the consequent item $[M^\gamma \rightarrow v\bullet, j_1, k_8 \mid -, - \mid \text{true}]$. Then, the adjunction of β_{r1} can be predicted by a step in $\mathcal{D}^{\text{RAdjPred}}_{\text{Mix}_2}$. Once this strongly right auxiliary tree has been completely traversed, the item $[M^\gamma \rightarrow v\bullet, j_1, k_9 \mid -, - \mid \text{true}]$ is generated by a a step in $\mathcal{D}^{\text{RAdjComp}}_{\text{Mix}_2}$.

b) A step in $\mathcal{D}^{\text{RAdjPred}}_{\text{Mix}_2}$ starts the adjunction of β_{r1}. Once this auxiliary tree has been completely traversed, the item $[M^\gamma \rightarrow v\bullet, j_2, k_9 \mid -, - \mid \text{true}]$ is generated by a step in $\mathcal{D}^{\text{RAdjComp}}_{\text{Mix}_2}$. Then, a step in $\mathcal{D}^{\text{Comb}}_{\text{Mix}_2}$ combines the items $[M^\gamma \rightarrow \bullet v, j_1, j_2 \mid -, - \mid \text{true}]$ and $[M^\gamma \rightarrow v\bullet, j_2, k_9 \mid -, - \mid \text{true}]$ to obtain the item $[M^\gamma \rightarrow v\bullet, j_1, k_9 \mid -, - \mid \text{true}]$.

This spurious ambiguity could be eliminated by imposing a more restrictive ordering of trees in simultaneous adjunctions: one possibility is to force that trees in $\boldsymbol{A'}$ should be adjoined first, then trees in $\boldsymbol{A}_{SL}$ and finally trees in $\boldsymbol{A}_{SR}$; other possibility is to force that trees in $\boldsymbol{A}_{SL}$ should be adjoined first, then trees in $\boldsymbol{A'}$ and finally trees in $\boldsymbol{A}_{SR}$.

4. Mixed parsing preserving the correct prefix property

Parsers satisfying the *correct prefix property* 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 $a_1 \ldots a_k$ read from the input string $a_1 \ldots a_k a_{k+1} \ldots a_n$ guarantees that there exists a string of tokens $b_1 \ldots b_m$, where b_i need not be part of the input string, such that $a_1 \ldots a_k b_1 \ldots b_m$ is a valid string of the language.

In this section we define a parsing system $\mathbb{P}_{\text{Mix}_3} = \langle \mathcal{I}_{\text{Mix}_3}, \mathcal{H}, \mathcal{D}_{\text{Mix}_3} \rangle$ corresponding to a mixed parsing algorithm for TAG and TIG preserving the correct prefix property and allowing simultaneous adjunctions on any node.

4.1. *Items*

We adapt the approach of Nederhof in [NED 99], adding a new position h to items corresponding to auxiliary trees in $\boldsymbol{A'}$. This new element is used to indicate the position of the input string corresponding to the left-most extreme of the frontier of the tree to which the dotted rule in the item belongs. To facilitate the understanding of items, we consider $\mathcal{I}_{\text{Mix}_3}$ as formed by the union of the following six subsets:

$\mathcal{I}^{(1a)}_{\text{Mix}_3}$**:** A subset with items of the form $[-, N^\gamma \rightarrow \bullet \upsilon, i, j \mid -, - \mid adj]$ such that $N^\gamma \rightarrow \upsilon \in \mathcal{P}(\gamma)$, $\gamma \in \boldsymbol{I} \cup \boldsymbol{A}_{SL} \cup \boldsymbol{A}_{SR}$, $0 \leq i \leq j$, and $adj \in \{\text{true}, \text{false}\}$. The last boolean component is used to control adjunctions: $adj = \text{true}$ if and only if a strongly left auxiliary tree has been adjoined at N^γ, otherwise $adj = \text{false}$ and $i = j$. If an adjunction has taken place at N^γ, i and j indicate the portion of the input string spanned by the strongly left auxiliary trees adjoined at N^γ.

$\mathcal{I}^{(1b)}_{\text{Mix}_3}$**:** A subset with items of the form $[-, N^\gamma \rightarrow \delta \bullet \nu, i, j \mid -, - \mid adj]$ such that $N^\gamma \rightarrow \delta\nu \in \mathcal{P}(\gamma)$, $\gamma \in \boldsymbol{I} \cup \boldsymbol{A}_{SL} \cup \boldsymbol{A}_{SR}$, $0 \leq i \leq j$, and $adj \in \{\text{true}, \text{false}\}$. The two indices with respect to the input string i and j indicate the portion of the input string spanned by δ. The boolean component adj is needed to manage mandatory adjunction: $adj = \text{true}$ if and only if one or more adjunctions have taken place at N^γ, otherwise $adj = \text{false}$. It is interesting to remark that if the auxiliary tree adjoined at N^γ belongs to $\boldsymbol{A}_{SL}$, the part of the input string spanned by that tree is a prefix of $a_{i+1} \ldots a_j$.

$\mathcal{I}^{(2a)}_{\text{Mix}_3}$**:** A subset with items of the form $[h, N^\beta \rightarrow \bullet \upsilon, i, j \mid -, - \mid adj]$ such that $N^\beta \rightarrow \upsilon \in \mathcal{P}(\beta)$, $\beta \in \boldsymbol{A'}$, $0 \leq h \leq i \leq j$, and $adj \in \{\text{true}, \text{false}\}$. These items are similar to items in the subset $\mathcal{I}^{(1a)}_{\text{Mix}_3}$, except for the fact that now the tree involved in each item belongs to $\boldsymbol{A'}$. Therefore, the value of adj has the same meaning than for items in $\mathcal{I}^{(1a)}_{\text{Mix}_3}$ but now the value of h must be set to the position of the input string at which the traversal of β was started. For trees in $\boldsymbol{A'}$, the position h is needed to ensure the correct prefix property is preserved at the time of predicting the subtrees pending from their foot nodes.

$\mathcal{I}^{(2b)}_{\text{Mix}_3}$**:** A subset with items of the form $[h, N^\beta \rightarrow \delta \bullet \nu, i, j \mid -, - \mid adj]$ such that $N^\beta \rightarrow \delta\nu \in \mathcal{P}(\beta)$, $\beta \in \boldsymbol{A'}$, $0 \le h \le i \le j$, $adj \in \{\text{true}, \text{false}\}$ and the foot node of β is not a descendant of any node in δ. These items are similar to items in the subset $\mathcal{I}^{(1b)}_{\text{Mix}_3}$, in particular the value of adj has the same meaning than for items in $\mathcal{I}^{(1b)}_{\text{Mix}_3}$.

$\mathcal{I}^{(2c)}_{\text{Mix}_3}$**:** A subset with items of the form $[h, N^\beta \rightarrow \delta \bullet \nu, i, j \mid p, q \mid adj]$ such that $N^\beta \rightarrow \delta\nu \in \mathcal{P}(\beta)$, $\beta \in \boldsymbol{A'}$, $0 \le h \le i \le j$, $adj \in \{\text{true}, \text{false}\}$, $(p, q) \le (i, j)$ and the foot node of β is a descendant of a node in δ. The substring spanned by δ is $a_{i+1} \ldots a_p\ \mathbf{F}^\beta\ a_{q+1} \ldots a_j$. Thus, p and q are positions in the input string indicating a discontinuity in the string recognized by β, due to the substring $a_{p+1} \ldots a_q$ should be spanned by the node at which the auxiliary tree β has been adjoined. With respect to the boolean component, $adj = \text{true}$ if some auxiliary tree has been previously adjoined at N^γ, otherwise $adj = \text{false}$.

$\mathcal{I}^{(3)}_{\text{Mix}_3}$**:** A subset with items of the form $[[M^\gamma \rightarrow \upsilon \bullet, i, j \mid p, q \mid \text{true}]]$ such that $M^\gamma \rightarrow \upsilon \in \mathcal{P}(\beta)$, $\beta \in \boldsymbol{A'}$, $0 \le i \le j$ and $(p, q) = (-, -)$ or $(p, q) \le (i, j)$. These items are generated as a kind of intermediate items during the completion of the adjunctions of auxiliary trees in $\boldsymbol{A'}$.

4.2. ***Deduction steps***

The set of deduction steps is formed by the following subsets:

$$\begin{aligned}\mathcal{D}_{\text{Mix}_3} = & \mathcal{D}^{\text{Init}}_{\text{Mix}_3} \cup \mathcal{D}^{\text{Scan}}_{\text{Mix}_3} \cup \mathcal{D}^{\varepsilon}_{\text{Mix}_3} \cup \mathcal{D}^{\text{Pred}^1}_{\text{Mix}_3} \cup \mathcal{D}^{\text{Pred}^3}_{\text{Mix}_3} \cup \mathcal{D}^{\text{Comp}}_{\text{Mix}_3} \cup \\ & \mathcal{D}^{\text{LAdjPred}}_{\text{Mix}_3} \cup \mathcal{D}^{\text{LAdjComp}}_{\text{Mix}_3} \cup \mathcal{D}^{\text{RAdjPred}}_{\text{Mix}_3} \cup \mathcal{D}^{\text{RAdjComp}}_{\text{Mix}_3} \cup \mathcal{D}^{\text{LRFoot}}_{\text{Mix}_3} \cup \\ & \mathcal{D}^{\text{AdjPred}}_{\text{Mix}_3} \cup \mathcal{D}^{\text{FootPred}}_{\text{Mix}_3} \cup \mathcal{D}^{\text{FootComp}}_{\text{Mix}_3} \cup \mathcal{D}^{\text{AdjComp}^1}_{\text{Mix}_3} \cup \mathcal{D}^{\text{AdjComp}^2}_{\text{Mix}_3}\end{aligned}$$

The parsing process starts by creating the items corresponding to productions having the root of an initial tree as right-hand side and the dot in the leftmost position of the right-hand side:

$$\mathcal{D}^{\text{Init}}_{\text{Mix}_3} = \frac{}{[-, \top \rightarrow \bullet\mathbf{R}^\alpha, 0, 0 \mid -, - \mid \text{false}]} \quad \alpha \in \boldsymbol{I}\ \wedge\ S = \text{label}(\mathbf{R}^\alpha)$$

In order to preserve the correct prefix property, we must be very careful when predicting the left-most child of a given node N^γ. Thus, to generate the consequent item in a deduction step corresponding to the subset $\mathcal{D}^{\mathrm{Pred}^1}_{\mathrm{Mix}_3}$

$$\mathcal{D}^{\mathrm{Pred}^1}_{\mathrm{Mix}_3} = \frac{[h, N^\gamma \to \bullet M^\gamma \nu, i, j \mid -, - \mid adj]}{[h, M^\gamma \to \bullet \upsilon, j, j \mid -, - \mid \text{false}]}$$

one of the following conditions must be satisfied:

1) Adjunction is forbidden at node N^γ and $adj = \text{false}$.

2) Adjunction is optional at node N^γ but any strongly left auxiliary tree can be adjoined at this node. As a consequence, the value of adj should be false.

3) Adjunction is optional at node N^γ and some strongly left auxiliary tree can be adjoined at this node or there exists some auxiliary trees belonging to $\boldsymbol{A'} \cup \boldsymbol{A}_{SR}$ that can be adjoined at N^γ. No restriction is applied on the value of adj.

4) Adjunction is mandatory at node N^γ but only strongly left auxiliary trees can be adjoined at this node. The value of adj should be $\mathtt{true}$ to guarantee that at least one adjunction has been performed at N^γ.

5) Adjunction is mandatory at node N^γ but any strongly left auxiliary tree can be adjoined at this node. As a consequence, the value of adj should be false.

6) Adjunction is mandatory at node N^γ but there exists some auxiliary trees belonging to $\boldsymbol{A'} \cup \boldsymbol{A}_{SR}$ that can be adjoined at N^γ. The value of adj is not restricted at this moment.

The rest of children of a given node N^γ are predicted as in the $\mathbb{P}_{\mathrm{Mix}_2}$ parsing system:

$$\mathcal{D}^{\mathrm{Pred}^2}_{\mathrm{Mix}_3} = \frac{[h, N^\gamma \to \delta \bullet M^\gamma \nu, i, j \mid p, q \mid adj]}{[h, M^\gamma \to \bullet \upsilon, j, j \mid -, - \mid \text{false}]} \quad \delta \neq \varepsilon$$

Once the children of M^γ have been completely traversed, a step in $\mathcal{D}^{\mathrm{Comp}}_{\mathrm{Mix}_3}$

$$\mathcal{D}^{\mathrm{Comp}}_{\mathrm{Mix}_3} = \frac{\begin{array}{c}[h, N^\gamma \to \delta \bullet M^\gamma \nu, i, j \mid p, q \mid adj],\\ [h, M^\gamma \to \upsilon \bullet, j, k \mid p', q' \mid adj']\end{array}}{[h, N^\gamma \to \delta M^\gamma \bullet \nu, i, k \mid p \cup p', q \cup q' \mid adj]}$$

should be applied, checking that one of the following conditions is satisfied:

1) Adjunction is mandatory at M^γ and $adj' = \mathtt{true}$.

2) Adjunction is forbidden at M^γ and $adj' = \text{false}$.

3) Adjunction is optional at M^γ and therefore there are no restrictions on the value of adj.

Input symbols and the empty string are recognized by deduction steps in $\mathcal{D}^{\mathrm{Scan}}_{\mathrm{Mix}_3}$ and $\mathcal{D}^{\varepsilon}_{\mathrm{Mix}_3}$, respectively:

$$\mathcal{D}^{\text{Scan}}_{\text{Mix}_3} = \frac{\begin{array}{c}[h, N^\gamma \to \delta \bullet M^\gamma \nu, i, j \mid p, q \mid adj],\\ [a, j, j+1]\end{array}}{[h, N^\gamma \to \delta M^\gamma \bullet \nu, i, j+1 \mid p, q \mid adj]} \;\; a = \text{label}(M^\gamma)$$

$$\mathcal{D}^{\varepsilon}_{\text{Mix}_3} = \frac{[h, N^\gamma \to \delta \bullet M^\gamma \nu, i, j \mid p, q \mid adj]}{[h, N^\gamma \to \delta M^\gamma \bullet \nu, i, j \mid p, q \mid adj]} \;\; \varepsilon = \text{label}(M^\gamma)$$

If a strongly left auxiliary tree $\beta \in \boldsymbol{A}_{SL}$ can be adjoined at a given node M^γ, a step in $\mathcal{D}^{\text{LAdjPred}}_{\text{Mix}_1}$ starts the traversal of β. When β has been completely traversed, a step in $\mathcal{D}^{\text{LAdjComp}}_{\text{Mix}_1}$ starts the traversal of the subtree corresponding to M^γ and sets the last element of the item to true in order to indicate that an adjunction has taken place on this node. As in $\mathbb{P}_{\text{Mix}_2}$, simultaneous adjunctions of several strongly left auxiliary trees on a node M^γ is achieved by applying a pair of steps $\mathcal{D}^{\text{LAdjPred}}_{\text{Mix}_3}$ and $\mathcal{D}^{\text{LAdjComp}}_{\text{Mix}_3}$ for each auxiliary tree.

$$\mathcal{D}^{\text{LAdjPred}}_{\text{Mix}_3} = \frac{[h, M^\gamma \to \bullet\upsilon, i, j \mid -, - \mid adj]}{[-, \top \to \bullet\mathbf{R}^\beta, j, j \mid -, - \mid \text{false}]} \;\; \beta \in \text{adj}(M^\gamma) \;\wedge\; \beta \in \boldsymbol{A}_{SL}$$

$$\mathcal{D}^{\text{LAdjComp}}_{\text{Mix}_3} = \frac{\begin{array}{c}[h, M^\gamma \to \bullet\upsilon, i, j \mid -, - \mid adj],\\ [-, \top \to \mathbf{R}^\beta\bullet, j, k \mid -, - \mid \text{false}]\end{array}}{[h, M^\gamma \to \bullet\upsilon, i, k \mid -, - \mid \text{true}]} \;\; \beta \in \boldsymbol{A}_{SL} \;\wedge\; \beta \in \text{adj}(M^\gamma)$$

If a strongly right auxiliary tree $\beta \in \boldsymbol{A}_{SR}$ can be adjoined at a given node M^γ, when the subtree corresponding to this node has been completely traversed, a step in $\mathcal{D}^{\text{RAdjPred}}_{\text{Mix}_1}$ starts the traversal of the tree β. When β has been completely traversed, a step in $\mathcal{D}^{\text{RAdjComp}}_{\text{Mix}_1}$ updates the input positions spanned by M^γ taking into account the part of the input string spanned by β, and sets the last element of the item to true in order to indicate that an adjunction has taken place on this node. As in $\mathbb{P}_{\text{Mix}_2}$, simultaneous adjunctions of several strongly right auxiliary trees on a node M^γ is achieved by applying a pair of steps $\mathcal{D}^{\text{RAdjPred}}_{\text{Mix}_3}$ and $\mathcal{D}^{\text{RAdjComp}}_{\text{Mix}_3}$ for each auxiliary tree.

$$\mathcal{D}^{\text{RAdjPred}}_{\text{Mix}_3} = \frac{[h, M^\gamma \to \upsilon\bullet, i, j \mid p, q \mid adj]}{[-, \top \to \bullet\mathbf{R}^\beta, j, j \mid -, - \mid \text{false}]} \;\; \beta \in \boldsymbol{A}_{SR} \;\wedge\; \beta \in \text{adj}(M^\gamma)$$

$$\mathcal{D}^{\text{RAdjComp}}_{\text{Mix}_3} = \frac{\begin{array}{c}[h, M^\gamma \to \upsilon\bullet, i, j \mid p, q \mid adj],\\ [-, \top \to \mathbf{R}^\beta\bullet, j, k \mid -, - \mid \text{false}]\end{array}}{[h, M^\gamma \to \upsilon\bullet, i, k \mid p, q \mid \text{true}]} \;\; \beta \in \boldsymbol{A}_{SR} \;\wedge\; \beta \in \text{adj}(M^\gamma)$$

The foot nodes of strongly left and right auxiliary trees are skipped by a step in the set $\mathcal{D}^{\text{LRFoot}}_{\text{Mix}_3}$:

$$\mathcal{D}^{\text{LRFoot}}_{\text{Mix}_3} = \frac{[-, \mathbf{F}^\beta \to \bullet\bot, j, j \mid -, - \mid adj]}{[-, \mathbf{F}^\beta \to \bot\bullet, j, j \mid -, - \mid adj]} \;\; \beta \in \boldsymbol{A}_{SL} \cup \boldsymbol{A}_{SR}$$

A step in $\mathcal{D}^{\text{AdjPred}}_{\text{Mix}_3}$ predicts the adjunction of an auxiliary tree $\beta \in \boldsymbol{A'}$ in a node of an elementary tree γ, storing the position j at which the traversal of β was started.

$$\mathcal{D}^{\text{AdjPred}}_{\text{Mix}_3} = \frac{[h, M^\gamma \to \bullet v, i, j \mid -, - \mid adj]}{[j, \top \to \bullet\mathbf{R}^\beta, j, j \mid -, - \mid \text{false}]} \;\; \beta \in \boldsymbol{A'} \;\wedge\; \beta \in \text{adj}(M^\gamma)$$

Once the foot of β has been reached, the traversal of β is momentary suspended by a step in $\mathcal{D}^{\text{FootPred}}_{\text{Mix}_3}$, which re-takes the subtree of γ which must be attached to the foot of β, checking the position at which the traversal of γ was suspended is compatible with the position at which the traversal of β was started.

$$\mathcal{D}^{\text{FootPred}}_{\text{Mix}_3} = \frac{\begin{array}{c}[h, M^\gamma \to \bullet v, i, j \mid -, - \mid adj]\\ [j, \mathbf{F}^\beta \to \bullet\bot, k, k \mid -, - \mid adj]\end{array}}{[h, M^\gamma \to \bullet v, k, k \mid -, - \mid adj]} \;\; \beta \in \boldsymbol{A'} \;\wedge\; \beta \in \text{adj}(M^\gamma)$$

When the traversal of M^γ has been completed, a step in $\mathcal{D}^{\text{FootComp}}_{\text{Mix}_3}$ re-takes the traversal of β continuing at the foot node, checking again that the position at which the traversal of γ was suspended is compatible with the position at which the traversal of β was started. These checkings are needed to guarantee the correct prefix property is preserved at any moment.

$$\mathcal{D}^{\text{FootComp}}_{\text{Mix}_3} = \frac{\begin{array}{c}[h, M^\gamma \to \bullet v, i, j \mid -, - \mid adj],\\ [j, \mathbf{F}^\beta \to \bullet\bot, l, l \mid -, - \mid adj],\\ [h, M^\gamma \to v\bullet, l, m \mid p', q' \mid adj']\end{array}}{[j, \mathbf{F}^\beta \to \bot\bullet, l, m \mid l, m \mid adj]} \;\; \beta \in \boldsymbol{A'} \;\wedge\; \beta \in \text{adj}(M^\gamma)$$

When the traversal of β is completely finished, a deduction step in $\mathcal{D}^{\text{AdjComp}^1}_{\text{Mix}_3}$ checks if the subtree attached to the foot of β corresponds with the adjunction node. The adjunction if finished by a step in $\mathcal{D}^{\text{AdjComp}^2}_{\text{Mix}_3}$, taking into account that p' and q' are instantiated if and only if the adjunction node is on the spine of γ.

$$\mathcal{D}^{\text{AdjComp}^1}_{\text{Mix}_3} = \frac{\begin{array}{c}[j, \top \to \mathbf{R}^\beta\bullet, j, r \mid l, m \mid \text{false}],\\ [h, M^\gamma \to v\bullet, l, m \mid p', q' \mid adj]\end{array}}{[[M^\gamma \to v\bullet, j, r \mid p', q' \mid \text{true}]]} \;\; \beta \in \boldsymbol{A'} \;\wedge\; \beta \in \text{adj}(M^\gamma)$$

$$\mathcal{D}^{\text{AdjComp}^2}_{\text{Mix}_3} = \frac{\begin{array}{c}[h, M^\gamma \to \bullet v, i, j \mid -, - \mid adj],\\ [[M^\gamma \to v\bullet, j, r \mid p', q' \mid \text{true}]],\\ [h, M^\gamma \to v\bullet, l, m \mid p', q' \mid adj]\end{array}}{[h, M^\gamma \to v\bullet, i, r \mid p', q' \mid \text{true}]} \;\; \beta \in \boldsymbol{A'} \;\wedge\; \beta \in \text{adj}(M^\gamma)$$

Simultaneous adjunctions of several auxiliary trees in $\beta \in \boldsymbol{A'}$ is achieved by using the consequent item generated by a deduction step in $\mathcal{D}^{\text{FootPred}}_{\text{Mix}_3}$ as antecedent of a deduction step in $\mathcal{D}^{\text{AdjPred}}_{\text{Mix}_3}$ to start the adjunction of an auxiliary tree $\beta' \in \boldsymbol{A'}$. When the traversal of β' has finished, a step in $\mathcal{D}^{\text{FootComp}}_{\text{Mix}_2}$ re-takes the traversal of β at the foot node. The process is repeated for each auxiliary tree which is to be simultaneously adjoined.

A major difference of this parsing system with respect to $\mathbb{P}_{\text{Mix}_2}$ is that *Comb* steps are not needed, due to the strong prediction performed by steps in $\mathcal{D}^{\text{Pred}^1}_{\text{Mix}_3}$ guarantees that simultaneous adjunctions are applied from left to right with respect to the input string.

The input string belongs to the language defined by the grammar if a final item in the set $\mathcal{F} = \left\{ \; [-, \top \rightarrow \mathbf{R}^\alpha \bullet, 0, n \mid -, - \mid \text{false}] \mid \alpha \in \boldsymbol{I} \;\wedge\; S = \text{label}(\mathbf{R}^\alpha) \; \right\}$ is generated.

5. Complexity

The worst-case space complexity of the algorithms described by $\mathbb{P}_{\text{Mix}_1}$ and $\mathbb{P}_{\text{Mix}_2}$ is in $\mathcal{O}(n^4)$, as at most four input positions are stored into items corresponding to auxiliary trees belonging to $\boldsymbol{A'}$. For $\mathbb{P}_{\text{Mix}_3}$, the worst-case space complexity is in $\mathcal{O}(n^5)$. In all cases, initial trees and strongly left and right auxiliary trees contribute $\mathcal{O}(n^2)$ to the final result.

With respect to the worst-case time complexity:

– TIG adjunction, the adjunction of a strongly left or right auxiliary tree on a node of a tree belonging to $\boldsymbol{I} \cup \boldsymbol{A}_{SL} \cup \boldsymbol{A}_{SR}$, is managed in $\mathcal{O}(n^3)$ by *LAdjComp* and *RAdjComp* steps in all algorithms.

– In $\mathbb{P}_{\text{Mix}_1}$ and $\mathbb{P}_{\text{Mix}_2}$, full TAG adjunction is managed in $\mathcal{O}(n^6)$ by *AdjComp* deduction steps, which are in charge of dealing with auxiliary trees belonging to $\boldsymbol{A'}$. In fact, $\mathcal{O}(n^6)$ is only attained when a wrapping auxiliary tree is adjoined on a spine node of a wrapping auxiliary tree. The adjunction of a wrapping auxiliary tree on a right node of a wrapping auxiliary tree is managed in $\mathcal{O}(n^5)$ due to *Comp* deduction steps.

– In $\mathbb{P}_{\text{Mix}_3}$, full TAG adjunction is managed in $\mathcal{O}(n^6)$ by *AdjComp*^{1} steps and in $\mathcal{O}(n^5)$ by *AdjComp*^{2} steps when a wrapping auxiliary tree is adjoined on a spine node of a wrapping auxiliary tree, thus given an overall complexity of $\mathcal{O}(n^6)$. The adjunction of a wrapping auxiliary tree on a right node of a wrapping auxiliary tree is managed in $\mathcal{O}(n^4)$ by *AdjComp*^{1} steps and in $\mathcal{O}(n^3)$ by *AdjComp*^{2} steps, but in $\mathcal{O}(n^5)$ by *Comp* deduction steps.

– The adjunction of a strongly right auxiliary tree on a spine or right node of a wrapping auxiliary tree is managed in $\mathcal{O}(n^5)$ time due to *RAdjComp* deduction steps.

– Other cases of adjunction, e.g., the adjunction of a strongly left or right auxiliary tree on a spine node of a tree belonging to $(\boldsymbol{A}_L - \boldsymbol{A}_{SL}) \cup (\boldsymbol{A}_R - \boldsymbol{A}_{SR})$, are managed in $\mathcal{O}(n^4)$.

Transitive and Ditransitive	
(1)	Srini bought a book
(2)	Srini bought Beth a book
Arguments and Adjuncts	
(3)	Srini bought a book at the bookstore
(4)	he put the book on the table
(5)	*he put the book
Ergative and Intransitive	
(6)	the sun melted the ice
(7)	the ice melted
(8)	Elmo borrowed a book
(9)	*a book borrowed
Sentential Complements	
(10)	he hopes Muriel wins
(11)	he hopes that Muriel wins
Relative Clauses	
(12)	the man who Muriel likes bought a book
(13)	the man that Muriel likes bought a book
Auxiliary Verbs	
(14)	the music should have been being played for the president
Extraction	
(15)	Clove caught a frisbee
(16)	who caught a frisbee
(17)	what did Clove catch
Unbounded Dependencies	
(18)	the aardvark smells terrible
(19)	the emu thinks that the aardvark smells terrible
(20)	who does the emu think smells terrible
(21)	who did the elephant think the panda heard the emu said smells terrible
Adjectives	
(22)	Herbert is angry
(23)	Herbert is angry and furious
(24)	Herbert is more livid than angry
(25)	Herbert is more livid and furious than angry

Table 1. *Sentences used in the XTAG experiment*

6. Experimental results

We have incorporated the parsing algorithms described in this article into a naive implementation in Prolog of the deductive parsing machine presented in [SHI 95]. As a first experiment, we have compared the performance of the Earley-like parsing algorithms for TIG [SCH 95] and TAG [ALO 99] with respect to TIGs. For this purpose, we have made the experiments on two simple TIGs $G_L = \{\alpha, \beta_L\}$ and $G_R = \{\alpha, \beta_R\}$ (see Figure 3). For a TIG, the time complexity of the adjunction completion step of a TAG parser is $\mathcal{O}(n^4)$, in contrast with the $\mathcal{O}(n^3)$ complexity of left and right adjunction completion for a TIG parser. Therefore, we expected the TIG parser to be considerably faster than the TAG parser. In effect, for G_L we have observed that the TIG parser is up to 18 times faster than the TAG parser, but in the case of G_R the difference becomes irrelevant.

These results have been corroborated by a second experiment performed on artificial TAGs with the mixed ($\mathbb{P}_{\text{Mix}}$) and the TAG parser: the performance of the mixed parser improves when strongly left auxiliary trees are involved in the analysis of the input string.

In a third experiment, we have taken a subset of the XTAG grammar [DOR 94], consisting of 27 elementary trees that cover a variety of English constructions: relative clauses, auxiliary verbs, unbounded dependencies, extraction, etc. In order to eliminate the time spent by unification, we have not considered the feature structures of elementary trees. Instead, we have simulated the features using local constraints. The set of sentences used in the experiment is shown in table 1. Every sentence has been parsed without previous filtering of elementary trees.

First of all, we have implemented a combined parser $\mathbb{P}_{\text{Mix}_1}$ where simultaneous adjunctions are forbidden and we have corroborated the results included in [ALO 02]: the application of the parser $\mathbb{P}_{\text{Mix}_1}$ results in a reduction in time, with respect to classical Earley-like parsers for TAG, that varies in percentage from 31% to 0%, depending on the kind of trees involved in the analysis of each sentence. Then, we have compared the parsers $\mathbb{P}_{\text{Mix}_1}$, $\mathbb{P}_{\text{Mix}_2}$ and $\mathbb{P}_{\text{Mix}_3}$ to test the benefits of simultaneous adjunctions and preserving the correct prefix property. Table 2 shows the results of this experiment:

– The first column is the number of the corresponding sentence in table 1.

– The second and third column show the time, in seconds, spent by parsers $\mathbb{P}_{\text{Mix}_1}$ and $\mathbb{P}_{\text{Mix}_2}$ in the analysis of each sentence, respectively.

– The fourth column, labeled Δ_{12}, shows the difference, in percentage, of the time spent by $\mathbb{P}_{\text{Mix}_2}$ with respect to $\mathbb{P}_{\text{Mix}_1}$. Negative values indicate real improvements. As we can observe, $\mathbb{P}_{\text{Mix}_2}$ obtains a reduction in time that varies in percentage from 46% to 12%, depending on the kind of trees involved in the analysis of each sentence. We would like to address the results obtained by our approach in sentences 12, 13 and 14 where simultaneous adjunctions of left and right auxiliary trees must be applied. In these cases, the parser $\mathbb{P}_{\text{Mix}_1}$ needs to apply a classical wrapping adjunction.

– The fifth column shows the time, in seconds, spent by the parser $\mathbb{P}_{\text{Mix}_3}$ for each sentence.

– The sixth column, labeled Δ_{23}, shows the difference, in percentage, of the time spent by $\mathbb{P}_{\text{Mix}_3}$ with respect to $\mathbb{P}_{\text{Mix}_2}$. It is interesting to remark that preserving the correct prefix property increases the computational cost of the parsing process from 11% to 50%. These results suggest that, although the time complexity is in $\mathcal{O}(n^6)$ for both parsers, some constants involved in the expression of complexity for $\mathbb{P}_{\text{Mix}_3}$ must be greater than the corresponding ones for $\mathbb{P}_{\text{Mix}_2}$. A detailed examination of the trace of both executions shows that:

1) In the traversal of initial and strongly left and right auxiliary trees, the number of deduction steps applied by both parsers is the same, i.e., all the gain in performance due to considering a part of the grammar as a TIG is attained by $\mathbb{P}_{\text{Mix}_2}$.

2) In the traversal of wrapping auxiliary trees, the number of deduction steps applied by $\mathbb{P}_{\text{Mix}_3}$ is slightly lower than $\mathbb{P}_{\text{Mix}_2}$.

3) Independently of the kind of trees involved in the analysis of a sentence, the number of inferences (i.e., the number of CALL and REDO performed by the Prolog interpreter) is higher in $\mathbb{P}_{\text{Mix}_3}$ than in $\mathbb{P}_{\text{Mix}_2}$, due to the complex checkings performed by *Pred*^{1} steps.

– Finally, the seventh column, labeled Δ_{13}, shows the difference, in percentage, of the time spent by $\mathbb{P}_{\text{Mix}_3}$ with respect to $\mathbb{P}_{\text{Mix}_1}$. We can observe that $\mathbb{P}_{\text{Mix}_3}$ obtain better results for 76% of the sentences.

7. Conclusion

We have defined several parsing algorithms which reduce the practical complexity of TAG parsing by taking into account that a large part of actual TAG grammars can be managed as a TIG. Several approaches has been tried: the first parser forbid simultaneous adjunctions, the second one extends the classical adjunction operation in TAG by considering the possibility of simultaneous adjunctions at a given node, and the third one allows simultaneous adjunctions at the time it preserves the correct prefix property.

Practical experiments performed on a subset of the XTAG grammars show that considering simultaneous adjunctions improves highly the parsing efficiency due to a larger number of adjunctions can be managed as TIG adjunctions. In contrast, preserving the correct prefix property in mixed parsers have shown to be of little interest due to the high cost involved by the stronger predictions that must be performed to satisfy such property.

The performance of the algorithms could be improved by means of the application of practical optimizations, such as the replacement of the components p and q of items $[N^\gamma \rightarrow \delta \bullet \nu, i, j \mid p, q] \in \mathcal{I}^{(a)}_{\text{Mix}}$ by the list of all adjunctions that are still under com-

Sentence	Time $\mathbb{P}_{\mathrm{Mix}_1}$	Time $\mathbb{P}_{\mathrm{Mix}_2}$	Δ_{12}	Time $\mathbb{P}_{\mathrm{Mix}_3}$	Δ_{23}	Δ_{13}
(1)	0.13	0.08	-38.46%	0.12	+50.00%	+7.69%
(2)	0.17	0.11	-35.29%	0.15	+35.36%	-11.76%
(3)	0.21	0.15	-28.57%	0.20	+33.33%	-4.76%
(4)	0.18	0.13	-27.78%	0.18	+38.46%	-0.00%
(5)	0.10	0.07	-30.00%	0.10	+42.85%	-0.00%
(6)	0.17	0.11	-35.29%	0.16	+45.45%	-5.88%
(7)	0.10	0.07	-30.00%	0.09	+28.57%	-10.00%
(8)	0.13	0.08	-38.46%	0.11	+37.05%	-15.38%
(9)	0.08	0.06	-25.00%	0.08	+33.33%	-0.00%
(10)	0.21	0.14	-33.33%	0.19	+35.71%	-9.52%
(11)	0,27	0.20	-25.93%	0.27	+35.00%	-0.00%
(12)	0,32	0,24	-25.00%	0.36	+50.00%	+12.50%
(13)	0.28	0.21	-25.00%	0.30	+42.85%	+7.14%
(14)	0.33	0.29	-12.12%	0.41	+37.93%	+24.24%
(15)	0.12	0.09	-25.00%	0.11	+22.22%	-8.33%
(16)	0.12	0.09	-25.00%	0.11	+22.22%	-8.33%
(17)	0.13	0.07	-46.15%	0.10	+42.85%	-23.08%
(18)	0.10	0.07	-30.00%	0.09	+28.57%	-10.00%
(19)	0.32	0.27	-15.63%	0.38	+40.74%	+18.75%
(20)	0.21	0.12	-42.86%	0.19	+58.33%	-9.52%
(21)	0.58	0.39	-32.76%	0.59	+51.28%	+1.72%
(22)	0.09	0.07	-22.22%	0.08	+14.28%	-11.11%
(23)	0.14	0.09	-35.71%	0.10	+11.11%	-28.57%
(24)	0.12	0.08	-33.33%	0.11	+25.00%	-8.33%
(25)	0.13	0.10	-23.08%	0.12	+20.00%	-7.69%

Table 2. *XTAG results, in seconds, for $\mathbb{P}_{\mathrm{Mix}_1}$ and $\mathbb{P}_{\mathrm{Mix}_2}$ and $\mathbb{P}_{\mathrm{Mix}_3}$ parsers*

pletion on N^γ [CLE 01], albeit this modification increase the worst-case complexity of the algorithm.

Acknowledgements: We would like to thank Margarita Alonso for her comments and suggestions. This research has been supported in part by Ministerio de Ciencia y Tecnología (grants TIC2000-0370-C02-01, FIT-150500-2002-416, HP2001-0044 and HF2002-81), Xunta de Galicia (grants PGIDT01PXI10506PN, PGIDIT02SIN01E and PGIDIT03SIN30501PR) and Universidade da Coruña.

8. References

[ABE 00] ABEILLÉ A., RAMBOW O., "Tree Adjoining Grammar: an Overview", ABEILLÉ A., RAMBOW O., Eds., *Tree Adjoining Grammars. Formalisms, Linguistic Analysis and*

Procesing, vol. 107 of *CSLI Lecture Notes*, chapter 1, p. 1–68, CSLI Publications, Stanford, California, 2000.

[ALO 99] ALONSO M., CABRERO D., DE LA CLERGERIE E., VILARES M., "Tabular Algorithms for TAG Parsing", *Proc. of EACL'99, Ninth Conference of the European Chapter of the Association for Computational Linguistics*, Bergen, Norway, June 1999, p. 150–157.

[ALO 02] ALONSO M., CARRILLO V., DÍAZ V., "Mixed Parsing of Tree Insertion and Tree Adjoining Grammars", GARIJO F. J., RIQUELME J. C., TORO M., Eds., *Advances in Artificial Intelligence - IBERAMIA 2002*, vol. 2527 of *Lecture Notes in Artificial Intelligence*, p. 694–703, Springer-Verlag, Berlin-Heidelberg-New York, 2002.

[ALO 03] ALONSO M., DÍAZ V., "Parsing Tree Adjoining Grammars and Tree Insertion Grammars with simultaneous adjunctions", *Proc. of 8th International Workshop on Parsing Technologies (IWPT 2003)*, p. 19–30, Nancy, France, April 2003.

[CLE 01] DE LA CLERGERIE E., "Refining Tabular Parsers for TAGs", *Proceedings of Language Technologies 2001: The Second Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL'01)*, CMU, Pittsburgh, PA, USA, June 2001, p. 167–174.

[DOR 94] DORAN C., EGEDI D., HOCKEY B. A., SRINIVAS B., ZAIDEL M., "XTAG System — A Wide Coverage Grammar for English", *Proc. of the 15th International Conference on Computational Linguistics (COLING'94)*, Kyoto, Japan, August 1994, p. 922–928.

[JOS 75] JOSHI A. K., LEVY L. S., TAKAHASHI M., "Tree Adjunt Grammars", *Journal of Computer and System Sciences*, vol. 10, num. 1, 1975, p. 136–162.

[JOS 97] JOSHI A. K., SCHABES Y., "Tree-Adjoining Grammars", ROZENBERG G., SALOMAA A., Eds., *Handbook of Formal Languages. Vol 3: Beyond Words*, chapter 2, p. 69–123, Springer-Verlag, Berlin/Heidelberg/New York, 1997.

[NED 99] NEDERHOF M.-J., "The Computational Complexity of the Correct-Prefix Property for TAGs", *Computational Linguistics*, vol. 25, num. 3, 1999, p. 345–360.

[SCH 91] SCHABES Y., "The Valid Prefix Property and Left to Right Parsing of Tree-Adjoining Grammar", *Proc. of II International Workshop on Parsing Technologies, IWPT'91*, Cancún, Mexico, 1991, p. 21–30.

[SCH 95] SCHABES Y., WATERS R. C., "Tree Insertion Grammar: A Cubic-Time Parsable Formalism That Lexicalizes Context-Free Grammar Without Changing the Trees Produced", *Computational Linguistics*, vol. 21, num. 4, 1995, p. 479–513, Also as Technical Report TR-94-13, June 1994, Mitsubishi Electric Research Laboratories, Cambridge, MA, USA.

[SHI 95] SHIEBER S. M., SCHABES Y., PEREIRA F., "Principles and Implementation of Deductive Parsing", *Journal of Logic Programming*, vol. 24, num. 1–2, 1995, p. 3–36.

[SIK 97] SIKKEL K., *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, 1997.