Workshop TAG+5, Paris, 25-27 May 2000

Bidirectional parsing of TAG without heads

Víctor J. Díaz[†], Miguel A. Alonso[‡] and Vicente Carrillo[†]

[†]Facultad de Informática y Estadística, Universidad de Sevilla
Avda. Reina Mercedes s/n, 41012 Sevilla (Spain)
{vjdiaz,carrillo}@lsi.us.es
[‡]Departamento de Computación, Universidad de La Coruña
Campus de Elviña s/n, 15071 La Coruña (Spain)
alonso@dc.fi.udc.es

Abstract

We present a bottom-up bidirectional parser for Tree Adjoining Grammars that is an extension of the parser defined by De Vreught and Honig for Context Free Grammars. Although this parser does not improve the complexity of the parsers defined in the literature, it presents several characteristics that can be of interest for practical parsing of natural languages.

1. Introduction

Several algorithms have been proposed for parsing tree adjoining grammars (TAGs), most of them derived from context-free tabular parsers, ranging from simple bottom-up algorithms, like CYK, to sophisticated extensions of Earley's algorithm (Alonso *et al.*, 1999). However, some of the bidirectional parsers proposed are not applicable in all the cases. Lavelli and Satta parser (1991) is restricted to elementary trees with only one anchor. Van Noord parser (1994) introduces several improvements to Lavelli and Satta parser: the substitution operation, the foot-driven recognition of auxiliary trees and the notion of headed elementary trees in order to take advantage of lexicalization.

According to Van Noord, a headed TAG is a TAG in which each elementary tree is a headed tree. For each internal node in a headed tree, there must be a daughter which is the head of the subtree rooted in that node. The reflexive and transitive closure of the head relation is called the head-corner relation. In order to establish the head-corner relation we must fulfill the following two constraints: (i) the anchor of an initial tree must be a head-corner of the root node of the initial tree and (ii) the foot node of an auxiliary tree must be head-corner of the root of the auxiliary tree. Since there exists the notion of anchor in the context of lexicalized TAG, it seems that the notion of head, as defined by Van Noord, is redundant. Moreover, in the case of anchor siblings the definition of head requires to select only one anchor as the head.

In this paper we present a bidirectional bottom-up parser for TAG, called **dVH**, derived from the context-free parser defined by de Vreught and Honig (de Vreught & Honig, 1989; Sikkel, 1997), which presents several interesting characteristics: (i) the bidirectional strategy allows us to implement the recognition of the adjoining operation in a simple way, (ii) the bottom-up behavior allows us to take advantage of lexicalization, reducing the number of trees under consideration during the parsing process, (iii) in the case of ungrammatical input sentences, the parser is able to recover most of partial parsings according to lexical entries, and (iv) the parser can be applied to every kind of anchored elementary trees without introducing the notion of head

1.1. Notation

Let $G = (V_T, V_N, S, \boldsymbol{I}, \boldsymbol{A})$ be a TAG where V_T and V_N are the terminal and non-terminal alphabets, $S \in V_N$ the axiom symbol, and $\boldsymbol{I}$ and $\boldsymbol{A}$ the set of initial and auxiliary trees respectively. As usual, $\boldsymbol{I} \cup \boldsymbol{A}$ consist of the set 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 TAG by considering each elementary tree γ as formed by a set of context-free productions $\mathcal{P}(\gamma)$: a node N^γ in γ and its g children $N_1^\gamma \dots N_g^\gamma$ are represented by a production $N^\gamma \rightarrow N_1^\gamma \dots N_g^\gamma$. The elements of the productions are the nodes of the tree, except for the case of elements belonging to $V_T \cup \{\varepsilon\}$ in the right-hand side of productions. Those elements may not have children and are not candidates to be adjunction nodes, so we identify such nodes labeled by a terminal or ε with the label[1]. We use $\beta \in \mathrm{adj}(N^\gamma)$ to denote that $\beta \in \boldsymbol{A}$ may be adjoined at node N^γ. If adjunction is not mandatory at N^γ, then $\mathbf{nil} \in \mathrm{adj}(N^\gamma)$. With respect to substitution, we use $\alpha \in \mathrm{sub}(M^\gamma)$ to denote that $\alpha \in \boldsymbol{I}$ can be substituted at node M^γ.

To simplify the description of parsing algorithms we consider additional productions: $\top \rightarrow \mathbf{R}^\alpha$, $\top \rightarrow \mathbf{R}^\beta$ and $\mathbf{F}^\beta \rightarrow \bot$ for each $\alpha \in \boldsymbol{I}$ and each $\beta \in \boldsymbol{A}$, where $\mathbf{R}^\alpha$ is the root node of α and $\mathbf{R}^\beta$ and $\mathbf{F}^\beta$ are the root node and foot node of β, respectively. After disabling $\top$ and $\bot$ as adjunction nodes the generative capability of the grammars remains intact.

2. The parser dVH

The definition of the parser is based on deductive systems similar to *Parsing Schemata* (Sikkel, 1997). Given the input string $w = a_1 \dots a_n$ with $n \geq 0$ and a TAG grammar, the formulas (called items in this context) in the deductive system will be of the form:

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

where $N^\gamma \rightarrow \nu\delta\omega \in P(\gamma)$ is a production decorated with two dots indicating the part of the subtree dominated by N^γ that has been recognized. When ν and ω are both empty, the whole subtree has been recognized. The two indices i and j denote the substring of w spanned by δ. If $\gamma \in \boldsymbol{A}$, p and q are two indices with respect to w indicating the substring spanned by the foot node of γ. In other case $p = q = -$, representing they are undefined.

With respect to deduction steps, we have that

$$\mathcal{D}_{\mathrm{dVH}} = \mathcal{D}_{\mathrm{dVH}}^{\mathrm{Ini}} \cup \mathcal{D}_{\mathrm{dVH}}^{\varepsilon} \cup \mathcal{D}_{\mathrm{dVH}}^{\mathrm{Inc}} \cup \mathcal{D}_{\mathrm{dVH}}^{\mathrm{Conc}} \cup \mathcal{D}_{\mathrm{dVH}}^{\mathrm{Foot}} \cup \mathcal{D}_{\mathrm{dVH}}^{\mathrm{Adj}} \cup \mathcal{D}_{\mathrm{dVH}}^{\mathrm{Subs}}$$

The initializer steps deduce those items associated to productions whose right hand side includes a terminal that matches with an input symbol. The position of the terminal in the input string determines the values of the indices. Empty-productions are considered to match any position in the input string. The indices associated to the foot node in the consequent of both deduction steps are undefined since no foot has been recognized yet:

$$\mathcal{D}_{\mathrm{dVH}}^{\mathrm{Ini}} = \frac{}{[N^\gamma \rightarrow \nu \bullet a \bullet \omega, j-1, j, -, -]} \quad a = a_j$$

$$\mathcal{D}_{\mathrm{dVH}}^{\varepsilon} = \frac{}{[N^\gamma \rightarrow \bullet\bullet, j, j, -, -]}$$

Once the subtree dominated by a node M^γ has been recognized completely, a include step in $\mathcal{D}_{\mathrm{dVH}}^{\mathrm{Inc}}$ continues the bottom-up recognition of the supertree dominated by M^γ when no adjoining

[1] Without lost of generality, we assume that if a node is labeled by ε then it has no siblings.

is mandatory on that node. Indexes are not modified when a step of this type is applied:

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

Given a node N^γ such that $N^\gamma \rightarrow \nu\delta_1\delta_2\omega$, the concatenate steps in $\mathcal{D}^{\text{Conc}}_{\text{dVH}}$ try to combine two partial analysis spanning consecutive parts of the input string, in order to recognize δ_1 and δ_2. The indices i and j in the consequent cover the whole recognized substring. The values of the indices p and q, corresponding to the foot node, are propagated bottom-up:

$$\mathcal{D}^{\text{Conc}}_{\text{dVH}} = \frac{\begin{array}{c}[N^\gamma \rightarrow \nu \bullet \delta_1 \bullet \delta_2\omega, i, j', p, q],\\ [N^\gamma \rightarrow \nu\delta_1 \bullet \delta_2 \bullet \omega, j', j, p', q']\end{array}}{[N^\gamma \rightarrow \nu \bullet \delta_1\delta_2 \bullet \omega, i, j, p \cup p', q \cup q']}$$

where $p \cup q$ is equal to p if q is undefined and is equal to q if p is undefined, being undefined in other case.

The foot steps $\mathcal{D}^{\text{Foot}}_{\text{dVH}}$ introduce in a bottom-up way a new instance of an auxiliary tree β in an adjunction node M^γ where $\beta \in Adj(M^\gamma)$. The recognition of the auxiliary tree begins with the introduction of the foot node. The string spanned by the node M^γ between position k and l determines the values of the indices in the consequent. The indices p and q in the antecedent are ignored in the consequent because a new adjoining has been introduced. The values of these indices will be considered by adjoining steps in order to conclude the adjoining of β in M^γ:

$$\mathcal{D}^{\text{Foot}}_{\text{dVH}} = \frac{[M^\gamma \rightarrow \bullet\delta\bullet, k, l, p, q]}{[\mathbf{F}^\beta \rightarrow \bullet\bot\bullet, k, l, k, l]} \;\; \beta \in \text{adj}(M^\gamma)$$

When the recognition of the auxiliary tree β reaches the root node, the adjoining steps $\mathcal{D}^{\text{Adj}}_{\text{dVH}}$ conclude the adjoining on M^γ, continuing the bottom-up recognition of the supertree of γ with respect to M^γ. This step is only applied when the string spanned by the foot node of β is equal to the string spanned by the adjunction node M^γ. Indices p and q in the consequent are obtained from the antecedent associated to the adjunction node. Now, the string spanned by the adjunction node M^γ corresponds with the string spanned by the root of the auxiliary tree β:

$$\mathcal{D}^{\text{Adj}}_{\text{dVH}} = \frac{\begin{array}{c}[\top \rightarrow \bullet\mathbf{R}^\beta\bullet, j, m, k, l],\\ [M^\gamma \rightarrow \bullet\delta\bullet, k, l, p, q]\end{array}}{[N^\gamma \rightarrow \nu \bullet M^\gamma \bullet \omega, j, m, p, q]} \;\; \beta \in \text{adj}(M^\gamma)$$

A substitution is performed when an initial tree α has been completely recognized. The initial tree establishes the string spanned by the node M^γ where α can be substituted.

$$\mathcal{D}^{\text{Subs}}_{\text{dVH}} = \frac{[\top \rightarrow \bullet\mathbf{R}^\alpha\bullet, i, j, -, -]}{[N^\gamma \rightarrow \nu \bullet M^\gamma \bullet \omega, i, j, -, -]} \;\; \alpha \in \text{sub}(M^\gamma)$$

The input string must belong to the language defined by the grammar, given $\alpha \in \boldsymbol{I}$ rooted with the axiom symbol, whenever an item $[\top \rightarrow \bullet\mathbf{R}^\alpha\bullet, 0, n, -, -]$ is deduced. The algorithm so described is just a recognizer. However, it is not difficult to construct an actual chart parser based on the specification presented above. From the set of derived items (the chart), a parser of the input can be constructed retracing the recognition steps in reverse order or annotating the items computed by the recognition algorithm with information about how they were obtained.

The time complexity of the algorithm with respect to the length n of the input is $O(n^6)$. This complexity is due to deduction steps in $\mathcal{D}^{\text{Adj}}_{\text{dVH}}$ since they present the maximum number of relevant input variables (j, m, k, l, p, q). The space-complexity of the parser is $O(n^4)$ since every item is composed of four input positions ranging on the length of the input string.

The number of items deduced by the parser, as stated before, can be reduced if we apply a filter on the concatenate steps. We can note that these steps produce redundant derivations when the trees are not binary branching. If we do so, the parser obtained will not actually be bidirectional. We will not consider this version because of clarity in the exposition.

3. A new parser dVH'

In the context of parsing lexicalized grammars for natural languages, the parser $\mathbf{dVH}$ can be slightly modified in order to speed up the recognition process. In this way, we will consider the characteristics of the English grammar defined in (XTAG, 1999). The study will be mainly centered on $\mathcal{D}^{\mathrm{Init}}_{\mathrm{dVH}}$, $\mathcal{D}^{\varepsilon}_{\mathrm{dVH}}$ and $\mathcal{D}^{\mathrm{Subs}}_{\mathrm{dVH}}$ deduction steps. We will call $\mathbf{dVH}'$ the new parser obtained after modifications.

First of all, we must note that $\mathcal{D}^{\mathrm{Init}}_{\mathrm{dVH}}$ steps can be applied on anchors as well. In the case of multi-anchors, the step will deduce one item for every anchor in the elementary tree with the suitable positions respect to the input. Furthermore, this step implies an important reduction in the search space since only those elementary trees with anchors matching the input will be consider in the recognition. In this way, $\mathcal{D}^{\mathrm{Foot}}_{\mathrm{dVH}}$ and $\mathcal{D}^{\mathrm{Subs}}_{\mathrm{dVH}}$ deduction steps will not introduce any elementary tree except for those trees considered by $\mathcal{D}^{\mathrm{Init}}_{\mathrm{dVH}}$ deduction steps.

When substitution nodes are siblings of no-substitution nodes the application of $\mathcal{D}^{\mathrm{Subs}}_{\mathrm{dVH}}$ steps can introduce items that are not necessary in order to recognize the input string. The reason is that $\mathcal{D}^{\mathrm{Subs}}_{\mathrm{dVH}}$ deduction steps always try to include an initial tree when this tree is completed. To avoid these redundant substitution operations we can introduce a filter as follows: this step will only applied when M^γ is a substitution node whose daughters do not dominate a terminal or the foot node of γ. In other cases, the substitution operation will be performed by the new deduction steps $\mathcal{D}^{\mathrm{SubsR}}_{\mathrm{dVH}}$ and $\mathcal{D}^{\mathrm{SubsL}}_{\mathrm{dVH}}$.

$$\mathcal{D}^{\mathrm{SubsR}}_{\mathrm{dVH}'} = \frac{\begin{array}{c}[\top \to \bullet \mathbf{R}^\alpha \bullet, j, k, -, -],\\ [N^\gamma \to \nu \bullet \delta \bullet M^\gamma \omega, i, j, p, q]\end{array}}{[N^\gamma \to \nu \bullet \delta M^\gamma \bullet \omega, i, k, p, q]}$$

$$\mathcal{D}^{\mathrm{SubsL}}_{\mathrm{dVH}'} = \frac{\begin{array}{c}[\top \to \bullet \mathbf{R}^\alpha \bullet, i, j, -, -],\\ [N^\gamma \to \nu M^\gamma \bullet \delta \bullet \omega, j, k, j, p, q]\end{array}}{[N^\gamma \to \nu \bullet M^\gamma \delta \bullet \omega, i, k, p, q]}$$

With respect to ε productions, the number of items deduced by $\mathcal{D}^{\varepsilon}_{\mathrm{dVH}}$ steps can be an important drawback in the application of the parser when the grammar has a lot of elementary trees with ε productions. As an example, in the English grammar (XTAG, 1999), it is usual that left hand sides of empty productions present a null adjoining constraint. The practical behavior of the parser can be improved if we filter the steps dealing with empty productions. Given a production $M^\gamma \to \varepsilon$ such that $\{\mathbf{nil}\} = \mathrm{adj}(M^\gamma)$, where $\{\mathbf{nil}\} = \mathrm{adj}(M^\gamma)$ represents a null-adjoining constraint on M^γ and M^γ has at least a daughter that dominates a terminal or the foot node of γ, the following deduction steps

$$\mathcal{D}^{\varepsilon\mathrm{R}}_{\mathrm{dVH}'} = \frac{[N^\gamma \to \nu \bullet \delta \bullet M^\gamma \omega, i, j, p, q]}{[N^\gamma \to \nu \bullet \delta M^\gamma \bullet \omega, i, j, p, q]}$$

$$\mathcal{D}^{\varepsilon\mathrm{L}}_{\mathrm{dVH}'} = \frac{[N^\gamma \to \nu M^\gamma \bullet \delta \bullet \omega, i, j, p, q]}{[N^\gamma \to \nu \bullet M^\gamma \delta \bullet \omega, i, j, p, q]}$$

drastically reduce the number of items generated. When the above constraints are not satisfied, a $\mathcal{D}^{\varepsilon}_{\mathrm{dVH}}$ step must be applied.

Input	**dVH**	**dVH′**	**VN**	**E**	**Ned**
Transitives and Ditransitives					
1	0.16	0.05	0.10	0.33	0.33
2	0.27	0.05	0.16	0.38	0.44
Arguments and Adjuncts					
3	0.38	0.11	0.22	0.49	0.55
4	0.33	0.05	0.16	0.44	0.49
5	0.16	0.01	0.05	0.27	0.33
Ergatives and Intransitives					
6	0.33	0.11	0.22	0.38	0.44
7	0.16	0.05	0.11	0.27	0.27
8	0.16	0.05	0.11	0.33	0.33
9	0.16	0.05	0.11	0.27	0.27
Sentential Complements					
10	0.16	0.05	0.11	0.55	0.44
11	0.22	0.05	0.17	0.66	0.49
Relative Clauses					
12	0.60	0.16	0.38	0.77	0.88
13	0.55	0.16	0.38	0.66	0.77
Auxiliary Verbs					
14	0.60	0.22	0.44	0.66	0.77
Extraction					
15	0.16	0.05	0.16	0.33	0.33
16	0.22	0.05	0.11	0.33	0.33
17	0.16	0.05	0.11	0.38	0.33
Unbounded Dependencies					
18	0.22	0.05	0.11	0.22	0.27
19	0.39	0.11	0.22	0.71	0.61
20	0.28	0.11	0.16	0.55	0.49
21	0.82	0.16	0.49	1.54	1.26
Adjectives					
22	0.11	0.05	0.05	0.22	0.27
23	0.16	0.05	0.11	0.27	0.27
24	0.22	0.05	0.11	0.27	0.27
25	0.33	0.05	0.16	0.33	0.33

Table 1: Parsing time in seconds

4. Experimental results

The results we are going to discuss have been obtained using a naive implementation in Prolog of the deductive parsing machine presented in (Shieber *et al.*, 1995) running on a Pentium II. We have implemented and tested the following parsers: **E** is an Earley-based parser without prefix valid property (Alonso *et al.*, 1999), **Ned** is an Earley-based parser with prefix valid property (Nederhof, 1999), **VN** is the bidirectional parser defined by Van Noord, and **dVH** and **dVH′** are the parsers defined in this paper.

The study is based on the English grammar presented in (XTAG, 1999). >From this document we have selected a subset of the grammar consisting of 27 elementary trees that cover a variety of English constructions: relative clauses, auxiliary verbs, unbounded dependencies, extraction, etc. In order to compare only the behavior of the parsers, we have not consider the feature structures of elementary trees. In this way, we have simulated the features using local constraints. Also, we have selected from the document 25 correct and incorrect sentences grouped with respect to the aspect treated. Every sentence has been parsed without previous filtering of elementary trees. Table 1 shows the time in seconds used for every algorithm and sentence.

From table 1, we can observe that **VN**, **dVH** and **dVH′** obtain better time results than predictive parsers **E** and **Ned**. However, in terms of the more expensive step, the adjoining operation, predictive parsers perform equal or less adjoining operations than bottom-up parsers. Therefore, we can argue that this result is a consequence of the implicit filtering of elementary trees of bottom-up strategies.

On the other hand, we can also note that although **dVH** presents worse time than **VN**, we can see **dVH′** improves the results of **VN**. Since the adjoining operations performed by all the bottom-up parsers are practically the same, we can conclude that this improvement is basically due to the reduction of items removed by the filter in the rules related to ε productions.

5. Conclusion

A bottom-up bidirectional parser for TAG has been defined based on the parser defined by De Vreught and Honig for CFG. The parser does not improve the worst-case bounds of already known parsing methods for TAG but the experiments show similar or better time results than classical parsers. Other benefits can be argued to consider this algorithm of interest in the context of bidirectional parsers. In particular, with respect to Lavelli-Satta parser the **dVH** schema can be applied to multi-anchor auxiliary trees. With respect to Van Noord parser, this new approach does not introduce the concept of head and it is applicable to every kind of anchored elementary tree.

As further work, it would be interesting to investigate the effects of compacting elementary trees, as performed by Lopez (2000), in the real performance of the parser.

6. Acknowledgments

This research was partially supported by the FEDER of EU (Grant 1FD97-0047-C04-02) and Xunta de Galicia (Grant PGIDT99XI10502B).

References

ALONSO M. A., CABRERO D., DE LA CLERGERIE E. & VILARES M. (1999). Tabular algorithms for TAG parsing. In *Proc. of EACL'99, Ninth Conference of the European Chapter of the Association for Computational Linguistics*, p. 150–157, Bergen, Norway: ACL.

DE VREUGHT J. P. M. & HONIG H. J. (1989). *A Tabular Bottom-up Recognizer*. Technical Report 89-78, Department of Applied Mathematics and Informatics, Delft University of Technology, Delft, The Netherlands.

LAVELLI A. & SATTA G. (1991). Bidirectional parsing of lexicalized tree adjoining grammars. In *Proceedings of the 5th Conference of the European Chapter of the Association for Computational Linguistics (EACL'91)*, Berlin, Germany: ACL.

LOPEZ P. (2000). Extended partial parsing for lexicalized tree grammars. In *Proc. of the Sixth International Workshop on Parsing Technologies (IWPT 2000)*, p. 159–170, Trento, Italy.

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

SHIEBER S. M., SCHABES Y. & PEREIRA F. (1995). Principles and implementation of deductive parsing. *Journal of Logic Programming*, **24** (1–2), p. 3–36.

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

VAN NOORD G. (1994). Head-corner parsing for TAG. *Computational Intelligence*, **10** (4), p. 525–534.

THE XTAG RESEARCH GROUP (1999). A lexicalized tree adjoining grammar for English. `http://www.cis.upenn.edu/~xtag`. Technical Report IRCS 95-03, IRCS, Institute for Research in Cognitive Science, University of Pennsylvania