Comparing tabular parsers for Tree Adjoining Grammars

Víctor J. Díaz and Miguel A. Alonso

Abstract

Most of tabular parsing algorithms for tree adjoining grammars has been demonstrate to attain a theoretical worst-case time complexity of $\mathcal{O}(n^6)$, where n is the length of the input string. In this work we study the experimental complexity of a variety of tabular parsers for tree adjoining grammars for the case of formal and natural language grammars.

1 Introduction

Tree Adjoining Grammars (TAGs) [4] are tree rewriting systems that can be considered an extension of context-free grammars (CFGs) where the basic structures are trees instead of productions and the main composition operation is adjoining instead of substitution. Adjoining is a more powerful operation than substitution, as a consequence:

1. The class of languages recognized by tree adjoining grammars is larger than the class recognized by context-free grammars.
2. The time complexity of parsers increases from $\mathcal{O}(n^3)$ to $\mathcal{O}(n^6)$ and the space complexity raises from $\mathcal{O}(n^2)$ to $\mathcal{O}(n^4)$, where n is the length of the input string, due to the management of the adjoining operation.

Most of parsing algorithms proposed for tree adjoining grammars are derived from tabular parsing algorithms previously defined for context-free grammars. Given this relationship, we could suppose that the general guidelines in the behavior of CFGs parsers are preserved in TAGs parsers. We must reconsider this assertion since the special characteristics of tree adjoining grammars introduce new facets not present in context-free parsers. In the case of lexicalized tree adjoining grammars, this fact is more evident because every tree must include a frontier node labeled with a terminal symbol. In this way, we consider of interest to study the practical behavior of the most popular parsing algorithms for TAGs.

Víctor J. Díaz Madrigal, Departamento de Lenguajes y Sistemas Informáticos, Universidad de Sevilla, Avda. Reina Mercedes s/n, 41012 Sevilla (Spain), `http://www.lsi.us.es/~vjdiaz/`, `vjdiaz@lsi.us.es`

Miguel A. Alonso Pardo, Departamento de Computación, Universidad de La Coruña, Campus de Elviña s/n, 15071 La Coruña (Spain), `http://www.dc.fi.udc.es/~alonso/`, `alonso@dc.fi.udc.es`

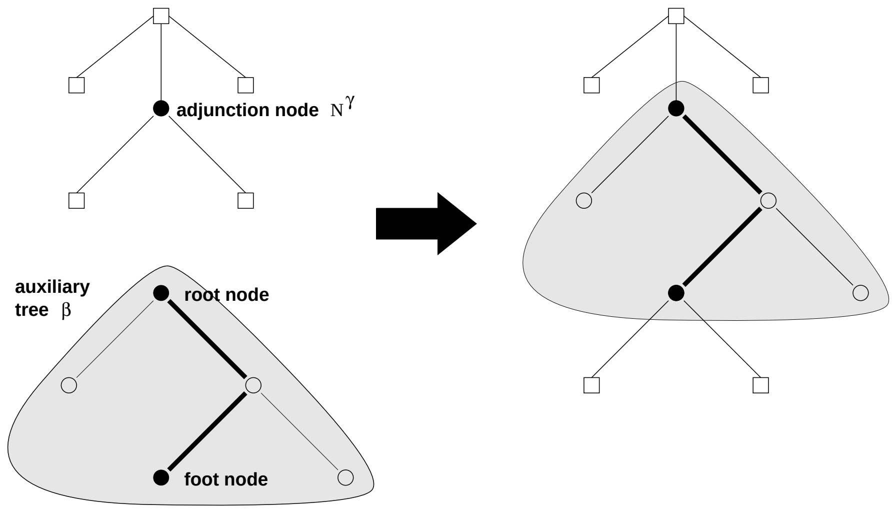

Figure 1: Adjoining operation

2 Tree adjoining grammars

A tree adjoining grammar 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 word 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 auxiliary tree from the root node to the foot node is called the *spine* of the tree.

New trees are derived by *adjoining*: let γ be a tree containing a node N^γ labeled by $A \in V_N$ and let β be an auxiliary tree whose root and foot nodes are also labeled by A. Then, as is depicted in figure 1, the adjoining 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.

In the case of lexicalized tree adjoining grammars, where each elementary tree is anchored by a lexical item, the operation of *substitution* is also considered. In this case, non-terminals can also label leaf nodes (called substitution nodes) of elementary trees. An initial tree can be substituted at a substitution node if its root is labeled by the same non-terminal that labels the substitution node.

3 Parsing algorithms for tree adjoining grammars

All parsers discussed in this paper have been tested using a common framework, a deductive parsing machine [9], implemented in Prolog and running on a Pentium II computer. The deductive parsing machine is a canonical implementation of chart-based parsers where parsers are defined as item-based deduction systems. Briefly, the definition of item-based deduction systems consists of a set of items (formulas) and a set of deduction steps. As usual, items include information about partial analysis and deduction steps establish how to combine these items in order to obtain new items. Given an input and a grammar, the deductive parsing machine compute every item that can be deduced by the parser. The recognition of the input string is indicated by the presence of some final items.

The parsers we have considered are the following ones:

- A bottom-up extension of the Earley's algorithm that traverses trees and recognizes adjoinings in a bottom-up way. We call **buE** to this algorithm, which can be seen as an extension of the CYK-like algorithm for TAGs working on non binary-branching trees [1].
- An extension of Earley's algorithm that traverses trees in a top-down way and recognizes adjoinings bottom-up [1]. We call **E** to this algorithm.
- An extension of the Earley's algorithm that traverses trees and adjoinings in a top-down way. We call **Sch** to this parser, defined by Schabes and Joshi in [8], which satisfies the valid prefix property [7] at the cost of a theoretical running time $\mathcal{O}(n^7)$ and a theoretical space complexity $\mathcal{O}(n^6)$.
- The extension of the Earley's algorithm defined by Nederhof [5] that preserves the valid prefix property maintaing the standard theoretical running time $\mathcal{O}(n^6)$. However, the theoretical space-complexity increases to $\mathcal{O}(n^5)$. We call **Ned** to this parser.
- The head-driven bidirectional algorithm **N** defined by Van Noord [10]. The notion of head is based on 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. Due to these limitations, we have only considered this algorithm in the case of linguistically motivated tree adjoining grammars.
- The bottom-up bidirectional parser **dVH**, an extension of de Vreught and Honig parser for CFGs, defined in [3] and improved in [2].

In order to test the parsers, we are interested on their practical behaviour with respect to the space and time complexity. Although the size of the grammar affect this behaviour [6], we will mainly consider the length of the input string. Because of every parser is defined as a chart-based deductive system, the recognition is shown as the set of items deduced. Therefore, the isolated study of this set can be considered a general way to test the parsers.

Given a grammar and an input string, we will compare the number of items deduced (space-complexity) and the number of adjoining deduction steps applied (time-complexity). Other deduction steps are not considered because adjoining deduction steps are the more expensive ones with respect to computational complexity.

4 Case study 1

For the first case study we have considered a set of 8 formal or artificial grammars. Each grammar is defined in Table 1 by means of its elementary trees (the full set of

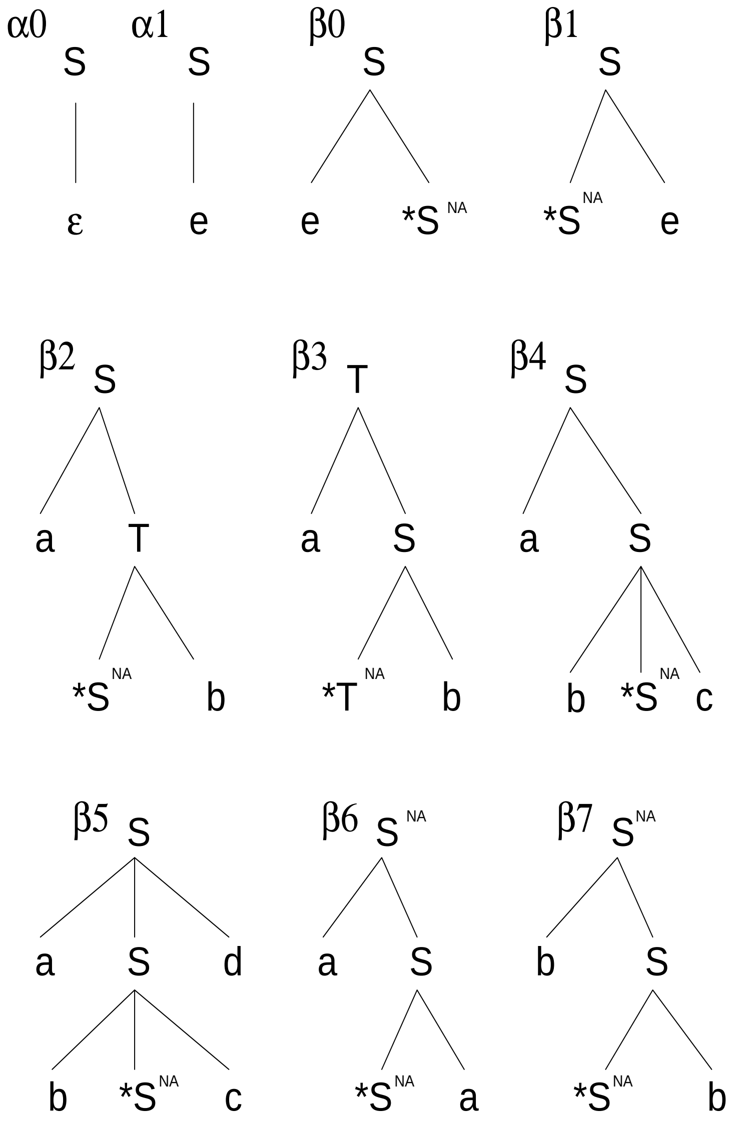

Figure 2: Elementary trees

Grammar	Trees	Language
$G1$	α_1, β_0	e^n with $n > 0$
$G2$	$\alpha_1, \beta_1, \beta_1$	e^n with $n > 0$
$G3$	$\alpha_0, \beta_0, \beta_1$	e^n with $n \geq 0$
$G4$	$\alpha_1, \beta_0, \beta_1$	e^n with $n > 0$
$G5$	$\alpha_0, \beta_2, \beta_3$	$a^n b^n$ with $n > 0$
$G6$	α_1, β_4	$a^n b^n c^n$ with $n > 0$
$G7$	α_1, β_5	$a^n b^n e c^n d^n$ with $n > 0$
$G8$	$\alpha_1, \beta_6, \beta_7$	wew with $w \in \{a, b\}*$

Table 1: Languages generated

elementary trees is shown in Figure 2). Table 1 also shows the language generated by each grammar.

These grammars have been taken as a representative set due to they present several interesting characteristics:

- Simple recursion: grammars presenting this characteristic consist of simple auxiliary trees where adjoining operations can only be applied on elementary tree roots and where the foot node is located on the left-most or right-most position. Grammar $G1$ is left recursive, grammar $G2$ is right recursive, while grammars $G3$ and $G4$ are left and right recursive.

- Adjoining operations on the spine: theoretical time-complexity $O(n^6)$ is reached when adjoining operation is performed on internal nodes located on the spine of auxiliary trees. Grammars $G5$ to $G8$ include auxiliary trees with nodes on the spine, apart from the root and foot, where adjoining operations can be performed. These grammars are well-known examples in the literature that explore the class of languages recognized by tree adjoining languages: $G1$ generates a context-free language while $G6$, $G7$ and $G8$ generate mildly context-sensitive languages.

Figures 3, 4, 5 and 6 show the experimental results for every grammar. The main conclusions derived from the study are:

- Predictive parsers show better results for $G1$ and $G2$ than other parsers. For the case of non-predictive parsers, **dVH** and **buE** present a similar behavior, slightly better for the former.

- The worst behavior is shown for $G3$ and $G4$ grammars, due to the high degree of ambiguity present in these grammars. Also, it is remarkable that the parser **Sch**, although it presents a higher theoretical worst-case time complexity, shows results similar to those obtained by the parser **E**.

- The parser **buE** performs badly in the case of grammars $G5$, $G6$, $G7$ and $G8$. For these grammars the behavior of **dVH** is similar to the behavior of predictive parsers.

- A comparison between **Ned** and **E** seems to show that preserving the valid prefix property does not imply to improve the practical results, at least with respect to space complexity: the behavior of **Ned** with respect to the number of items deduced is worse than the behavior of **E** in the case of grammars $G3$ and $G4$.

In brief, predictive parsers show the best results whereas **dVH**, and specially **buE**, show the worst behavior. This result seems to indicate that top-down filtering improves the performance of parsers. Moreover, the reduction in the number of items deduced can be remarkable when the length of the input string grows. However, this reduction is also a drawback if we are interested in partial parses when the input does not belong to the language defined by the grammar. It is also interesting to emphasize that pure bottom-up strategies tend to increase the number of adjoining

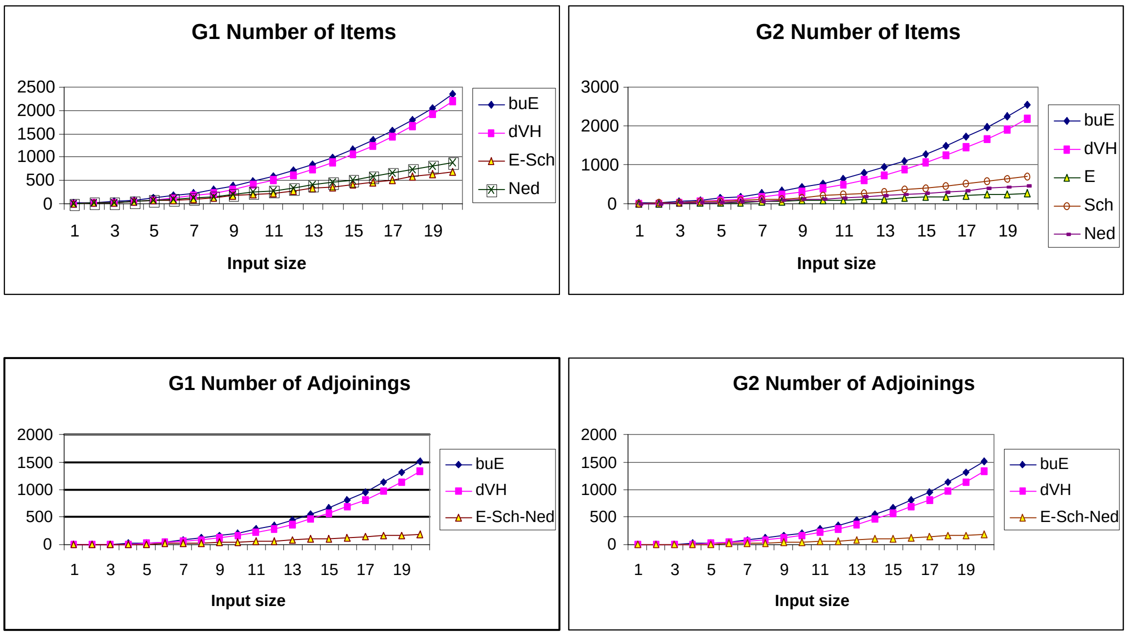

Figure 3: Experimental results for $G1$ and $G2$

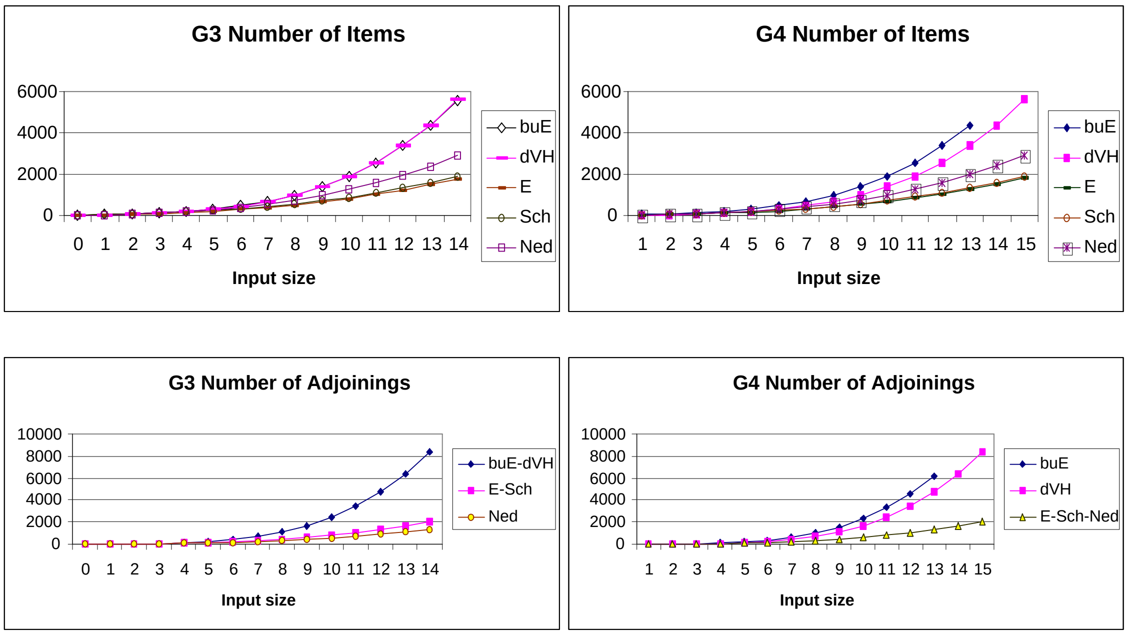

Figure 4: Experimental results for $G3$ and $G4$

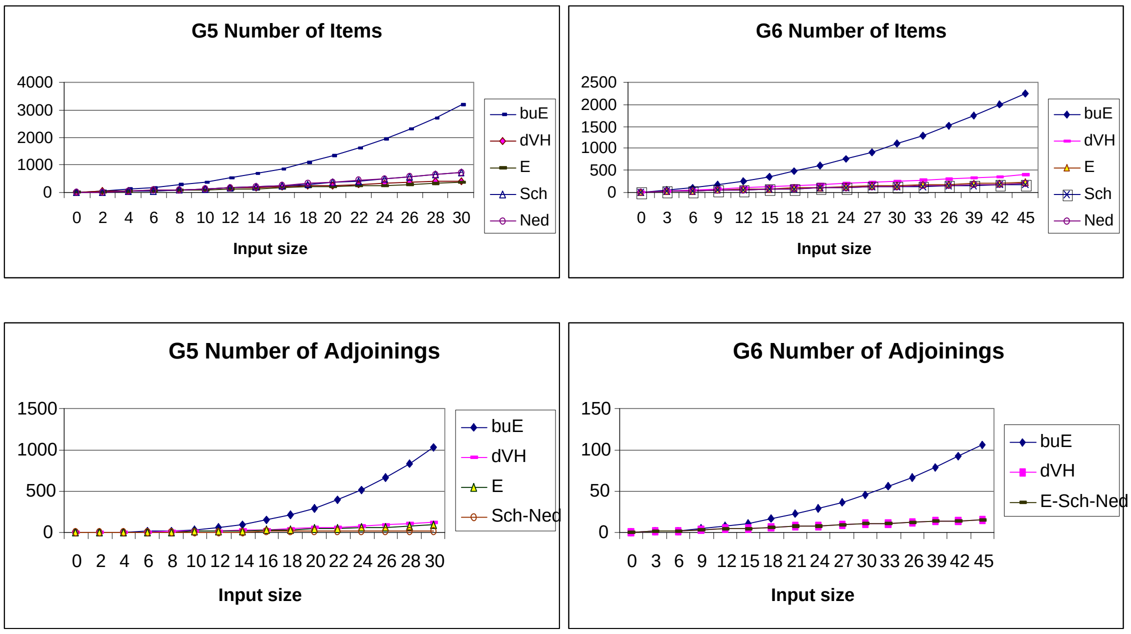

Figure 5: Experimental results for $G5$ and $G6$

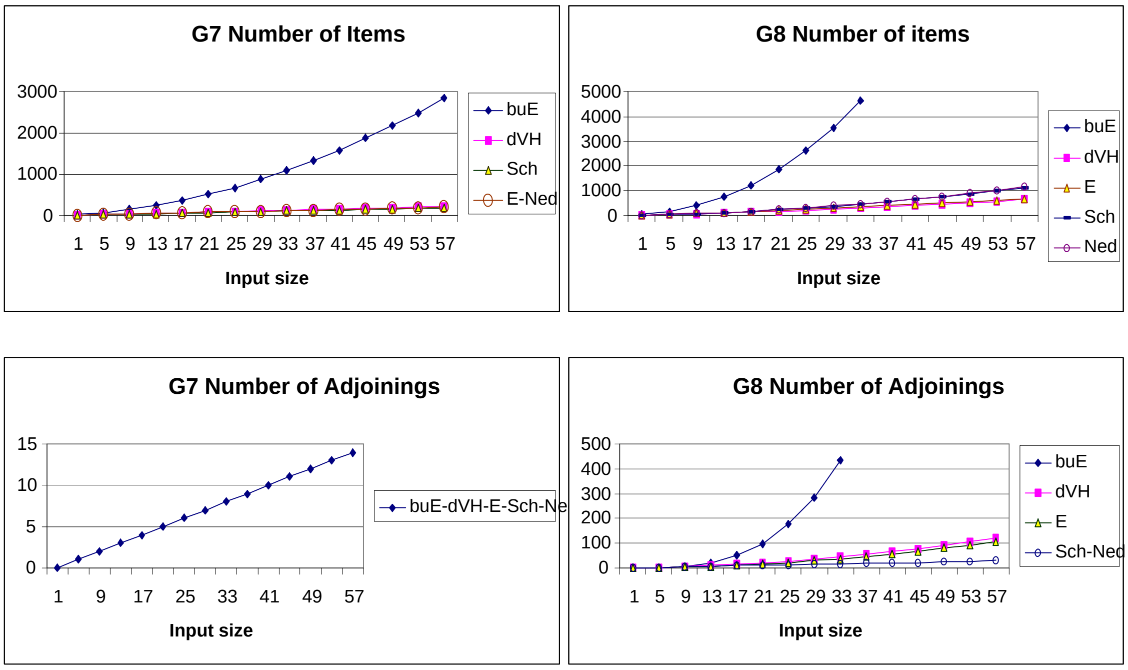

Figure 6: Experimental results for $G7$ and $G8$

operations performed with respect to predictive parsers, at least in the case of simple recursion.

5 Case Study 2

The second study is based on the English grammar presented in [11]. From this document we have selected a subset of the grammar consisting of 27 elementary trees that covers a variety of English constructions. In order to compare only the syntactic behavior of the parsers we have simulated the features structures by means of local constraints. Also, we have selected from the document 25 correct and incorrect sentences grouped with respect to the aspect treated: (1-2) transitives and ditransitives, (3-5) arguments and adjuncts, (6-9) ergatives and intransitives, (10-11) sentential complements, (12-13) relative clauses, (14) auxiliary verbs, (15-17) extraction, (18-21) unbounded dependencies and (22-25) adjectives.

Figure 7 shows the results obtained in the experiment. We have now included the head-driven parser **N** since the grammar is linguistically motivated. Also, in the experiment we have excluded **buE** and **Sch** on behalf of **dVH** and **Ned**, respectively, in order to simplify the experiment. From the study, we can argue that the best results are obtained by the parser **N** despite it performs a higher number of adjoinings than parsers **Ned** and **E**. Bidirectional parsers increase the search space since the recognition starts over every position in the string. However, in the context of lexicalized TAGs this kind of strategies implies a reduction of the trees that are needed during the recognition. The reason is that only those trees that include an anchor that matches with some input symbol are considered. In the case of predictive parsers, we need a previous phase in order to filter the trees that are needed.

The parser **dVH** deduces more items than other parsers. It is probably due to the number of items introduced by ε-leaves (**dVH** is a bottom-up bidirectional parser that is not head-driven). However, the number of adjoining operations performed by **dVH** is similar to the number of adjoinings performed by **N**.

6 Conclusions

Practical behavior of context-free parsing algorithms has been studied in some extend. To our knowledge, this has not been the case with respect to parsers for tree adjoining grammars. This paper is an attempt to introduce some highlights about this issue. We have tested several tabular parser for TAGs with respect to artificial and natural language grammars. Since TAGs are mainly used for the recognition of natural languages, we consider that the study of natural languages grammars is of special relevance. In this way, we can conclude that head-driven bidirectional strategies when parsing lexicalized tree adjoining grammars seems to be of interest because of the reduction of trees considered during the recognition. In order to achieve this reduction in predictive parsers, a previous filter of these trees must be applied.

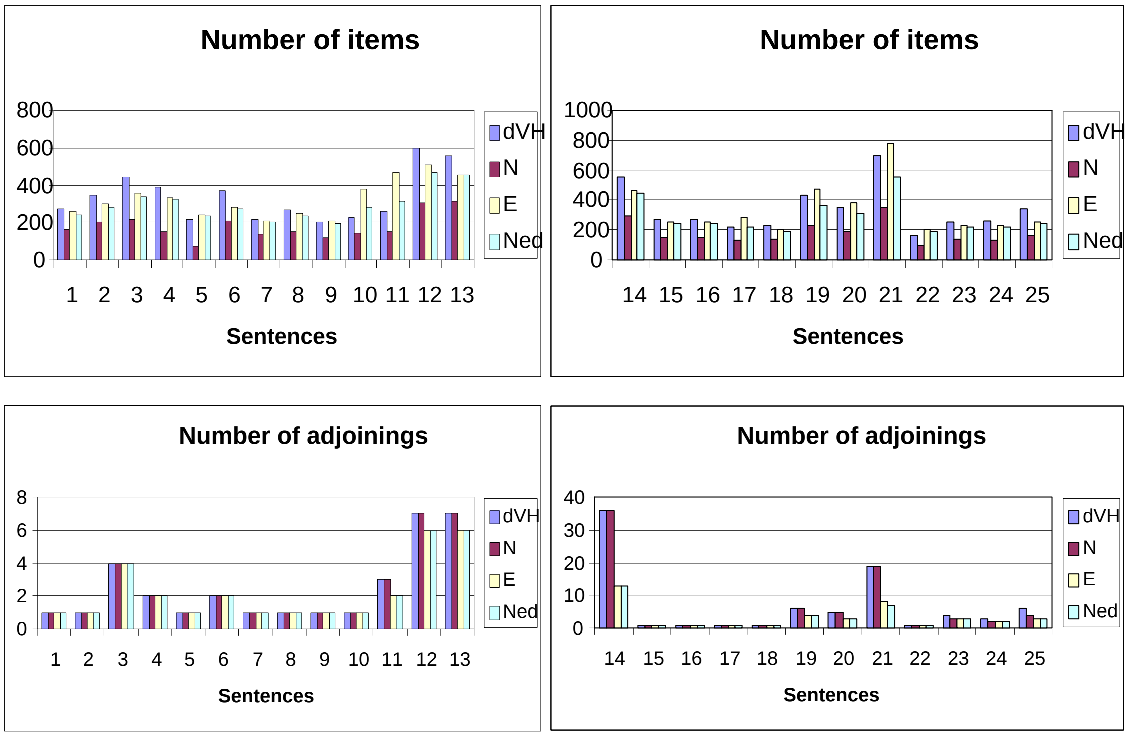

Figure 7: Experimental results for natural language sentences

7 Acknowledgments

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

References

[1] Miguel A. Alonso, David Cabrero, Eric de la Clergerie, and Manuel Vilares. Tabular algorithms for TAG parsing. En *Proc. of EACL'99, Ninth Conference of the European Chapter of the Association for Computational Linguistics*, pp. 150–157, Bergen, Norway, 1999. ACL.

[2] Víctor J. Díaz, Miguel A. Alonso, and Vicente Carrillo. Bidirectional parsing of TAG without heads. In *Proc. of 5th International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+5)*, pp. 67–72, Paris, France, May 2000.

[3] Víctor J. Díaz, Vicente Carrillo, and Miguel A. Alonso. A bidirectional bottom-up parser for TAG. In *Proc. of the Sixth International Workshop on Parsing Technologies (IWPT 2000)*, pages 299–300, Trento, Italy, February 2000.

[4] Aravind K. Joshi and Yves Schabes. Tree-adjoining grammars. In Grzegorz Rozenberg y Arto Salomaa, eds., *Handbook of Formal Languages. Vol 3: Beyond*

Words, chapter 2, pp. 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] Anoop Sarkar. Practical experiments in parsing using tree adjoining grammars. In *Proc. of 5th International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+5)*, pages 193–198, Paris, France, May 2000.

[7] Yves Schabes. The valid prefix property and left to right parsing of tree-adjoining grammar. In *Proc. of II International Workshop on Parsing Technologies, IWPT'91*, pages 21–30, Cancún, Mexico, 1991.

[8] Yves Schabes and Aravind K. Joshi. An Earley-type parsing algorithm for tree adjoining grammars. In *Proc. of 26th Annual Meeting of the Association for Computational Linguistics*, pages 258–269, Buffalo, NY, USA, 1988. ACL.

[9] 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, 1995.

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

[11] 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, Philadelphia PA, USA