PARSING TREE ADJOINING GRAMMARS AND TREE INSERTION GRAMMARS WITH SIMULTANEOUS ADJUNCTIONS

Miguel A. Alonso
Departamento de Computación

Universidade da Coruña
Campus de Elviña s/n
15071 La Coruña (Spain)
alonso@udc.es

Víctor J. Díaz
Departamento de Lenguajes
y Sistemas Informáticos
Universidad de Sevilla
Avda. Reina Mercedes s/n
41012 Sevilla (Spain)
vjdiaz@lsi.us.es

Abstract

A large part of wide coverage Tree Adjoining Grammars (TAG) is formed by trees that satisfy the restrictions imposed by Tree Insertion Grammars (TIG). 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. In this paper, we describe a parsing algorithm managing simultaneous adjunctions in TAG and TIG.

1 Introduction

Tree Adjoining Grammar (TAG) [5] and Tree Insertion Grammar (TIG) [7] are grammatical formalisms that make use of a tree-based operation called adjunction. 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. In contrast, TIG generates context-free languages and can be parsed in $\mathcal{O}(n^3)$ for time and in $\mathcal{O}(n^2)$ for space, due to restrictions on the form of trees.

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 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 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. 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 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

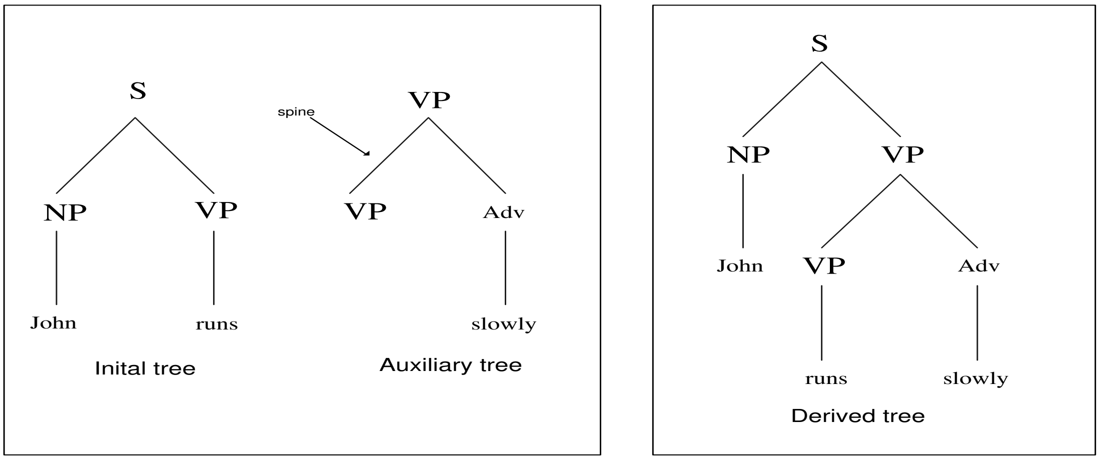

Figure 1: Adjunction operation

adjoining the VP auxiliary tree on the node labeled by VP located in the initial tree is also shown.

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 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. Given an auxiliary tree, we call *spine nodes* to those nodes placed on the spine and *left nodes* (resp. *right nodes*) to those nodes placed to the left (resp. right) of the spine. 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. Figure 2 shows three derived trees resulting from the adjunction of a wrapping, left and right auxiliary tree, respectively. We denote by $\boldsymbol{A'}$ the set $\boldsymbol{A} - (\boldsymbol{A}_{SL} \cup \boldsymbol{A}_{SR})$.

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.

It has been found that most of the trees and adjunction operations involved in wide coverage grammars like XTAG [4] are compatible with the TIG formalism [7]. 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. Therefore, we propose to create mixed parsers taking the best of both worlds: those parts of the grammar that correspond to a TIG should be 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 should be managed in $\mathcal{O}(n^6)$ time and $\mathcal{O}(n^4)$ space complexity.

A first approach towards this aim has been shown in [2], where a Earley-like TAG parser has been merged with an Earley-like TIG parser. Some questionable decisions were taken in order to make both parsers compatible, the most important one being the disabling of simultaneous adjunctions. The rationale behind this decision was to follow the standard TAG definition in case of mismatching between TAG and TIG definitions. Albeit an important speed-up was

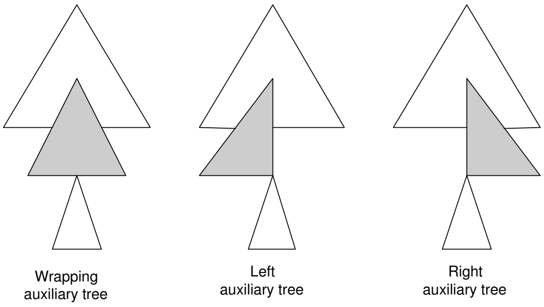

Figure 2: TAG vs. TIG adjunction operation

obtained by the resulting algorithm, its usefulness is limited by the fact that a lot of trees with a TIG-skeleton do not satisfy the definition of strongly left or strongly right auxiliary trees because we can not combine both types of trees on a single node. For example, determiners or adjectives are usually modelled with left auxiliary trees but relative clauses are modelled with right auxiliary trees. Then, we need to combine left and right auxiliary trees when a noun is modified at the same time with determiners and relative clauses. The only way to do that is using adjunction operations. If these adjunctions are not performed simultaneously at the same node, auxiliary trees for determiners and relative clauses cannot be considered as strongly left and strongly right auxiliary trees, respectively, and therefore the algorithm in [2] behaves like a classical TAG parser with respect to these adjunctions.

1.1 Notation for parsing algorithms

We will describe parsing algorithms using *Parsing Schemata*, a framework for high-level descriptions of parsing algorithms [9]. 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.

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 $N^\gamma \to N_1^\gamma \dots N_g^\gamma$. 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 $\top \to \mathbf{R}^\alpha$ for each $\alpha \in \boldsymbol{I}$ and the two additional productions $\top \to \mathbf{R}^\beta$ and $\mathbf{F}^\beta \to \bot$ 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

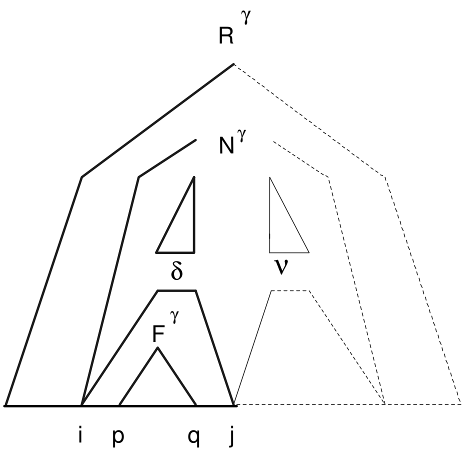

Figure 3: Graphical representation of items

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 $\beta \in \text{adj}(N^\gamma)$ 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 $\textbf{nil} \in \text{adj}(N^\gamma)$ where $\textbf{nil} \notin \boldsymbol{I} \cup \boldsymbol{A}$ is a dummy symbol. If adjunction is not allowed at N^γ then $\{\textbf{nil}\} = \text{adj}(N^\gamma)$.

2 The parsing algorithm

In this section we define a parsing system $\mathrm{P}_{\text{Mix}} = \langle \mathcal{I}_{\text{Mix}}, \mathcal{H}_{\text{Mix}}, \mathcal{D}_{\text{Mix}} \rangle$ corresponding to a mixed parsing algorithm for TAG and TIG in which the adjunction of strongly left and strongly right auxiliary trees[1] will be managed by specialized deduction steps. 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 [7].

For P_{Mix}, we consider a set of items $\mathcal{I}_{\text{Mix}} = \mathcal{I}_{\text{Mix}}^{(a)} \cup \mathcal{I}_{\text{Mix}}^{(b)}$ formed by the union of the following subsets:

- A subset $\mathcal{I}_{\text{Mix}}^{(a)}$ with items of the form $[N^\gamma \rightarrow \delta \bullet \nu, i, j \mid p, q \mid adj]$ such that $N^\gamma \rightarrow \delta\nu \in \mathcal{P}(\gamma)$, $\gamma \in \boldsymbol{I} \cup \boldsymbol{A}$, $0 \leq i \leq j$, $(p, q) = (-, -)$ or $(p, q) \leq (i, j)$, and $adj \in \{\text{true}, \text{false}\}$. 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 $adj = \text{false}$. 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 3). 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 $p = q = -$ representing they are undefined. Therefore, this

[1] 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.

kind of items satisfy one of the following conditions:

1. $\gamma \in \boldsymbol{A'}$, $\delta \neq \varepsilon$, $(p,q) \neq (-,-)$ and δ spans the string $a_{i+1} \ldots a_p \, \mathbf{F}^\gamma \, a_{q+1} \ldots a_j$
2. $\delta \neq \varepsilon$, $(p,q) = (-,-)$ and δ spans the string $a_{i+1} \ldots a_j$.
3. $\delta = \varepsilon$, $(p,q) = (-,-)$, adj = true and there exists a sequence of strongly left auxiliary trees that have been adjoined at N^γ. In this case, i and j indicate the portion of the input string spanned by the strongly left auxiliary trees adjoined at N^γ.

- A subset $\mathcal{I}^{(b)}_{\text{Mix}}$ with items of the form $[N^\gamma \to \bullet\upsilon, j, j \mid -, - \mid \text{false}]$ such that $M^\gamma \to \delta\nu \in \mathcal{P}(\gamma)$, $\gamma \in \boldsymbol{I} \cup \boldsymbol{A}$ and any auxiliary tree has been adjoined on N^γ.

The hypotheses defined for this parsing system encode the input string $a_1 \ldots a_n$ in the standard way:

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

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

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

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

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

Then, a set of deduction steps in $\mathcal{D}^{\text{Pred}}_{\text{Mix}}$ and $\mathcal{D}^{\text{Comp}}_{\text{Mix}}$ traverse each elementary tree, while steps in $\mathcal{D}^{\text{Scan}}_{\text{Mix}}$ and $\mathcal{D}^{\varepsilon}_{\text{Mix}}$ scan input symbols and the empty string, respectively:[2]

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

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

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

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

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 $\mathcal{D}^{\text{LAdjPred}}_{\text{Mix}}$ starts the traversal

[2] The conditions checked by steps in $\mathcal{D}^{\text{Comp}}_{\text{Mix}}$ correspond to the special cases of mandatory adjunction and forbidden adjunction.

of β. When β has been completely traversed, a step in $\mathcal{D}_{\text{Mix}}^{\text{LAdjComp}}$ starts the traversal of the subtree corresponding to M^γ. Simultaneous adjunction of several strongly left auxiliary trees on a node M^γ is achieved by repeating this process for each tree.

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

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

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 $\mathcal{D}_{\text{Mix}}^{\text{RAdjPred}}$ starts the traversal of the tree β. When β has been completely traversed, a step in $\mathcal{D}_{\text{Mix}}^{\text{RAdjComp}}$ updates the input positions spanned by M^γ taking into account the part of the input string spanned by β. Simultaneous adjunction of several strongly right auxiliary trees on a node M^γ is achieved by repeating this process for each tree.

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

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

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 $\mathcal{D}_{\text{Mix}}^{\text{LRFoot}}$.

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

A step in $\mathcal{D}_{\text{Mix}}^{\text{AdjPred}}$ predicts the adjunction of an auxiliary tree $\beta \in \boldsymbol{A'}$ in 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 $\mathcal{D}_{\text{Mix}}^{\text{FootPred}}$, 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 $\mathcal{D}_{\text{Mix}}^{\text{FootComp}}$ re-takes the traversal of β continuing at the foot node. When the traversal of β is completely finished, a deduction step in $\mathcal{D}_{\text{Mix}}^{\text{AdjComp}}$ checks if the subtree attached to the foot of β corresponds with the adjunction node. The adjunction if finished by a step in $\mathcal{D}_{\text{Mix}}^{\text{Comp}}$, 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 [6], splitting the completion of adjunction between $\mathcal{D}_{\text{Mix}}^{\text{AdjComp}}$ and $\mathcal{D}_{\text{Mix}}^{\text{Comp}}$.

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

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

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

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

Simultaneous adjunction of several auxiliary trees in $\beta \in \boldsymbol{A'}$ is achieved by applying steps in $\mathcal{D}^{\text{AdjPred}}_{\text{Mix}}$ taking as antecedent the consequent item of a $\mathcal{D}^{\text{FootPred}}_{\text{Mix}}$ step.
The subset $\mathcal{D}^{\text{Comb}}_{\text{Mix}}$ is needed to put together the results corresponding to the simultaneous adjunctions of strongly left and wrapping auxiliary trees:

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

The input string belongs to the language defined by the grammar if and only 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 An example

Figure 4 illustrate 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:

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

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 $\mathcal{D}^{\text{AdjPred}}_{\text{Mix}}$ predicts the adjunction of β_{w1}, generating the item $[\top \to \bullet\mathbf{R}^{\beta_{w1}}, j_2, j_2 \mid -, - \mid \text{false}]$.

3. When the foot node of β_{w1} is reached at position j_3, a $\mathcal{D}^{\text{FootPred}}_{\text{Mix}}$ step generates the item $[M^{\gamma} \to \bullet\upsilon, j_3, j_3 \mid -, - \mid \text{false}]$, which is taken as antecedent by a step in $\mathcal{D}^{\text{AdjPred}}_{\text{Mix}}$ to start the adjunction of β_{w2}, generating the item $[\top \to \bullet\mathbf{R}^{\beta_{w2}}, j_3, j_3 \mid -, - \mid \text{false}]$.[3] 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 $\mathcal{D}^{\text{FootPred}}_{\text{Mix}}$, generating the item $[M^{\gamma} \to \bullet\upsilon, j_4, j_4 \mid -, - \mid \text{false}]$.

4. The adjunction of β_{l2} is then predicted by a step in $\mathcal{D}^{\text{LAdjPred}}_{\text{Mix}}$. The completion of this adjunction by a step in $\mathcal{D}^{\text{LAdjComp}}_{\text{Mix}}$ yields the item $[M^{\gamma} \to \bullet\upsilon, j_4, j_5 \mid -, - \mid \text{true}]$. It

[3] It is interesting to remark that the adjunction of strongly left auxiliary trees could also be predicted at this moment, but this is not the case in our example.

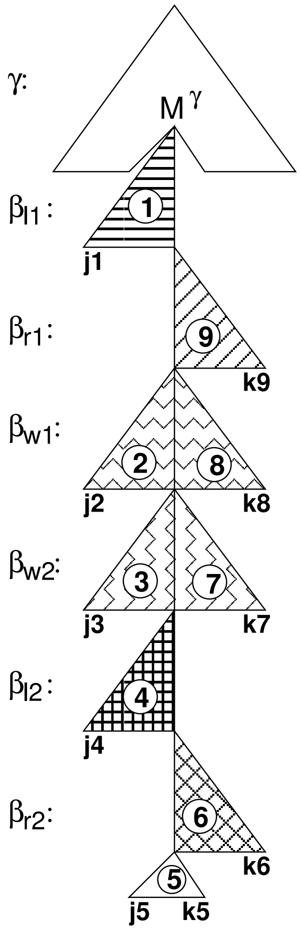

Figure 4: An example of simultaneous adjunctions

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 $\mathcal{D}_{\text{Mix}}^{\text{RAdjPred}}$ starts the adjunction of β_{r2} by generating the item $[\top \rightarrow \bullet\mathbf{R}^{\beta_{r2}}, k_5, k_5 \mid -, - \mid \text{false}]$. The item $[M^\gamma \rightarrow \bullet v, j_4, k_6 \mid -, - \mid \text{true}]$ is produced by a step in $\mathcal{D}_{\text{Mix}}^{\text{RAdjComp}}$ as a result of the completion of this adjunction.

7. At this point, a step in $\mathcal{D}_{\text{Mix}}^{\text{FootComp}}$ re-takes the traversal of β_{w2}, generating the item $[\mathbf{F}^{\beta_{w2}} \rightarrow \bot\bullet, j_4, k_6 \mid j_4, k_6 \mid \text{false}]$ which means that the subtree corresponding to the adjunction node of this auxiliary tree is expected to span the substring $a_{j_4+1} \dots a_{k_6}$. 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 $\mathcal{D}_{\text{Mix}}^{\text{AdjComp}}$ 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. A step in $\mathcal{D}_{\text{Mix}}^{\text{FootComp}}$ re-takes the traversal of β_{w1}, generating the item $[\mathbf{F}^{\beta_{w1}} \rightarrow \bot\bullet, j_3, k_7 \mid j_3, k_7 \mid \text{false}]$ which means that the subtree corresponding to the adjunction node of this

auxiliary tree is expected to span the substring $a_{j_3+1} \dots a_{k_7}$. The adjunction of β_{w1} is finished by a step in $\mathcal{D}^{\text{AdjComp}}_{\text{Mix}}$, yielding the item $[M^\gamma \to v\bullet, j_2, k_8 \mid -, - \mid \text{true}]$.

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

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

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

   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 Complexity

The worst-case space complexity of the algorithm is $\mathcal{O}(n^4)$, as at most four input positions are stored into items corresponding to auxiliary trees belonging to $\boldsymbol{A'}$. 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 steps in $\mathcal{D}^{\text{LAdjComp}}_{\text{Mix}}$ and $\mathcal{D}^{\text{RAdjComp}}_{\text{Mix}}$.
- Full TAG adjunction is managed in $\mathcal{O}(n^6)$ by deduction steps in $\mathcal{D}^{\text{AdjComp}}_{\text{Mix}}$, 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 deduction steps in $\mathcal{D}^{\text{Comp}}_{\text{Mix}}$. The same complexity is attained by the adjunction of a strongly right auxiliary tree on a spine or right node of a wrapping auxiliary tree, due to deduction steps in $\mathcal{D}^{\text{RAdjComp}}_{\text{Mix}}$.
- 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 $\mathcal{O}(n^4)$.

5 Experimental results

We have incorporated the parsing algorithms described in this paper into a naive implementation in Prolog of the deductive parsing machine presented in [8]. As a first experiment, we have compared the performance of the Earley-like parsing algorithms for TIG [7] and TAG [1] with respect to TIGs. For this purpose, we have designed two artificial TIGs G_l (with $\boldsymbol{A}_{SR} = \emptyset$)

Table 1: XTAG results, in seconds, for $\mathrm{P}_{\mathrm{Mix}}$ and $\mathrm{P}_{\mathrm{Mix}_0}$parsers

	Sentence	$\mathrm{P}_{\mathrm{Mix}}$	$\mathrm{P}_{\mathrm{Mix}_0}$	Reduction
(1)	Srini bought a book	0.08	0.13	38.46%
(2)	Srini bought Beth a book	0.11	0.17	35.29%
(3)	Srini bought a book at the bookstore	0.15	0.21	28.57%
(4)	he put the book on the table	0.13	0.18	27.78%
(5)	*he put the book	0.07	0.10	30.00%
(6)	the sun melted the ice	0.11	0.17	35.29%
(7)	the ice melted	0.07	0.10	30.00%
(8)	Elmo borrowed a book	0.08	0.13	38.46%
(9)	*a book borrowed	0.06	0.08	25.00%
(10)	he hopes Muriel wins	0.14	0.21	33.33%
(11)	he hopes that Muriel wins	0.20	0,27	25.93%
(12)	the man who Muriel likes bought a book	0,24	0,32	25.00%
(13)	the man that Muriel likes bought a book	0.21	0.28	25.00%
(14)	the music should have been being played for the president	0.29	0.33	12.12%
(15)	Clove caught a frisbee	0.09	0.12	25.00%
(16)	who caught a frisbee	0.09	0.12	25.00%
(17)	what did Clove catch	0.07	0.13	46.15%
(18)	the aardvark smells terrible	0.07	0.10	30.00%
(19)	the emu thinks that the aardvark smells terrible	0.27	0.32	15.63%
(20)	who does the emu think smells terrible	0.12	0.21	42.86%
(21)	who did the elephant think the panda heard the emu said smells terrible	0.39	0.58	32.76%
(22)	Herbert is angry	0.07	0.09	22.22%
(23)	Herbert is angry and furious	0.09	0.14	35.71%
(24)	Herbert is more livid than angry	0.08	0.12	33.33%
(25)	Herbert is more livid and furious than angry	0.10	0.13	23.08%

and G_r (with $\boldsymbol{A}_{SL} = \emptyset$). 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 ($\mathrm{P}_{\mathrm{Mix}}$) 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 [4], 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. Every sentence has been parsed without previous filtering of elementary trees.

First of all, we have implemented a combined parser $\mathrm{P}_{\mathrm{Mix}_0}$where simultaneous adjunctions are forbidden and we have corroborated the results included in [2]: the parser $\mathrm{P}_{\mathrm{Mix}_0}$ preserves or improves the results obtained by a TAG parser. With this results, we have compared the parser $\mathrm{P}_{\mathrm{Mix}_0}$ with our approach to test the benefits of simultaneous adjunctions. Table 1 shows the results of this experiment including information about the time in seconds spent by both parsers. As we can observe in the table, our approach obtains a reduction in time that varies in percentage from 12% to 46%, 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 $\mathrm{P}_{\mathrm{Mix}_0}$ needs to apply a classical wrapping adjunction.

6 Conclusion

We have defined a parsing algorithm which reduces the practical complexity of TAG parsing by taking into account that a large part of actual TAG grammars can be managed as a TIG. The resulting parser extends the classical adjunction operation in TAG by considering the possibility of simultaneous adjunctions at a given node.

The performance of the algorithm could be improved by means of the application of practical optimizations, such as the replacement of the components p and q of items $[N^\gamma \rightarrow \delta \bullet \nu, i, j \mid p, q] \in \mathcal{I}^{(a)}_{\mathrm{Mix}}$ by the list of all adjunctions that are still under completion on N^γ [3], albeit this modification increase the worst-case complexity of the algorithm. As further work, we are investigating a variant of the algorithm presented in this paper that preserves the correct prefix property [6].

Acknowledgements

The research reported in this paper has been supported in part by Ministerio de Ciencia y Tecnología (grants TIC2000-0370-C02-01, FIT-150500-2002-416, HP2001-0044 and HF2002-0081), Xunta de Galicia (grants PGIDT01PXI10506PN and PGIDIT02SIN01E) and Universidade da Coruña.

References

[1] Miguel A. Alonso, David Cabrero, Eric de la Clergerie, and Manuel Vilares. 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, June 1999.

[2] Miguel A. Alonso, Vicente Carrillo, and Víctor J. Díaz. Mixed parsing of tree insertion and tree adjoining grammars. In Francisco J. Garijo, José C. Riquelme, and Miguel Toro, editors, *Advances in Artificial Intelligence - IBERAMIA 2002*, volume 2527 of *Lecture Notes in Artificial Intelligence*, pages 694–703. Springer-Verlag, Berlin-Heidelberg-New York, 2002.

[3] Eric de la Clergerie. 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, Pittsburgh, PA, USA, June 2001.

[4] Christy Doran, Dania Egedi, Beth Ann Hockey, B. Srinivas, and Martin Zaidel. XTAG system — a wide coverage grammar for English. In *Proc. of the 15th International Conference on Computational Linguistics (COLING'94)*, pages 922–928, Kyoto, Japan, August 1994.

[5] Aravind K. Joshi and Yves Schabes. Tree-adjoining grammars. In Grzegorz Rozenberg and Arto Salomaa, editors, *Handbook of Formal Languages. Vol 3: Beyond Words*, chapter 2, pages 69–123. Springer-Verlag, Berlin/Heidelberg/New York, 1997.

[6] Mark-Jan Nederhof. The computational complexity of the correct-prefix property for TAGs. *Computational Linguistics*, 25(3):345–360, 1999.

[7] Yves Schabes and Richard C. Waters. Tree insertion grammar: A cubic-time parsable formalism that lexicalizes context-free grammar without changing the trees produced. *Computational Linguistics*, 21(4):479–513, December 1995.

[8] Stuart M. Shieber, Yves Schabes, and Fernando C. N. Pereira. Principles and implementation of deductive parsing. *Journal of Logic Programming*, 24(1–2):3–36, July-August 1995.

[9] Klaas Sikkel. *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.