Improved Head-Corner Parsing of Tree-Adjoining Grammars

Anoop Sarkar
Dept. of Computer and Information Science
University of Pennsylvania
200 South 33rd Street,
Philadelphia, PA 19104-6389 USA
anoop@linc.cis.upenn.edu

June 23, 2001

Abstract

In this paper we present a chart-based head corner parsing algorithm for Tree Adjoining Grammars (TAGs). We then present an improved algorithm that has a $O(n)$ bound for the number of head-corner predictions as compared to previous approaches which have a bound of $O(n^2)$. The improvement is made possible by changing the definition of head-corner traversal and moving some aspects of top-down prediction in the recognizer into the bottom-up component.

1 Introduction

The parsing algorithm introduced in this paper is a chart-based head-corner algorithm for Tree Adjoining Grammars (TAGs) (see [JLT75, Jos88, JS92] for an introduction to TAGs). The use of head-driven prediction to enchance efficiency was first suggested by [Kay89] for CF parsing (see [Sik97] for a more detailed survey). [LS91] provided the first head-driven algorithm for LTAGs which was a chart-based algorithm but it lacked any top-down prediction. [vN94] describes a Prolog implementation of a head-corner parser for LTAGs which includes top-down prediction. Significantly, [vN94] uses a different closure relation from [LS91]. The head-corner traversal for auxiliary trees starts from the footnode rather than from the anchor.

The parsing algorithm we use is a chart-based variant of the [vN94] algorithm. We use the same head-corner closure relation as proposed there. Our parser differs from the algorithm in [vN94] in some important respects: our algorithm is chart-based and explicitly tracks *goal* and *item* states and does not perform any implicit backtracking or selective memoization, we do not need any additional variables to keep track of which words are already 'reserved' by an auxiliary tree (which [vN94] needs to guarantee termination), and we have an explicit *completion* step.

[Sar00] presents an experiment that measures the performance of an implementation of the head-corner parsing algorithm presented here. The experiment measured performance when dealing with naturally occuring newspaper text from the Wall Street Journal. The experiment

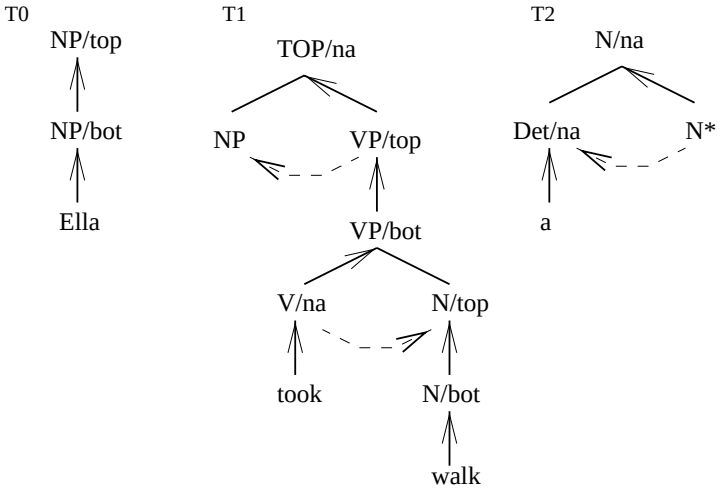

Figure 1: Tree traversal and recognition of substitutions and adjunctions with a head-corner traversal. The figure shows schematically the traversal used to parse the sentence *Ella took a walk*.

used 2250 sentences from the Penn Treebank WSJ, sentence length was at most 21 words and the experiments measured the time and space complexity of the head-corner parsing algorithm.

In this paper, we present an improved head-corner parsing algorithm that has a $O(n)$ bound for the head-corner prediction step as compared to previous approaches which have a $O(n^2)$ bound.

2 Head-Corner Traversal

We take the concept of head-corner traversal for TAGs as defined in [vN94]. We illustrate the head-corner closure relation using an example in this section and define it more formally later.

Each node in an elementary tree in a TAG is associated with a distinguished node in the same tree called the *headcorner*. Parsing is initiated by making top-down predictions on certain nodes and proceeds by moving bottom-up from the headcorner associated with the goal node. This is done recursively producing new goal nodes for siblings and for adjunction prediction. For example, in Figure 1, parsing begins with the prediction that node *S/na* of tree T_1 will span the entire input sentence. The headcorner for node *S/na* is the terminal symbol *took*. The parser then proceeds from that node in the elementary tree T_1 moving up the tree to reach the goal node which is the root node of that tree. Each sibling node is generated as a new goal node before proceeding upwards with the parent of the node. In the figure, the dotted lines are traversed first, before traversing up to the parent node. In the example figure, after visiting the node *V/na* the node *N/top* is introduced as a new goal node which leads to the headcorner node *walk*. Any node where adjunction can occur is represented as two nodes for the parser: *Node/top* and *Node/bottom*. The adjunction is recognized between the bottom and top portions

of the node. In cases where adjunction is prohibited the node is written as *Node/na*. In the example figure, reaching node *N/bot* after moving up from *walk* causes a new goal node, the root node of tree T_2 *N/na* to be instantiated. The headcorner of an auxiliary tree is always the foot node. The auxiliary tree is then recursively ascended using the head-corner traversal until the root node *N/na* is reached. The root node of tree T_2 now spans the input string from $2, 4, 3, 4$ where the foot node spans $3, 4$. The goal is now completed and hence the parser executes a *completion* step and continues traversal in tree T_1 at node *N/top*. Once siblings have been recognized the traversal moves up to the parent of the headcorner node. The other kind of prediction is when a substitution node is reached during the traversal up to the root node. For example, the node *NP* in tree T_1 is a substitution node which introduces a goal node which is the root node *NP/top* of tree T_0. Traversal of tree T_0 occurs as usual. A *completion* step matches the root node of T_0 with the substitution node *NP* and the parser subsequently reaches the first goal node that was predicted: the root node *S/na* of tree T_1. The parser has then successfully parsed the input string *Ella took a walk* (see Figure 1).

For each subtree, the headcorner of the root of the subtree is the leftmost node from the following priority list:

1. Foot Node
2. Anchor Node
3. Substitution Node
4. Terminal Symbol on frontier

3 Chart-based Recognition using Head-Corner Traversal

In this section we give a formal definition of a head-corner parser for TAGs. We will show using an example that it includes a prediction step that produces $O(n^2)$ goal items in the chart. In the next section we will show how the number of items can be reduced to $O(n)$.

We make certain simplifying assumptions without any loss of generality about the nature of the TAGs that are handled by the recognizer/parser. We assume that nodes in each elementary tree of the grammar are either marked *N/na* where no adjunction is permitted on node N or they are marked *N/bot* and *N/top* where *N/top* immediately dominates *N/bot* in the elementary tree which indicates an obligatory adjunction, i.e. an obligatory traversal of an auxiliary tree between nodes *N/bot* and *N/top*. Particular nodes are written as $A/top, B/na, \ldots$ while variables over nodes are written as $\eta/top, \eta'/na, \ldots$ and the symbol η is used to cover node types $\eta/top, \eta'/na, \ldots$

We also assume a single start symbol denoted by *TOP/na* which has to span the entire input string for a valid recognition of the string. We also assume that unary nodes are handled in a

way similar to the binary branching case by insertion of a hidden *unary* sibling for each unary branch in a tree. This also makes the definition of the parse forest easier to define [VSW93].

The recognizer manipulates items of the form:

$$[n, i, j, f_l, f_r], \tag{1}$$

where

- n is a node of some elementary tree γ
- i and j denotes the span of n;
- f_l and f_r denote the spanning in w of the foot node of γ, if γ is an auxiliary tree, and $f_l = f_r = -$ otherwise.

The algorithm also uses goal items to track predictions. Goal items are of the form:

$$[\![n, i, j]\!] \tag{2}$$

where

- n is a node of some elementary tree γ;
- i and j denotes the spanning of the prediction of n;

Parsing begins with the insertion of the first goal state:

$$[\![TOP/na, 0, N]\!] \tag{3}$$

where the input string is ${}_0\ a_0\ {}_1\ \ldots\ a_{N-1}\ {}_N$.

The head-corner recognition algorithm is specified formally as the following deductive items in the style of [SSP95]. The function *hc(η)* returns the headcorner (in the manner defined in Section 1) of the node η.

Head-Corner

$l \leq i \leq r$:

$$\frac{[\![\eta, l, r]\!]}{[hc(\eta)/na, i, i+1, -, -]} \tag{4}$$

Sibling Goal Prediction (right)

$j \geq r$

$$\frac{[\eta, l, r, f_l, f_r]}{[\![\eta'/top, r, j]\!]} \tag{5}$$

Adjunction Prediction

If η' is a footnode and the node-label of η' matches node-label of η:

$$\frac{[\eta/bot, l, r, f_l, f_r]}{[\eta'/na, l, r, l, r]} \tag{6}$$

Adjunction Completion

If η' is the rootnode of an auxiliary tree and the node-label of η' matches node-label of η:

$$\frac{\begin{array}{c}[\eta', l, r, f_l, f_r] \\ [\eta/bot, f_l, f_r, f'_l, f'_r]\end{array}}{[\eta/top, l, r, f'_l, f'_r]} \tag{7}$$

Substitution Prediction

If η' is the rootnode of an initial tree which can be substituted into the node $hc(\eta)$, i.e. the node-label of $hc(\eta)$ matches node-label of η', N is the length of the input string and $i \leq l \leq N$ and $l \leq r \leq N$:

$$\frac{[\eta, i, j, f_l, f_r]}{[\![\eta', l, r]\!]} \tag{8}$$

Substitution Completion

If η' is the rootnode of an initial tree which can be substituted into the substitution node η in some tree, i.e. the node-label of η matches node-label of η':

$$\frac{[\eta', i, j, -, -]}{[\eta, i, j, -, -]} \tag{9}$$

Move to Parent (right head)

When a node η immediately dominates nodes η' and η'' in an elementary and if η'' dominates the footnode (if the tree is an auxilary tree) then:

$$\frac{\begin{array}{c}[\eta'/top, i, k, -, -] \\ [\eta''/top, k+1, j, f_l, f_r]\end{array}}{[\eta, i, j, f_l, f_r]} \tag{10}$$

The recognition algorithm terminates if there are no more items that can be added by applying the above deductive rules or if an item $[TOP/na, 0, N, -, -]$ is found.

Analogous to the deduction rules **Sibling Goal Prediction (right)** and **Move to Parent (right head)** there are symmetric rules for the left sibling cases which are not shown here.

This head-corner recognition algorithm differs from the algorithm in [vN94] in some important respects: our implementation is chart-based and explicitly tracks *goal* and *item* states and does not perform any implicit backtracking or selective memoization, we do not need any additional variables to keep track of which words are already 'reserved' by an auxiliary tree (which

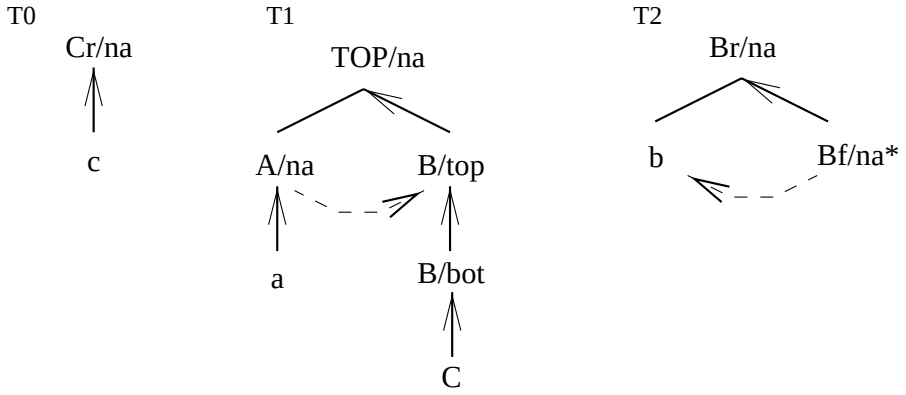

Figure 2: Example grammar that illustrates the complexity of substitution prediction during a head-corner traversal for parsing. Note that subscripts are used to permit reference to treenodes by nodelabels rather than using Gorn addresses. For example, Br, Bf and B all have the same nodelabel B.

[vN94] needs to guarantee termination), and we have explicit *completion* steps for substitution and adjunction. Also the number of goal predictions are explictly recorded. We shall see in the next section how to improve the time complexity of goal predictions in this algorithm.

4 Improving Head-Corner Prediction

In this section we analyse the deductive rule for **Substitution Prediction** which was presented as part of the recognition algorithm. Examining the rule, we can see that the number of goal predictions produced by this step is bounded by $O(n^2)$. This is a major source of inefficiency which intuitively seems to be unnecessary. Let us consider an example to explain why $O(n^2)$ many goal predictions have to made for the recognition algorithm described above to be well-defined.

Consider the grammar given in Figure 2. It consists of trees T_0, T_1 and T_2. This grammar can be used to recognize the string *abc*. T_2 adjoins between nodes *B/top* and *B/bot* in the tree T_1 while T_0 substitutes into the node C in the tree T_1. In recognizing the input string $_0\ a\ _1\ b\ _2\ c\ _3$, the span of node *A/na* in T_1 is $(0, 1)$. Note that the substitution goal prediction for recognizing the substitution of T_0 has to be the set of spans: $(1, 2)(1, 3)(2, 3)$. This is because we have to recognize the adjunction of tree T_2 above the substitution node. Thus, for input *abc* the substitution node has span $(2, 3)$ while the span of the auxiliary tree is $(1, 3, 2, 3)$. This example illustrates the reason why the Substitution Prediction step produces $O(n^2)$ goal items.

Now that we have noticed this drawback of the head-corner traversal, we can proceed to fix it. The solution to reducing the number of goal items produced is to transfer some of the complexity of substitution prediction into the headcorner computation step.

Since we have always assumed that the parser/recognizer should work only with fully lexi-

calized elementary trees, we are assured that each initial tree will have at least one anchor.

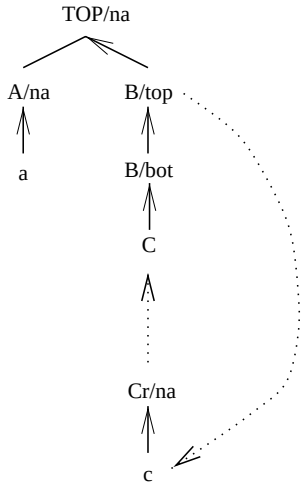

Figure 3: New headcorner computation for internal nodes. The headcorner relation for a node like *B/top* takes into account substitution into nodes that it dominates, like substitution into node C in the tree shown here.

Based on this new headcorner relation, we can now rewrite the Substitution Prediction step so that it will produce at most $O(n)$ items. The goal items are altogether removed. Note that the Substitution Completion step remains unchanged from the earlier algorithm.

Substitution Prediction

If η' is the rootnode of an initial tree, then the anchor of η' is defined as the headcorner of node η. Note that η and η' are in different elementary trees, η dominates a substitution node η'' and the nodelabel of η'' and η' are the same and $l \leq i \leq r$:

$$\frac{[\![\eta, l, r]\!]}{[hc(\eta)/na, i, i+1, -, -]} \tag{11}$$

By using this new definition, the resulting algorithm treats both substitution and adjunction predictions identically. Just as in the original head-corner recognition algorithm, the span of the footnode is always known and never prediction, so too in our modified algorithm, the span of the substitution node in every elementary tree is always known since it is now computed bottom-up. In our modified algorithm, the goal items for substitution are eliminated entirely, and the Substitution Prediction step produces atmost $O(n)$ new items.

5 Conclusion

In this paper we presented a chart-based head corner parsing algorithm for Tree Adjoining Grammars. We point out a limitation of the algorithm in that the number of goal items it produces while performing Substitution Prediction is $O(n^2)$. We present an improved algorithm

that has a $O(n)$ bound for the number of head-corner predictions. The improvement is made possible by moving some aspects of top-down prediction in the recognizer into the bottom-up component. The method we propose has a more complex head-corner relation than those proposed in the past. The new algorithm has larger headcorner tables which also take longer to compute as compared to the previous algorithm. But since the headcorner tables can be computed offline it speeds up the online parsing/recognition time complexity considerably.

References

[JLT75] Aravind K. Joshi, L. Levy, and M. Takahashi. Tree Adjunct Grammars. *Journal of Computer and System Sciences*, 1975.

[Jos88] A. K. Joshi. An introduction to tree adjoining grammars. In A. Manaster-Ramer, editor, *Mathematics of Language*. John Benjamins, Amsterdam, 1988.

[JS92] A. K. Joshi and Y. Schabes. Tree-adjoining grammar and lexicalized grammars. In M. Nivat and A. Podelski, editors, *Tree automata and languages*, pages 409–431. Elsevier Science, 1992.

[Kay89] Martin Kay. Head driven parsing. In *Proc. of IWPT '89*, pages 52–62, Pittsburgh, PA, 1989.

[LS91] Alberto Lavelli and Giorgio Satta. Bidirectional parsing of Lexicalized Tree Adjoining Grammars. In *Proc. 5th EACL*, Berlin, Germany, April 1991.

[Sar00] Anoop Sarkar. Practical experiments in parsing using tree adjoining grammars. In *Proceedings of the Fifth Workshop on Tree Adjoining Grammars, TAG+ 5*, Paris, France, May 25-27 2000.

[Sik97] Klaas Sikkel. *Parsing Schemata*. EATCS Series. Springer-Verlag, 1997.

[SSP95] Stuart Shieber, Yves Schabes, and Fernando Pereira. Principles and implementation of deductive parsing. *Journal of Logic and Computation*, 24(1-2):3–36, 1995.

[vN94] Gertjan van Noord. Head-corner parsing for TAG. *Computational Intelligence*, 10(4), 1994.

[VSW93] K. Vijay-Shanker and D. J. Weir. The use of shared forests in TAG parsing. In *Proc of 6th Meeting of the EACL*, pages 384–393, Utrecht, The Netherlands, 1993.