A BIDIRECTIONAL BOTTOM-UP PARSER FOR TAG *

Víctor J. Díaz **Vicente Carrillo**

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

Miguel A. Alonso

Departamento de Computación, Universidad de La Coruña
Campus de Elviña s/n, 15071 La Coruña (Spain)
alonso@dc.fi.udc.es

1 Introduction

Several parsing algorithms have been proposed for Tree Adjoining Grammars (TAGs), most of them derived from context-free tabular parsers [1]. However, only a few bidirectional parsing algorithms have been proposed [3, 5] although this kind of strategy seems to be naturally adapted for TAG. In this paper, we present a new bidirectional bottom-up parser for TAG derived from the context-free parser defined by de Vreught and Honig [4], which has several interesting characteristics: (i) the bidirectional strategy allow us to implement the recognition of the adjoining operation in a simple way, (ii) the bottom-up behavior allow us to take advantage of lexicalization, reducing the number of trees under consideration during the parsing process, and (iii) in the case of ungrammatical input sentences, the parser is able to recover most of partial parsings according to lexical entries.

2 The parser

A Tree Adjoining Grammar is a 5-tuple $\mathcal{G} = (V_N, V_T, S, \boldsymbol{I}, \boldsymbol{A})$, where V_N and V_T are finite sets of non-terminal and terminal symbols, S the axiom of the grammar, and $\boldsymbol{I}$ and $\boldsymbol{A}$ are finite sets of *initial trees* and *auxiliary trees*, respectively. $\boldsymbol{I} \cup \boldsymbol{A}$ is the set of *elementary trees*. The reader is referred to [2] for details in the definition and properties of TAGs. Given an elementary tree γ, a node N^γ and its g children $N_1^\gamma \ldots N_g^\gamma$ are represented by a production $N^\gamma \to N_1^\gamma \ldots N_g^\gamma$. $\mathcal{P}(\gamma)$ is the set of productions formed in such a way for every node in γ. Terminal symbols and ε will be represented in the productions with the proper symbol. Also, we consider additional productions: $\top \to \mathbf{R}^\alpha$, $\top \to \mathbf{R}^\beta$ and $\mathbf{F}^\beta \to \bot$ for each initial tree α and each auxiliary tree β, 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. The nodes $\top$ and $\bot$ are considered no-adjoining nodes.

We will define the parser using the *Parsing Schemata* framework [4]. To this end, we assume the reader is familiar with this kind of parser specification. Let be the input string $w = a_1 \ldots a_n$ with

This work was partially supported by the FEDER of EU (project 1FD97-0047-C04-02) and Xunta de Galicia (projects PGIDT99XI10502B and XUGA20402B97).

$n \geq 0$, the parser works with items $[N^\gamma \rightarrow \nu \bullet \delta \bullet \omega, i, j, p, q]$, where $N^\gamma \rightarrow \nu\delta\omega \in \mathcal{P}(\gamma)$ is decorated with two dots indicating the part, δ, 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 indicate that part of w spanned by δ. If $\gamma \in \boldsymbol{A}$, the indices p and q indicate that part of w spanned by the foot node of γ. In other case $p = q = -$, representing they are undefined.

The deduction steps of the parser are the following:

$$\mathcal{D}^{\text{Ini}} = \frac{[a, j-1, j]}{[N^\gamma \rightarrow \nu \bullet a \bullet \omega, j-1, j, -, -]}$$

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

$$\mathcal{D}^{\epsilon} = \frac{}{[N^\gamma \rightarrow \bullet\bullet, j, j, -, -]}$$

$$\mathcal{D}^{\text{Foot}} = \frac{[M^\gamma \rightarrow \bullet\delta\bullet, k, l, p, q]}{[\mathbf{F}^\beta \rightarrow \bullet\bot\bullet, k, l, k, l]}$$

$$\mathcal{D}^{\text{Inc}} = \frac{[M^\gamma \rightarrow \bullet\delta\bullet, i, j, p, q]}{[N^\gamma \rightarrow \nu \bullet M^\gamma \bullet \omega, i, j, p, q]}$$

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

The steps $\mathcal{D}^{\text{Init}}$ and $\mathcal{D}^{\epsilon}$ are the *initializer* deduction steps in charge of starting the parsing process. The steps $\mathcal{D}^{\text{Inc}}$ and $\mathcal{D}^{\text{Con}}$ traverse elementary trees when no adjoining is performed. Finally, given $\beta \in \text{adj}(M^\gamma)$, the steps $\mathcal{D}^{\text{Foot}}$ and $\mathcal{D}^{\text{Adj}}$ are used to recognize the adjoining operation, 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.

Given $\alpha \in \boldsymbol{I}$, the input string w belongs to the language defined by the grammar when an item $[\top \rightarrow \bullet\mathbf{R}^\alpha\bullet, 0, n, -, -]$ is deduced. 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 adjoining step since presents the maximum number of relevant input variables (j, m, k, l, p, q). With respect to the size of the grammar, the time-complexity is $O(|G|^2)$, since at most two elementary trees are simultaneously considered in a single step. 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.

References

[1] Alonso, M. A., D. Cabrero, E. de la Clergerie and M. Vilares. 1999. Tabular algorithms for TAG parsing. *Proc. of EACL'99*, pages 150–157, Bergen, Norway.

[2] Joshi, A. K. and Y. Schabes. 1997. 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.

[3] Lavelli, A. and G. Satta. 1991. Bidirectional parsing of lexicalized tree adjoining grammars. *Proc. of EACL'91*, Berlin, Germany.

[4] Sikkel, K. 1997. Parsing Schemata — A Framework for Specification and Analysis of Parsing Algorithms. Springer-Verlag, Berlin/Heidelberg/New York.

[5] van Noord, G. 1994. Head-corner parsing for TAG. *Computational Intelligence*, 10(4):525–534.