Mixed Parsing of Tree Insertion and Tree Adjoining Grammars

Miguel A. Alonso[1], Vicente Carrillo[2], and Víctor J. Díaz[2]

[1] Departamento de Computación, Universidade da Coruña
Campus de Elviña s/n, 15071 La Coruña (Spain)
alonso@udc.es
[2] Departamento de Lenguajes y Sistemas Informáticos, Universidad de Sevilla
Avda. Reina Mercedes s/n, 41012 Sevilla (Spain)
{carrillo, vjdiaz}@lsi.us.es

Abstract. Adjunction is a powerful operation that makes Tree Adjoining Grammar (TAG) useful for describing the syntactic structure of natural languages. In practice, a large part of wide coverage grammars written following the TAG formalism is formed by trees that can be combined by means of the simpler kind of adjunction defined for Tree Insertion Grammar. In this paper, we describe a parsing algorithm that makes use of this characteristic to reduce the practical complexity of TAG parsing: the expensive standard adjunction operation is only considered in those cases in which the simpler cubic-time adjunction cannot be applied.

1 Introduction

Tree Adjoining Grammar (TAG) [4] and Tree Insertion Grammar (TIG) [6] 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.

Albeit 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 [3] do not make use of such operation, and so a large portion of XTAG is in fact a TIG [6]. 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 paper, we propose a mixed parser 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.

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 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.

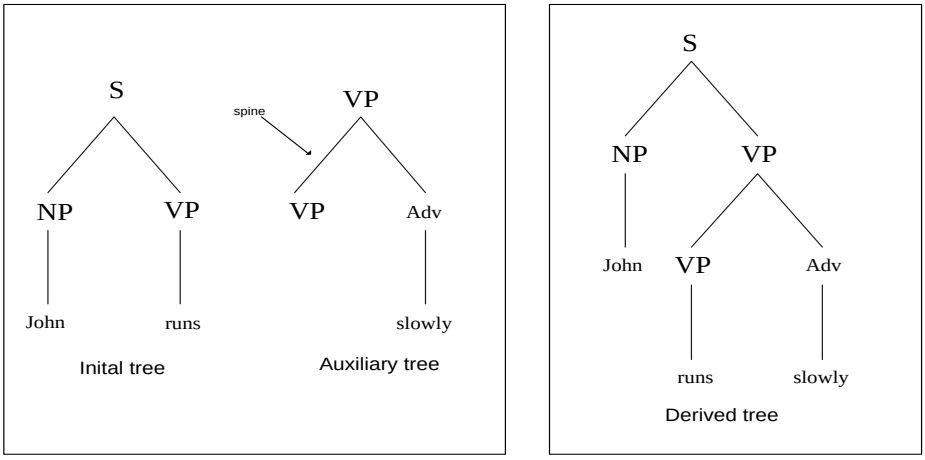

Fig. 1. Adjunction operation

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

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

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 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.

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.

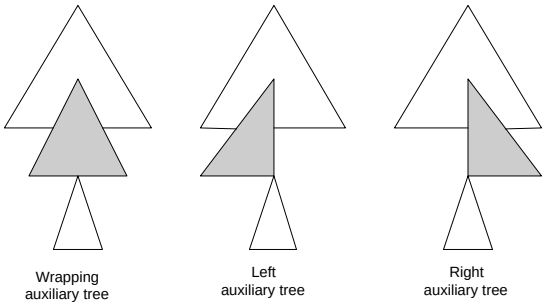

Fig. 2. TAG vs. TIG adjunction operation

1.3 Notation for Parsing Algorithms

We will describe parsing algorithms using *Parsing Schemata*, a framework for high-level descriptions of parsing algorithms [8]. 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 $\frac{\eta_1,\ldots,\eta_k}{\xi}\ cond$, meaning that if all antecedents η_i of a deduction step are present and the conditions *cond* are satisfied, then the consequent ξ should be generated by the parser. A set $\mathcal{F} \subseteq \mathcal{I}$ of *final items* 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 $N_1^\gamma \ldots N_g^\gamma$ are represented by a production $N^\gamma \rightarrow N_1^\gamma \ldots N_g^\gamma$. 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.

2 A Mixed Parser for TIG and TAG

In this section we define a parsing system $_{\text{Mix}} = \langle \mathcal{I}_{\text{Mix}}, \mathcal{H}_{\text{Mix}}, \mathcal{D}_{\text{Mix}} \rangle$ 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, the rest of adjunctions will be managed with the classical deduction steps included in most of TAG parsers [1].

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

- A subset $\mathcal{I}^{(a)}_{\text{Mix}}$ 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 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

[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.

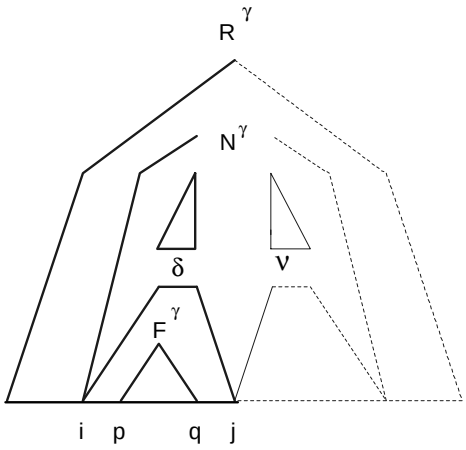

Fig. 3. Graphical representation of items

node of γ if it is a descendant of δ. In other case $p = q = -$ representing they are undefined. Therefore, this kind of items satisfy one of the following conditions:

1. $\gamma \in \boldsymbol{A} - (\boldsymbol{A}_{SL} \cup \boldsymbol{A}_{SR})$, $\delta \neq \epsilon$, $(p, q) \neq (-, -)$ and δ spans the string $a_{i+1} \ldots a_p \, \mathbf{F}^{\gamma} \, a_{q+1} \ldots a_j$
2. $\delta \neq \epsilon$, $(p, q) = (-, -)$ and δ spans the string $a_{i+1} \ldots a_j$.

The last boolean component of items is used to avoid several adjunctions 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 are forbidden. If $adj =$ true and $\nu \neq \epsilon$, it means that a strongly left auxiliary tree $\beta \in \boldsymbol{A}_L$ has been adjoined at N^{γ}. If $adj =$ true and $\nu = \epsilon$, it means that an auxiliary tree has been adjoined at N^{γ}. 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 prescribed by the tree adjoining grammar formalism.

- A subset $\mathcal{I}^{(b)}_{\text{Mix}}$ with items of the form $[N^{\gamma} \rightarrow \bullet \upsilon, j, j \mid -, - \mid \text{false}]$ such that $M^{\gamma} \rightarrow \delta\nu \in \mathcal{P}(\gamma)$, $\gamma \in \boldsymbol{I} \cup \boldsymbol{A}$ and $0 \leq i \leq j$. The last boolean component indicates any tree has been adjoined at N^{γ}.
- A subset $\mathcal{I}^{(b)}_{\text{Mix}}$ with items of the form $[N^{\gamma} \rightarrow \bullet \upsilon, i, j \mid -, - \mid \text{true}]$ such that $M^{\gamma} \rightarrow \delta\nu \in \mathcal{P}(\gamma)$, $\gamma \in \boldsymbol{I} \cup \boldsymbol{A}$, $0 \leq i \leq j$ and there exists a $\beta \in \boldsymbol{A}_{SL}$ such that $\beta \in \text{adj}(N^{\gamma})$ and $\mathbf{R}^{\beta}$ spans $a_{i+1} \ldots a_j$ (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}.

The hypotheses defined for this parsing system encode the input string in the standard way: $\mathcal{H}_{\text{Mix}} = \big\{ [a, i-1, i] \mid a = a_i, 1 \leq i \leq n \big\}$.

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

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

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:

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

Then, a set of deductive 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 $\mathcal{D}^{\epsilon}_{\text{Mix}}$ scan input symbols and the empty symbol, respectively:

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

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

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

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

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

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 of β. When β has been completely traversed, a step in $\mathcal{D}^{\text{LAdjComp}}_{\text{Mix}}$ 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.

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

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

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, and sets the last element of the item to true in order to forbid further adjunctions on this node.

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

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

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

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

A step in $\mathcal{D}_{\text{Mix}}^{\text{AdjPred}}$ predicts the adjunction of an auxiliary tree $\beta \in \boldsymbol{A} - (\boldsymbol{A}_{SL} \cup \boldsymbol{A}_{SR})$ 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 [5], splitting the completion of adjunction between $\mathcal{D}_{\text{Mix}}^{\text{AdjComp}}$ and $\mathcal{D}_{\text{Mix}}^{\text{Comp}}$.

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

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

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

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

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

3 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 $\boldsymbol{A} - (\boldsymbol{A}_{SL} \cup \boldsymbol{A}_{SR})$. 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 $\mathcal{D}_{\text{Mix}}^{\text{RAdjComp}}$ and $\mathcal{D}_{\text{Mix}}^{\text{Comp}}$.
- Full TAG adjunction is managed in $\mathcal{O}(n^6)$ by deduction steps in $\mathcal{D}_{\text{Mix}}^{\text{AdjComp}}$, which are in charge of dealing with auxiliary trees belonging to $\boldsymbol{A} - (\boldsymbol{A}_{SL} \cup \boldsymbol{A}_{SR})$. 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 $\mathcal{D}_{\text{Mix}}^{\text{Comp}}$. 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 $\mathcal{D}_{\text{Mix}}^{\text{RAdjComp}}$.
- 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 $\mathcal{O}(n^4)$.

4 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 [7].

As a first experiment, we have compared the performance of the Earley-like parsing algorithms for TIG [6] and TAG [1] with respect to TIGs. For this purpose, we have designed two artificial TIGs G_l (with $\boldsymbol{A}_{SR} = \emptyset$) 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 $\mathcal{O}(n^2)$ and $\mathcal{O}(n^3)$ complexities of left and right adjunction completion for a TIG parser, respectively. 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 ($_{\text{Mix}}$) 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.

Table 1. XTAG results, in seconds, for the TAG and Mixed parsers

Sentence	TAG	Mixed	Reduction
Srini bought a book	0.61	0.49	19.67%
Srini bought Beth a book	0.77	0.71	7.79%
Srini bought a book at the bookstore	0.94	0.93	1.06%
he put the book on the table	0.83	0.71	14.46%
the sun melted the ice	0.71	0.66	7.04%
the ice melted	0.44	0.38	13.64%
Elmo borrowed a book	0.55	0.49	10.91%
a book borrowed	0.39	0.33	15.38%
he hopes Muriel wins	0.93	0.77	17.20%
he hopes that Muriel wins	1.26	1.16	7.94%
the man who Muriel likes bought a book	2.14	1.48	30.84%
the man that Muriel likes bought a book	1.21	1.04	14.05%
the music should have been being played for the president	1.27	1.26	0.79%
Clove caught a frisbee	0.55	0.49	10.91%
who caught a frisbee	0.55	0.44	20.00%
what did Clove catch	0.60	0.55	8.33%
the aardvark smells terrible	0.44	0.38	13.64%
the emu thinks that the aardvark smells terrible	1.48	1.32	10.81%
who does the emu think smells terrible	0.99	0.77	22.22%
who did the elephant think the panda heard the emu said smells terrible	3.13	2.36	24.60%
Herbert is more livid than angry	0.50	0.44	12.00%
Herbert is more livid and furious than angry	0.50	0.50	0.00%

In a third experiment, we have taken a subset of the XTAG grammar [3], 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. Table 1 shows the results of this experiment. The application of the Mixed parser results in a reduction in time that varies in percentage from 31% to 0%, depending on the kind of trees involved in the analysis of each sentence.

5 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.

This parsing algorithm does not preserve the correct prefix property [5]. It is possible to obtain a variant satisfying this property by means of the introduction of an additional element h into items, which is used to indicate the position of

the input string in which the traversal of the elementary tree involved in each item was started. The worst-case space complexity increases to $\mathcal{O}(n^5)$ but the worst-case time complexity remains $\mathcal{O}(n^6)$ if we modify steps AdjComp_0 and Comp as indicated in [5].

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 $[N^\gamma \rightarrow \delta \bullet \nu, i, j \mid p, q \mid adj] \in \mathcal{I}^{(a)}_{\text{Mix}}$ by the list of all adjunctions that are still under completion on N^γ [2], albeit this modification can increase the worst-case complexity of the algorithm.

Acknowledgements

Supported in part by Plan Nacional de Investigación Científica, Desarrollo e Innovación Tecnológica (TIC2000-0370-C02-01), Ministerio de Ciencia y Tecnología (HP2001-0044) and Xunta de Galicia (PGIDT01PXI10506PN).

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. ACL.
2. 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, CMU, Pittsburgh, PA, USA, June 2001.
3. 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.
4. 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.
5. Mark-Jan Nederhof. The computational complexity of the correct-prefix property for TAGs. *Computational Linguistics*, 25(3):345–360, 1999.
6. 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. Also as Technical Report TR-94-13, June 1994, Mitsubishi Electric Research Laboratories, Cambridge, MA, USA.
7. 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.
8. 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.