A Review of Earley-Based Parser for TIG

Víctor J. Díaz, Vicente Carrillo, and Miguel Toro

Universidad de Sevilla
Departamento de Lenguajes y Sistemas Informáticos
Avda. Reina Mercedes s/n, Sevilla 41012, Spain
{vjdiaz,carrillo}@lsi.us.es

Abstract. Tree Insertion Grammar (TIG) is a compromise between Context-Free Grammars (CFG) and Tree Adjoining Grammars (TAG), that combines the efficiency of the former with the strong lexicalizing power of the latter. In this paper, we present a plain representation of TIG elementary trees that can be used directly as the input grammar for the original Earley parser without the additional considerations established in the Schabes and Waters Earley-based parser for TIG.

1 Introduction

According to Schabes, a formalism is lexicalized when each one of its basic composition structures contains some terminal symbol [6]. The lexicalization is a very interesting property from a linguistic point of view (see Abeille in [1]). In fact, most of the current linguistic theories tend to include information in the lexicon that can be considered purely syntactic. Furthermore, the time complexity of the parsers can be reduced when applying to lexicalized formalisms.

Among well-known formalisms in the literature, TAG defined by Joshi, Levy and Takahashi is a naturally lexicalized formalism [5]. The TAG formalism is a context-sensitive one, therefore, account for an important computational cost compared to CFG, see Nederhof in [7] for more details. In general, the class CFG is not naturally lexicalized, since we can define rules without terminal symbols in its right side. We can transform a CFG into another CFG' that fulfills this condition by a transformation called grammar lexicalization. In the literature, we can find different strategies of CFGs lexicalization that presents different problems [3].

An interesting formalism with respect to the problem of CFG lexicalization is Tree Insertion Grammar (TIG), presented by Schabes and Waters in [8]. TIGs are a compromise between TAGs and CFGs, and are characterized by the following: TIGs are cubic-time parsable as CFGs; TIG grammars are a subclass of TAG grammars, therefore, TIGs are naturally lexicalized. Furthermore, there are lexicalization algorithms that establish the strong equivalence of both formalisms.

A TIG is a 5-tuple (Σ, NT, I, A, S) where Σ is a set of terminal symbols, NT is a set of nonterminal symbols, I is a finite set of finite initial trees, A is a finite set of finite auxiliary trees, and S is a distinguished nonterminal symbol. The set $I \cup A$ is referred to as the set of elementary trees.

In each elementary tree, the root and interior- i.e. nonroot, nonleaf-nodes are labeled with nonterminal symbols. The nodes on the frontier are labeled with terminal symbols, nonterminal symbols or the empty string (ϵ). In initial trees, the nonterminal symbols on the frontier are marked for susbstitution. The root of at least one elementary initial tree must be labeled S. The nonterminal symbols on the frontier of an auxiliary tree are marked for susbstitution, except one nonterminal frontier node marked as the foot. The foot must be labeled with the same label as the root and the mark is normally an asterisk. The path from the root of an auxiliary tree to the foot is called the *spine*.

Auxiliary trees in which every nonempty frontier node is to the left of the foot are called *left* auxiliary trees. Similarly, auxiliary trees in which every nonempty frontier node is to the right of the foot are called *right* auxiliary trees. Other auxilary trees are called *wrapping* auxiliary trees.

Frontier nodes labeled with ϵ are referred to as empty. If all the frontier nodes of an initial tree are empty the tree is referred to as *empty*. If all the frontier nodes other than the foot of an auxiliary tree are empty, the tree is referred to as *empty*.

With respect to operations, substitution replaces a node marked for substitution with an initial tree. Adjunction replaces a node with an auxiliary tree. TIG does not allow there to be any elementary wrapping auxiliary trees or elementary empty auxiliary trees. This ensures that every elementary auxiliary tree will be uniquely either a left auxiliary tree or a right auxiliary tree.

TIG does not allow a left(right) auxiliary tree to be adjoined on any node that is on the spine of a right (left) auxiliary tree. Further, no adjunction whatever is permitted on a node μ that is to the right (left) of the spine of an elementary left(right) auxiliary tree T.

TIG allows arbitrarily many simultaneous adjunctions on a single node. Simultaneous adjunction is specified by two sequences, one of left auxiliary trees and the other of right auxiliary trees that specify the order of strings corresponding to the trees combined.

A TIG derivation starts with an initial tree rooted at S. This tree is repeatedly extended using substitution and adjunction. A derivation is complete when every frontier node in the tree(s) derived is labeled with a terminal symbol. By means of adjunction, complete derivations can be extended to bigger complete derivations.

To eliminate useless ambiguity in derivations, TIG prohibits adjunction: at nodes marked for substitution, because the same trees can be created by adjoining on the root of the trees substituted at these nodes; at foot or root nodes of auxiliary trees, because the same trees can be created by simultaneous adjunction on the nodes the auxiliary trees are adjoined on.

2 Multilayer Elementary Tree Representation

With the purpose of redefining the algorithm of Earley for TIG, Schabes and Waters [8] introduce a multilayer representation of the elementary trees of the

TIG grammars. An elementary tree will be represented by a set of CFG rules. The symbols in the left side of a rule will correspond with the direct ancestor node in an elementary tree. The right side of the rules consists of the sequence of symbols dominated in the elementary tree for the symbol on the left. We will use greek letters μ,ρ,ν to denote nodes in elementary trees and subscripts to indicate the nodes label, e.g., μ_X.

In order to specify the role performed by some nodes in an elementary tree, the set of CFG rules is enriched with the following predicates:

- $Root(\mu_X)$ when a node μ_X is the root of an initial tree
- $LAux(\mu_X)$ when a node μ_X is the root of a left auxiliary tree
- $RAux(\mu_X)$ when a node μ_X is the root of a right auxiliary tree
- $Sub(\mu_X)$ when a node μ_X is marked for substitution
- $Foot(\mu_X)$ when a node μ_X is the foot of an auxiliary tree
- $Adj(\beta, \mu_X)$ when an auxiliary tree β can be adjoined on a node μ_X

Let $G = (\Sigma, NT, I, A, S)$ be a TIG and let $a_1, \ldots, a_n$ be an input string. The Earley-style TIG parser collects states into a set called the chart. A state is a 3-tuple $[p, i, j]$, where p is a position in an elementary tree (i.e in a CFG rule in the set associated to the elementary tree) and $0 \leq i \leq j \leq n$ are integers indicating a span of the input string.

The Earley parser for TIGs (see figure) can be defined using the deductive parsing notation presented by Shieber, Schabes and Pereira [9]. The inference rules (steps) associated to initialization (1), acceptance (13) and scanning (4)(5) are interpreted in the classical sense. Predictor and completor steps are redefined due to the elementary tree representation used. For each kind of operation defined in the TIG formalism is defined a predictor/completor step. Then, six inference rules are grouped in connection with left adjunction (2)(3), right adjunction (11)(12) and substitution (7)(8). These rules use the predicates above to filter those operations that are not adjusted to the formal definition of TIG.

Furthermore, a new type of rule (Subtree Traversal) is needed (9)(10) to traverse correctly the set of CFG rules associated to an elementary tree. The simultaneous adjunction is performed by the steps adjunction predictor (left and right) and an additional scanner rule that ignores the foot node (6) of the auxiliary trees.

3 Plain Elementary Tree Representation

We will introduce a representation of TAGs, presented by Díaz and Toro [2], that tries to reduce the problems presented above. First of all, we will describe an alternative notation for trees that uses a word-based representation instead of the traditional graphical representation. The notation is as follows: a stands for $a \in \Sigma$; $X(t_1 \ldots t_n)$ stands for the elementary tree having root $X \in NT$ and direct subtrees $t_1, \ldots, t_n$. When X has not children we will use the notation X instead of $X()$.

$$Init(\mu_S) \vdash [\mu_S \rightarrow \bullet\alpha, 0, 0] \quad (1)$$

$$[\mu_A \rightarrow \bullet\alpha, i, j] \wedge LAux(\rho_A) \wedge Adj(\rho_A, \mu_A) \vdash [\rho_A \rightarrow \bullet\gamma, j, j] \quad (2)$$

$$[\mu_A \rightarrow \bullet\alpha, i, j] \wedge [\rho_A \rightarrow \gamma\bullet, j, k] \wedge LAux(\rho_A) \wedge Adj(\rho_A, \mu_A) \vdash [\mu_A \rightarrow \bullet\alpha, i, k] \quad (3)$$

$$[\mu_A \rightarrow \alpha \bullet \nu_a\beta, i, j] \wedge a = a_{j+1} \vdash [\mu_A \rightarrow \alpha\nu_a \bullet \beta, i, j+1] \quad (4)$$

$$[\mu_A \rightarrow \alpha \bullet \nu_a\beta, i, j] \wedge a = \epsilon \vdash [\mu_A \rightarrow \alpha\nu_a \bullet \beta, i, j] \quad (5)$$

$$[\mu_A \rightarrow \alpha \bullet \nu_B\beta, i, j] \wedge Foot(\nu_B) \vdash [\mu_A \rightarrow \alpha\nu_B \bullet \beta, i, j] \quad (6)$$

$$[\mu_A \rightarrow \alpha \bullet \nu_B\beta, i, j] \wedge Sub(\nu_B) \wedge Init(\rho_B) \vdash [\rho_B \rightarrow \bullet\gamma, j, j] \quad (7)$$

$$[\mu_A \rightarrow \alpha \bullet \nu_B\beta, i, j] \wedge [\rho_B \rightarrow \gamma\bullet, j, k] \wedge Sub(\nu_B) \wedge Init(\rho_B) \vdash [\mu_A \rightarrow \alpha\nu_B \bullet \beta, i, k] \quad (8)$$

$$[\mu_A \rightarrow \alpha \bullet \nu_B\beta, i, j] \vdash [\nu_B \rightarrow \bullet\gamma, j, j] \quad (9)$$

$$[\mu_A \rightarrow \alpha \bullet \nu_B\beta, i, j] \wedge [\nu_B \rightarrow \gamma\bullet, j, k] \vdash [\mu_A \rightarrow \alpha\nu_B \bullet \beta, i, k] \quad (10)$$

$$[\mu_A \rightarrow \alpha\bullet, i, j] \wedge RAux(\rho_A) \wedge Adj(\rho_A, \mu_A) \vdash [\rho_A \rightarrow \bullet\gamma, j, j] \quad (11)$$

$$[\mu_A \rightarrow \alpha\bullet, i, j] \wedge [\rho_A \rightarrow \gamma\bullet, j, k] \wedge RAux(\rho_A) \wedge Adj(\rho_A, \mu_A) \vdash [\mu_A \rightarrow \alpha\bullet, i, k] \quad (12)$$

$$Init(\mu_S) \wedge [\mu_S \rightarrow \alpha\bullet, 0, n] \vdash Acceptance \quad (13)$$

Fig. 1. Earley-Based Parser for TIGs

We will transform the word representation of $X(t_1 \ldots t_n)$ in a trivially equivalent form $X_L\, t_1 \ldots t_n\, X_R$ for every nonterminal symbol X. In other words, a category symbol X splits into two new non terminal symbols, X_L and X_R, that will divide the left and right side contexts of the symbol.

For example, the plain representation of an initial tree α with the form $S(e)$, will be $S_L\, e\, S_R$ and respectively $S_L\, e\, S_L^*\, S_R^*\, e\, S_R$ for an auxiliary tree β with the form $S(e, S^*, e)$ being S^* the foot node.

In general, the representation of an auxiliary tree β will be of the form: $X_L\, r_1\, X_L^*\, X_R^*\, r_2\, X_R$ where r_1 and r_2 are sequences of symbols, being X and X^* the root and foot symbols. If we observe carefully, we can establish that $X_L\, r_1\, X_L^*$ is just the left contextual tree dominated by the root in β with respect to his foot node. Similarly, $X_R^*\, r_2\, X_R$ will be the right context.

The adjunction operation can also be divided into two sides with respect to the spine of an auxilary tree. Suppose that β is an auxiliary tree X-rooted with frontier $w_L\, X\, w_R$ being w_L and w_R sequences of symbols. Let α be an initial tree that contains a category X with frontier $r_1\, w\, r_2$, where r_1,w,r_2 are sequences of symbols and w is the string that spans the category X. When we adjunct β in α at X we will have the frontier $r_1\, w_L\, w\, w_R\, r_2$. We can see that w_L (resp. w_R) is the string that spans the left (resp. right) contextual tree dominated by X in β.

Briefly, the trees above can be represented using a plain notation as follows:

$$\alpha = S_L\, r_1\, X_L\, w\, X_R\, r_2\, S_R$$

$$\beta = X_L\, w_L\, X_L^*\, X_R^*\, w_R\, X_R$$

With this considerations, the next three CFG-based rules can be stated to translate the elementary trees:

$$S \rightarrow S_L\, r_1\, X_L\, w\, X_R\, r_2\, S_R \text{ rule for } \alpha$$

$$X_L \rightarrow w_L\, X_L^* \text{ rule for } \beta_L$$

$$X_R \rightarrow X_R^*\, w_R \text{ rule for } \beta_R$$

where S is the label of the root of α. The plain representation of β splits into two rules representing left and right contextual sides. We eliminate the reference associated to root symbols in auxiliary trees, because adjunction operation at a root and foot nodes of an auxiliary tree are equivalent.

As we said, in TIG formalism only left and right auxiliary trees exist. If the tree is a left auxiliary tree, the right side produces only an empty string ϵ. Furthermore, it is not allowed any adjunction in the right side of the tree and it can not be adjoined a right auxiliary tree in the spine. These three constraints mean that the right part β_R only generates the empty string and, then, this part is not very important in the definition of a left auxiliary tree. Also, it is not possible to adjoin a left auxiliary tree in the root and foot nodes but we can not eliminate the foot node at all because his presence is necessary in order to obtain multiple adjunctions on the same node. Really, we only need the left side of the foot node to guarantee this kind of adjunctions. With this considerations we can eliminate the right side completely. The no-adjunction is represented by an ϵ-transition. For so much, the left auxiliary trees will have the form:

$$X_L \rightarrow w_L\, X_L^*$$

Respectively, the right auxiliary trees are defined with the following rule:

$$X_R \rightarrow X_R^*\, w_R$$

The nonterminal symbols marked for substitution can not be adjoined. These symbols do not continue the general norm of being divided into two new nonterminal symbols (left and right contexts), in this way we prevent to substitute a rule associated with an auxiliary tree.

This representation presents a fundamental advantage with respect to the multilayer representation: each elementary tree is represented by a rule. This avoids the navigation through the different levels of a tree and, therefore, the definition of the parser is simplified. We will see now how all the valid operations in TIG formalism can be performed using a plain representation.

Left adjunction is equivalent to a substitution in nonterminal symbols subscripted with L. We observe that the constraints that exist on the left adjunction are obtained with this substitution mechanism. It is not allowed to substitute a rule associated with a right auxiliary tree in the symbols associated with the right context of the spine, since these symbols do not exist. Then, the adjoining constraint of an right auxiliary tree in the spine of a left auxiliary tree is ensured.

It is not allowed to substitute a rule associated with a right auxiliary tree in the the right context symbols of left auxiliary tree, since these symbols do not exist in the rule. This guarantees the adjunction constraint of an right auxiliary tree in the right side to a left auxiliary tree. It is possible to adjoin a left auxiliary tree in the foot node. It can seem that this contradict the constraint of adjunction on the foot node, however, this operation is equivalent to a multiple left adjunction on a node in a derived tree.

Right adjunction is equivalent to a substitution in nonterminal symbols subscripted with R. The same as the left adjunction, all the constraints on the right adjunction are maintained. *Simultaneous adjunction* is equivalent to a substitution in L-symbols and later in the R-symbols. With respect to *Substitution*, the rules associated with initial trees are the only ones that can be substituted in the nodes marked for substitution, since the other rules represent auxiliary trees (labeled with right or left contexts).

4 Reviewing Earley-Based Parser for TIGs

In this section we will review the steps included in the parser presented above when it is used a plain representation. After some considerations, we will observe that the obtained parser is equal to the classical Earley parser for CFG [4].

- Scanning. The inference rules (4) and (5) are maintained. The rule (6) is not necessary, since the foot node can be substituted in multiple adjunctions. This process is equivalent to the operation *scan* in the Earley parser for CFG.
- Substitution. The predicates *Sub* and *Init* are not necessary, due to the fact that the substitution operation is already filtered. Therefore, the inference rules (7) and (8) are equivalent to the operations *predictor* and *completor* in the Earley parser for CFG, respectively.
- Subtree Traversal. The navigation of the trees is not necessary and, then, is not needed this operation. The rules (9) and (10) are equal to the operations *predictor* and *completor* in the Earley parser for CFG, respectively.
- Left and Right Adjunctions. Neither the predicates $LAux$ and $RAux$ are necessary, since the auxiliary trees have been labeled with new nonterminal symbols, nor the predicate *Adj*. In fact, the rules (2) and (11) is equivalent to the *predictor* in the Earley parser for CFGs, and the same the rules (3) and (12) respect to the *completor*.

5 Conclusions

The representation of TIG elementary trees can be exploited in order to take advantage of classical definition of CFGs parser. When using a multilayer representation, we must include parser rules to navigate in each elementary tree and predicates to ensure the adjunction constraints. We present an alternative

word-based representation that can be used directly as the input grammar for the original Earley parser for CFGs, without extra considerations.

We argue this alternative representation captures in a more suitable way the evidence of equivalence expresiveness between CFGs and TIGs. Furthermore, the two-sides interpretation of each symbol reflects the three kind of adjunctions included in TIG formalisms.

Adding new nonterminals to the grammar does not represent a problem when constructing the derived tree. We can easily register the adjuctions performed when parsing, since we have only one rule associated with an elementary tree.

References

1. Abeille, A.: Une grammaire lexicalisee d'Arbres adjoints pour le Francais: Application a l'analyse automatique. Ph. D. Universite de Paris 7, France (1991)
2. Díaz, V., Toro, M.: Parsing TAGs with Prolog. Joint Conference on Declarative Programming, AGP'97 eds. Falaschi, M., Navarro, M., and Policrit, A. Grado, Italy (1997) 359-367
3. Díaz, M., Toro, M., Carrillo, V.: Un Algoritmo de Lexicalización de CFG mediante TAGs. Procesamiento del Lenguaje Natural **19** (1996) 201-208
4. Earley, J.: An Efficient Context-Free Parsing Algorithm. ACM **13(2)** (1970) 94-102
5. Joshi, A. K., Levy, L., Takahashi, M.: Tree Adjunct Grammars. Journal of Computer and Systems Science **10** (1975) 136-163
6. Schabes, Y.: Mathematical and Computational Aspects of Lexicalized Grammars. Ph. D. University of Pennsylvania, Philadelphia, USA (1990)
7. Nederhof, M.: Solving the Correct-Prefix Property for TAGs. 5th Meeting of Mathematics of Language. Schloss Dagstuhl, Germany (1997)
8. Schabes, Y., Waters, R.: Tree Insertion Grammars: A Cubic-Time, Parsable Formalism that Lexicalized Context-Free Grammars Without Changing the Trees Produced. Computational Linguistic **21(4)** (1995) 479-515
9. Shieber, S., Schabes, Y., Pereira, F.: Principles and Implementation of Deductive Parsing. Journal of Logic Programming, **4(1&2)** (1995) 3-36