February 1990

Parsing With Lexicalized Tree Adjoining Grammar

Yves Schabes
University of Pennsylvania

Aravind K. Joshi
University of Pennsylvania, joshi@cis.upenn.edu

Parsing With Lexicalized Tree Adjoining Grammar

Abstract
Most current linguistic theories give lexical accounts of several phenomena that used to be considered purely syntactic. The information put in the lexicon is thereby increased in both amount and complexity: see, for example, lexical rules in LFG (Kaplan and Bresnan, 1983), GPSG (Gazdar, Klein, Pullum and Sag, 1985), HPSG (Pollard and Sag, 1987), Combinatory Categorial Grammars (Steedman, 1987), Karttunen's version of Categorial Grammar (Karttunen 1986, 1988), some versions of GB theory (Chomsky 1981), and Lexicon-Grammars (Gross 1984).

We would like to take into account this fact while defining a formalism. We therefore explore the view that syntactical rules are not separated from lexical items. We say that a grammar is lexicalized (Schabes, AbeilK and Joshi, 1988) if it consists of:

(1) a finite set of structures each associated with lexical items; each lexical item will be called the anchor of the corresponding structure; the structures define the domain of locality over which constraints are specified;

(2) an operation or operations for composing the structures.

The notion of anchor is closely related to the word associated with a functor-argument category in Categorial Grammars. Categorial Grammar (as used for example by Steedman, 1987) are 'lexicalized' according to our definition since each basic category has a lexical item associated with it.

Comments
University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-90-11.

Parsing With Lexcalized Tree Adjoining Grammar

MS-CIS-90-11
LINC LAB 164

Yves Schabes
Aravind K. Joshi

Department of Computer and Information Science
School of Engineering and Applied Science
University of Pennsylvania
Philadelphia, PA 19104-6389

February 1990

Acknowledgements:
This work is partially supported by DARPA grant N00014-85-K-0018, ARO grant DAAL03-89-C-003PRI, NSF grant IRI84-10413 A02.

Parsing with Lexicalized Tree Adjoining Grammar *

Yves Schabes and Aravind K. Joshi
Department of Computer and Information Science
University of Pennsylvania, Philadelphia, PA 19104-6389
schabes/joshi@linc.cis.upenn.edu

Introduction

Most current linguistic theories give lexical accounts of several phenomena that used to be considered purely syntactic. The information put in the lexicon is thereby increased in both amount and complexity: see, for example, lexical rules in LFG (Kaplan and Bresnan, 1983), GPSG (Gazdar, Klein, Pullum and Sag, 1985), HPSG (Pollard and Sag, 1987), Combinatory Categorial Grammars (Steedman, 1987), Karttunen's version of Categorial Grammar (Karttunen 1986, 1988), some versions of GB theory (Chomsky 1981), and Lexicon-Grammars (Gross 1984).

We would like to take into account this fact while defining a formalism. We therefore explore the view that syntactical rules are not separated from lexical items. We say that a grammar is **lexicalized** (Schabes, Abeillé and Joshi, 1988) if it consists of:[1]

- a finite set of structures each associated with lexical items; each lexical item will be called the **anchor**[2] of the corresponding structure; the structures define the domain of locality over which constraints are specified;
- an operation or operations for composing the structures.

The notion of anchor is closely related to the word associated with a functor-argument category in Categorial Grammars. Categorial Grammar (as used for example by Steedman, 1987) are 'lexicalized' according to our definition since each basic category has a lexical item associated with it.

Lexicalized grammars are finitely ambiguous. A sentence (of finite length) selects a finite number of structures which can be combined in finitely many ways since each structure is associated with at least one lexical item. The finite ambiguity of lexicalized grammars is relevant for processing. For example, it ensures that the recognition problem of feature based lexicalized grammars is decidable.

There is a natural general two-step parsing strategy that can be defined for 'lexicalized' grammars. In the first stage, the parser selects a set of elementary structures associated with the lexical items in the input sentence, and in the second stage the sentence is parsed with respect to this set. The strategy is independent of the nature of the elementary structures in the underlying grammar. In principle, any parsing algorithm can be used in the second stage.

The first step selects a relevant subset of the entire grammar, since only the structures associated with the words in the input string are selected for the parser. The number of structures filtered during this pass depends on the nature of the input string and on characteristics of the grammar such as the number of structures, the number of lexical entries, the degree of lexical ambiguity, and the languages it defines.

Since the structures selected during the first step encode the morphological value of their anchor (and therefore its position in the input string), the first step also enables the parser to use non-local bottom-up information to guide its search. The encoding of the value of the anchor of each structure constrains the way the structures can be combined. This information is particularly useful for a top-down component of the parser.

*This paper will appear in *Current Issues in Parsing Technologies* (Tomita, Masaru editor), Kluwer Accademic Publishers.

[1]By 'lexicalized' we mean that in each structure there is a lexical item that is realized. We do not mean simply adding feature structures (such as head) and unification equations to the rules of the formalism.

[2]In previous publications, the term 'head' was used instead of the term 'anchor'. Henceforth, we will use the term anchor instead; the term 'head' introduces some confusion because the lexical items which are the source of the elementary trees need not be the same as the traditional syntactic head of those structures.

This parsing strategy is general and any standard parsing technique can be used in the second step. Since in the worst case, the grammar filtering may select the entire grammar,[3] the strategy does not intrinsically lead to a better upper bound (worst case complexity) of the parsing problem. However, according to our intuition that words anchoring the structures may give useful information for parsing, we have experienced in practice an improvement of the performance of the parser when used on natural language grammars.

We consider Lexicalized Tree Adjoining Grammars as an instance of lexicalized grammars. We take three main types of parsing algorithms: purely top-down (as in definite clause parsing), purely bottom-up (as the CKY-algorithm), bottom-up parsing with top-down information (as the Earley-type parser). For each type, we investigate if the two-step strategy provides any improvement. For the Earley-type parser, we evaluate the two-step strategy with respect to two characteristics. First, the amount of filtering on the entire grammar is considered: once the first pass is performed, the parser uses only a subset of the grammar. Second, we evaluate the use of non-local information: the structures selected during the first pass encode the morphological value (and therefore the position in the string) of their anchor.

1 Lexicalization of CFGs

In the process of lexicalizing a grammar, we require that the 'lexicalized' grammar produce not only the same language as the original grammar, but also the same structures (or tree set).

Not every grammar is in a 'lexicalized' form. Because of its restricted domain of locality, a CFG, in general, will not be in a 'lexicalized' form. The domain of locality of CFGs can be easily extended by using a tree rewriting grammar (Schabes, Abeillé and Joshi, 1988) that uses only substitution as a combining operation. This tree rewriting grammar consists of a set of trees that are not restricted to be of depth one (as in CFGs). Substitution can take place only on non-terminal nodes of the frontier of each tree. Substitution replaces a node marked for substitution by a tree rooted by the same label as the node (see Figure 1; the substitution node is marked by a down arrow $\downarrow$).

CFGs cannot be 'lexicalized', if only tree substitution is used. Even when a CFG can be lexicalized with tree substitution, in general, there is not enough freedom to choose the anchor of each structure. This is important because we want the choice of the anchor for a given structure to be determined on purely linguistic grounds.

If adjunction is used as an additional operation to combine these structures, finitely ambiguous CFGs can be lexicalized. Adjunction builds a new tree from an auxiliary tree β and a tree α . It inserts an auxiliary tree in another tree (see Figure 1). Adjunction is more powerful than substitution. It can weakly simulate substitution, but it also generates languages that could not be generated with substitution.[4]

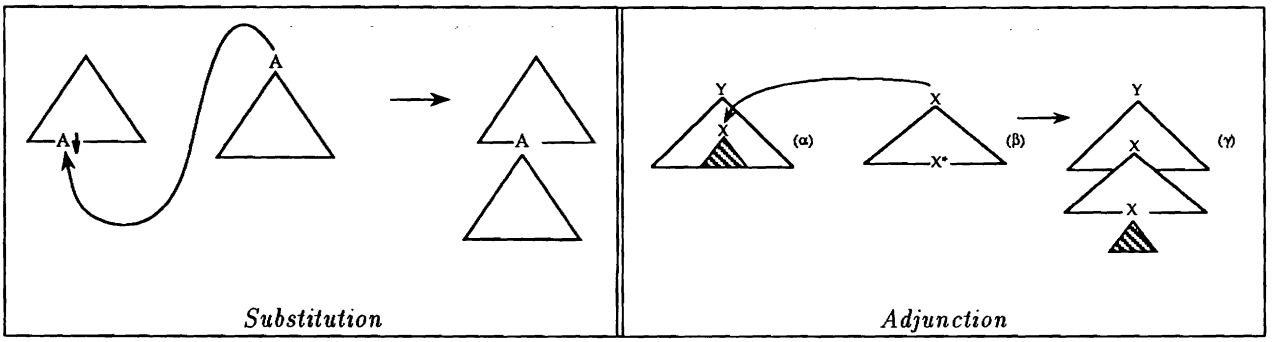


Figure 1: *Combining operations*

[3]There are grammars for which the two-step parsing strategy will not help at all. A grammar generating $\{a^*\}$ in which the elementary structures are headed by a is such an example.

[4]It is also possible to encode a context-free grammar with auxiliary trees using adjunction only. However, although the languages correspond, the set of trees do not correspond.

2 Lexicalized TAGs

Elementary structures of extended domain of locality (each associated with a lexical item) combined with substitution and adjunction yield Lexicalized TAGs. This system falls in the class of mildly context-sensitive languages (Joshi, 1985).

TAGs[5] were first introduced by Joshi, Levy and Takahashi (1975) and Joshi (1985). For more details on the original definition of TAGs, we refer the reader to Joshi (1985,1988), Kroch and Joshi (1985), or Vijay-Shanker (1987). It is known that Tree Adjoining Languages (TALs) are mildly context sensitive. TALs properly contain context-free languages.

A Lexicalized Tree Adjoining Grammar is a tree-based system that consists of two finite sets of trees: a set of initial trees, I and a set of auxiliary trees A (see Figure 2). The trees in $I \cup A$ are called **elementary trees**. Each elementary tree is constrained to have at least one terminal symbol which acts as its anchor.

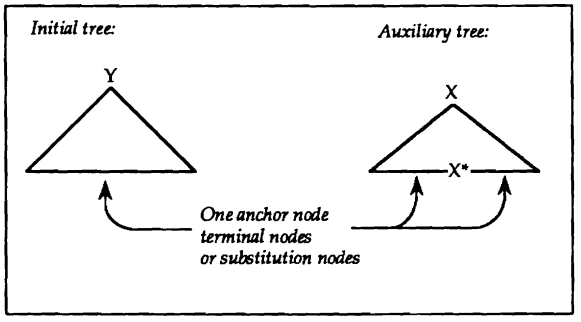


Figure 2: *Schematic initial and auxiliary trees*

The **tree set** of a lexicalized TAG G, $\mathcal{T}(G)$ is defined to be the set of all derived trees starting from S-type[6] initial trees in I (all substitution nodes being filled). The **string language** generated by a TAG, $\mathcal{L}(G)$, is defined to be the set of all terminal strings of the trees in $\mathcal{T}(G)$.

By lexicalizing TAGs, we have associated lexical information to the 'production' system encoded by the TAG trees. We have therefore kept the computational advantages of 'production-like' formalisms (such as CFGs, TAGs) while allowing the possibility of linking them to lexical information (Abeillé, 1990). Formal properties of TAGs hold for Lexicalized TAGs.

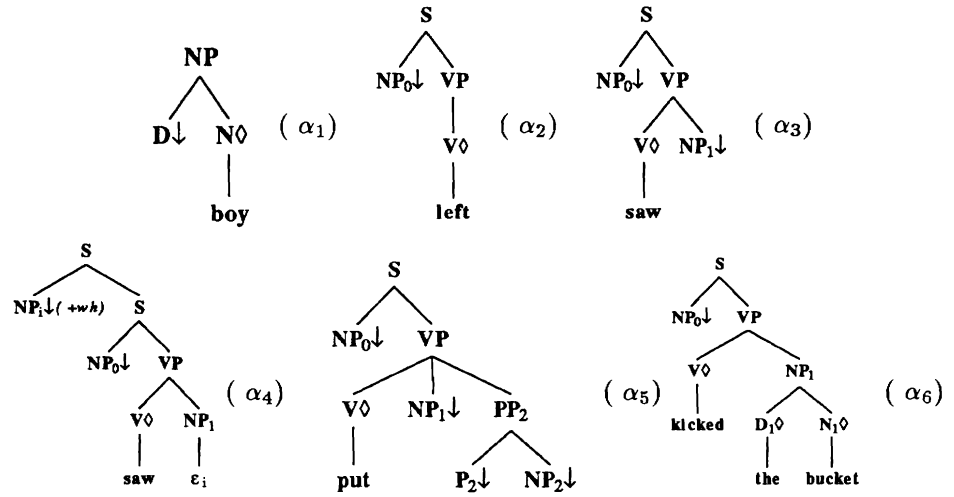
As first shown by Kroch and Joshi (1985), the properties of TAGs permit us to encapsulate diverse syntactic phenomena in a very natural way. TAG's extended domain of locality and its factoring recursion from local dependencies enables us to localize most syntactic dependencies (such as filler-gap) as well as some semantic dependencies (such as predicate-arguments). Abeillé (1988) uses the distinction between substitution and adjunction to capture the different extraction properties between sentential subjects and complements. Abeillé (1988) makes use of the extended domain of locality and lexicalization to account for NP island constraint violations in light verb constructions; in such cases, extraction out of NP is to be expected, without the use of reanalysis. The relevance of Lexicalized TAGs to idioms has been suggested by Abeillé and Schabes (1989,1990).

We will now give some examples of structures that appear in a Lexicalized TAG lexicon. In the following trees, $\downarrow$ is the mark for substitution nodes, $*$ is the mark for the foot node of an auxiliary tree, $\diamond$ is the mark for the node under which the anchor is lexically inserted. We put numerical indices on some non-terminals to express semantic roles. The index shown on the empty string (ϵ) and the corresponding filler in the same tree is for the purpose of indicating the filler-gap dependency.

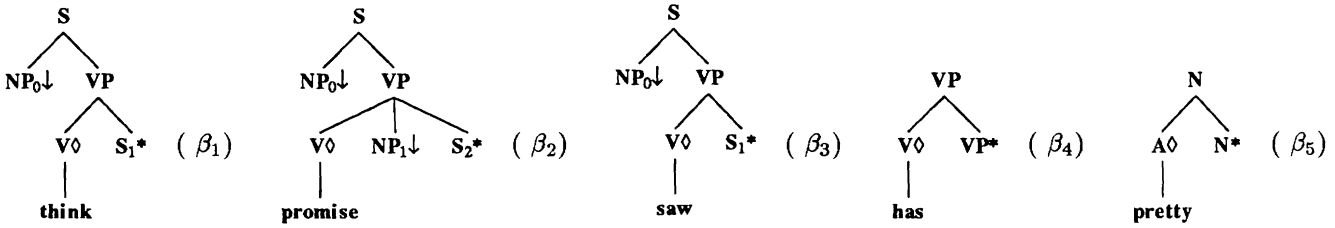
[5] In some earlier work of Joshi (1969, 1973), the use of the two operations 'adjoining' and 'replacement' (a restricted case of substitution) was investigated both mathematically and linguistically. However, these investigations dealt with string rewriting systems and not tree rewriting systems.

[6] Trees whose root is labeled by S.

Some examples of initial trees are (for simplicity, we have omitted unification equations associated with the trees):



Examples of auxiliary trees (they correspond to predicates taking sentential complements or modifiers):

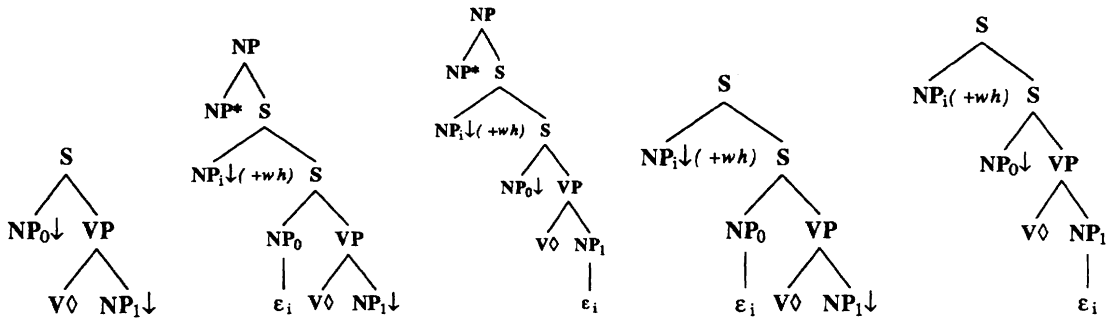


A Lexicalized TAG is organized into two major parts: a **lexicon** and **tree families** (sets of trees). Although it is not necessary to separate trees from their realization in the lexicon, we chose to do so in order to capture some generalities about the structures. TAG's factoring recursion from dependencies, the extended domain of locality of TAGs, and lexicalization of elementary trees make Lexicalized TAG an interesting framework for grammar writing. Abeillé (1988) discusses the writing of a Lexicalized TAG for French. Abeillé, Bishop, Cote and Schabes (1989) similarly discuss the writing of a Lexicalized TAG grammar for English.

A **tree family** is essentially a set of sentential trees sharing the same argument structure abstracted from the lexical instantiation of the anchor. Because of the extended domain of locality of Lexicalized TAG, the argument structure need not be explicitly represented or explicitly enforced since it is implicitly stated in the topology of the trees in a tree family. The syntactic structure is constructed with the lexical value of the predicate and with all the nodes of its arguments. This fact eliminates the redundancy often noted between phrase structure rules and subcategorization frames.[7] Each tree in a family can be thought of as a possible syntactic 'transformation' of a given argument structure. Information (in the form of feature structures) that is valid independently of the value of the anchor is stated on the trees of the tree families. For example, the agreement between the subject and the main verb or auxiliary verb is stated on each tree of the tree family. Currently, the trees in a family are explicitly enumerated.

[7]Optional arguments are stated in the structure.

The following trees, among others, compose the tree family of verbs taking one object (the family is named *np0Vnp1*):[8]



$(\alpha np0Vnp1)$ $(\beta R0np0Vnp1)$ $(\beta R1np0Vnp1)$ $(\alpha W0np0Vnp1)$ $(\alpha W1np0Vnp1)$

$\alpha np0Vnp1$ is an initial tree corresponding to the declarative sentence, $\beta R0np0Vnp1$ is an auxiliary tree corresponding to a relative clause where the subject has been relativized, $\beta R1np0Vnp1$ corresponds to the relative clause where the object has been relativized, $\alpha W0np0Vnp1$ is an initial tree corresponding to a wh-question on the subject, $\alpha W1np0Vnp1$ corresponds to a wh-question on the object.

The **lexicon** associates a word with tree families. Words are not associated with basic categories as in a CFG-based grammar, but with tree-structures corresponding to minimal linguistic structures.

The lexicon also states some word-specific feature structure equations that have to be added to the ones already stated on the trees (such as the equality of the value of the subject and verb agreements).[9] In our approach the category of a word is not a non-terminal symbol but a multi-level structure corresponding to minimal linguistic structures: sentences (for predicative verbs, nouns and adjectives) or phrases (NP for nouns, AP for adjectives, PP for prepositions yielding adverbial phrases).

3 Parsing Lexicalized TAGs

In order to evaluate the two-step parsing strategy, we divide the parsing algorithms in three main types: purely top-down (as in the usual definite clause parsing), purely bottom-up (as the CKY-algorithm), bottom-up parsing with dynamic top-down information (as the Earley-type parser). We will also mention current work on bottom-up parsing with compiled top-down information (as LR-style parsing).

For each algorithm, we discuss the possible advantages provided by the two-step strategies.

Two ways to take advantage of lexicalization

An offline parsing algorithm can take advantage of lexicalization in two ways.

- The trees corresponding to the input string are selected and then the parser parses the input string with respect to this set of trees. We will refer to this process as **grammar filtering**.
- The fact that, after the first pass, each structure encodes the morphological value (and therefore the positions in the string) of its anchor imposes constraints on the way the structures can be combined (the anchor positions must appear in increasing order in the combined structure). We will refer to this information as **bottom-up information**. It seems that this free bottom-up information is particularly useful in a top-down component of the parser.

[8] The trees are simplified. ◊ is the mark for the node under which the anchor word of the tree is attached.

[9] In practice, there is a morphological lexicon that associates the inflection forms of a word to its base form and to their morphological features. Then the syntax associates the base form of the word with tree families or individual trees. Attributes specific to the anchor are passed by unification to the node under which it is lexically inserted.

For example, given the sentence:

The $_1$ **men** $_2$ **who** $_3$ **hate** $_4$ **women** $_5$ **that** $_6$ **smoke** $_7$ **cigarettes** $_8$ **are** $_9$ **intolerant** $_{10}$

the trees shown in Figure 3 are selected (among others) after the first pass.[10]

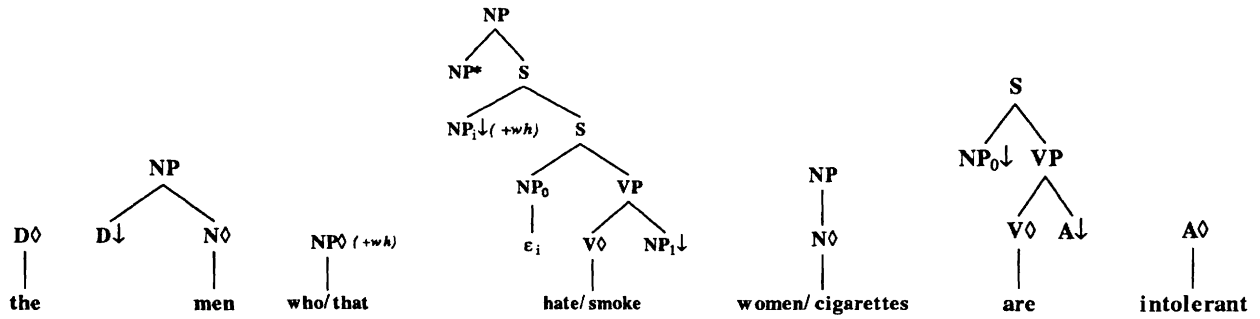


Figure 3: Trees selected by: **The men who hate women that smoke cigarettes are intolerant**

The fact that no adverbial or no bi-transitive trees will be selected after the first pass illustrates the filtering process on the grammar.

The anchor positions of each structure impose constraints on the way the structures can be combined (the anchor positions must appear in increasing order in the combined structure). The tree corresponding to **men** cannot be substituted at NP_1 in the tree selected by **hate** and **smoke** since the anchor positions would not be in the right order. This constitutes an example of the use of bottom-up information.

Internal Representation of Lexicalized Trees

The definition of lexicalized grammar does not imply a particular internal representation of a tree that a specific parsing algorithm may use. Each parsing algorithm may use the representation which is most efficient for it.

We present two ways (see Figure 4) according to which a parsing algorithm may internally represent a lexicalized tree. We will call a **parser-tree**, the internal representation that a parser has of a tree selected by an anchor in a given input string.

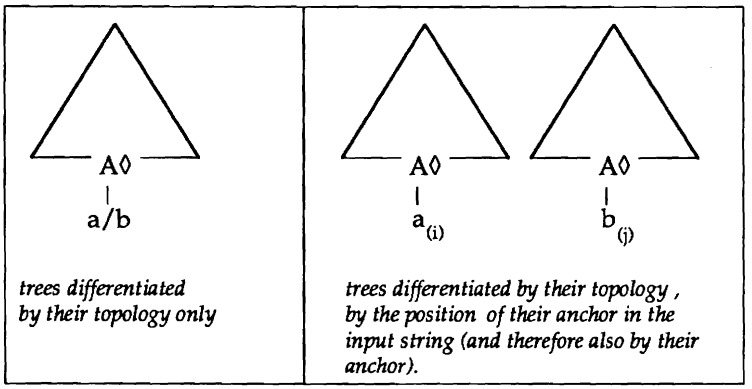


Figure 4: *Internal representations of a lexicalized tree*

The first representation (first box in Figure 4) considers each tree with a pre-terminal node that corresponds to the anchor node. All lexical items that are anchoring a tree can be lexically inserted at the node below which the anchor appears and the tree holds this information as part of its definition. This

[10] The example is simplified to illustrate our point.

representation differentiates trees by their topology only. The representation is close to the usual notion of pre-terminals in CFGs. It does not violate the notion of lexicalization since the tree itself contains the information which specifies which lexical items can appear as anchor. If attributes in the form of feature structures are used, the tree will have attributes stated abstractly from the the anchor and the anchor will specifies attributes that will be passed through the anchor node (See section 3.5). This representation will be used for the CKY-type, the Earley-type and the LR-like parsers for Lexicalized TAGs.

The second representation (second box in Figure 4) differentiates trees by their topology, by the value of their anchor and also by the position of the anchor in the input string. In this case, two identical words but at different positions in the input sentence are not associated to the same trees (since the position of the input differentiate them). This representation multiplies the number of trees (compared to the first one) by the number of terminals and by the length of the input sentence. However, this representation allows the parser to use a specific tree only once in the derivation since a tree corresponds to a specific word at a given position in the input string[11]. This representation will be used by the DCG-type parser for Lexicalized TAGs.

3.1 Bottom-up Parsing

Vijay-Shanker and Joshi (1985) designed a CKY-type parser for TAGs. It is a pure bottom-up parser that uses dynamic programming techniques. Since this algorithm is data driven, the bottom-up information given by the first pass has no effect on the algorithm. However, the grammar filtering fill reduces the number of nodes put in the recognition matrix.

3.2 Top-down Parsing

In a similar manner as CFGs, TAG can be axiomatized with definite clauses. Bernard Lang (1990) uses logical PDAs to interpret with dynamic programming techniques such axiomatization. Four indices instead are required for TAGs (instead of two for CFGs). The four positions correspond to the positions of the strings to the left and right of a foot node of an auxiliary tree (see Figure 5).

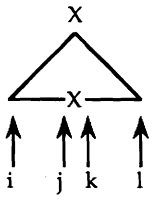


Figure 5: *Four positions needed for an auxiliary tree*

For example, the trees for *left* and *quickly* can be axiomatized as in Figure 6.

`connects` axiomatizes the fact that a terminal spans the substring from position `I` to `J`.

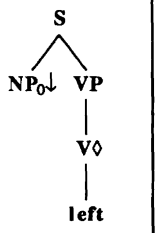
`aux_node(Cat,I,J,K,L)` axiomatizes the fact that an auxiliary tree spanning the substrings $a_i \cdots a_j$ and $a_k \cdots a_l$, can be adjoined at a node labeled by `Cat`. Initial trees span a contiguous substring ($a_i \cdots a_j$) and auxiliary trees span two substrings ($a_i \cdots a_j$ and $a_k \cdots a_l$,).

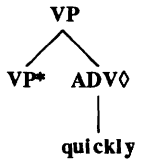
The set of definite clauses can be interpreted in a top-down fashion. As in CFGs, this algorithm loops on left recursive rules. This problem is particularly acute for TAGs since left recursive rules are quite frequent.

However, the two-step parsing strategy enables us to define a top-down interpretation of the axiomatization of a lexicalized TAGs which will halt in all cases. Given an input sentence, the parser considers the trees selected by the first pass. If the parser internally distinguishes the trees by their topology and by the position of their anchor, each parser-tree should be used only once since it corresponds to a word at a unique position in the input string. Therefore each time the recognition of a parser-tree is attempted by the

[11] Using it more then once would imply using the same word more then once in the parse

interpreter, this parser-tree is deleted from the set of parser-trees available. When backtracking is necessary, the parser-tree will be put back into the list of available parser-trees.






```
connects(Terminal,I,J) :- ...
% if the terminal spans the substring from position I to J

% node on which adjunction can be performed
aux_node(_,I,I,L,L).  % no adjunction
aux_node(Cat,I,J,K,L) :-
  aux_tree(Cat,I,J,K,L).

% initial tree left
init_tree(alpha,s,I,L) :-
  aux_node(s,I,J,K,L),
  init_tree(Tree_name,np,,J,J1),
  aux_node(vp,J1,M,N,K),
  aux_node(v,M,Q,P,N),
  connects(left,Q,P).

% auxiliary tree quickly
aux_tree(beta,vp,I,J,K,L) :-
  aux_node(vp,I,J1,K1,L),
  aux_node(vp,J1,J,K,Q),
  aux_node(adv,Q,M,N,K1),
  connects(quickly,M,N).
```

Figure 6: Axiomatization of a TAG

Since lexicalized grammar are finitely ambiguous, the search space of a top-down parser can be made finite. One can therefore design a pure top-down parser that will halt on all cases.

3.3 Bottom-up Parsing with Top-down Information

An Earley-type parser for TAGs has been designed by Schabes and Joshi (1988).[12] As Earley's CFG parser, it is a bottom-up parser that uses top-down information provided by prediction. This parser can take advantage of lexicalization. It uses the structures selected after the first pass to parse the sentence. The parser is able to use the non-local information given by the first step to filter out prediction and completion states. It has been extended to deal with feature structures for TAGs as defined by Vijay-Shanker and Joshi (1988). The extended algorithm we propose always halts when used on Lexicalized TAGs without special devices such as restrictors. Unification equations are associated with both extended linguistic structures and lexical information given by the anchor. This representation allows a more natural and more direct statement of unification equations.

[12]The system is available on Symbolics machines. It includes a graphical tree editor, the Earley-type parser for lexicalized feature-based TAGs and tools for developing lexicons.

Taking Advantage of Lexicalization

An offline version of the Earley-type parser for TAGs can be used with no modification for parsing Lexicalized TAGs. First, the trees corresponding to the input string are selected and then the parser parses the input string with respect to this set of trees.

However, Lexicalized TAGs simplify some cases of the algorithm. For example, since by definition each tree has at least one lexical item attached to it (its anchor), it will not be the case that a tree can be predicted for substitution and completed in the same states set. Similarly, it will not be the case that an auxiliary tree can be left predicted for adjunction and right completed in the same states set.

But most importantly, the algorithm can be extended to make crucial use of the two-stage parsing of Lexicalized TAGs. Once the first pass has been performed, a subset of the grammar is selected. Each structure encodes the morphological value (and therefore the positions in the string) of its anchor. Identical structures with different anchor values are merged together (by identical structures we mean identical trees and identical information, such as feature structures, stated on those trees).[13] This enables us to use the anchor position information while efficiently processing the structures. For example, given the previous sentence:

The $_1$ **men** $_2$ **who** $_3$ **hate** $_4$ **women** $_5$ **that** $_6$ **smoke** $_7$ **cigarettes** $_8$ **are** $_9$ **intolerant** $_{10}$

The trees in Figure 3 (among others) are selected after the first pass. The anchor positions of each structure impose constraints on the way that the structures can be combined (the anchor positions must appear in increasing order in the combined structure). This helps the parser to filter out predictions or completions for adjunction or substitution. For example, the tree corresponding to **men** will not be predicted for substitution in any of the trees corresponding to **hates** or **smoke** since the anchor positions would not be in the right order (see Figure 3).

We have evaluated the influence of the grammar filtering and the use of anchor position information on the behavior of the Earley-type parser. We have conducted experiments on a feature structure-based Lexicalized English TAG whose lexicon defines 200 entries associated with 130 different elementary trees (the trees are differentiated by their topology and their feature structures but not by their anchor value). Twenty five sentences of length ranging from 3 to 14 words were used to evaluate the parsing strategy. For each experiment, the number of trees given to the parser and the number of states were recorded.

In the first experiment (referred to as *one pass, OP*), no first pass was performed. The entire grammar (i.e., the 130 trees) was used to parse each sentence. In the second experiment (referred to as *two passes no anchor, NA*), the two-pass strategy was used but the anchor positions were not used in the parser. And in the third experiment (referred to as *two passes with anchor, A*), the two-pass strategy was used and the information given by the anchor positions was used by the parser.

The average behavior of the parser for each experiment is given in Figure 7. The first pass filtered on average 85% (always at least 75%) of the trees. The grammar filtering alone decreased the number of states $((NA - OP)/OP)$ by 86%. The additional use of the information given by the anchor positions further decreased $((A - NA)/NA)$ the number of states by 50%. The decrease given by the filtering of the grammar and by the information of the anchor positions is even bigger on the number of attempts to add a state (not reported in the table).[14]

This set of experiments shows that the two-pass strategy can greatly increase the performance of the Earley-type parser for TAGs. The more significant factor is the filtering of the grammar. The information given by anchor position in the first pass allows further improvement of the parser's performance (50% reduction of the number of states). The bottom-up non-local information given by the anchor positions improves the top-down component of the Earley-type parser.

We performed our evaluation on a relatively small grammar and did not evaluate the variations across grammars. The lexical degree of ambiguity of each word, the number of structures in the grammar, the number of lexical entries, and the length (and nature) of the input sentences are parameters which certainly must be considered. Although it might appear easy to conjecture the influence of these parameters, the

[13] Unlike our previous suggestions (Schabes, Abeillé and Joshi, 1988), we do not distinguish each structure by its anchor position since it increases unnecessarily the number of states of the Earley parser. The Earley parser enables us to process only once parts of a tree that are associated with several lexical items selecting the same tree.

[14] A state is effectively added to a state set if it does not exist in the set already.

	(NA-OP)/OP (%)	(A-OP)/OP (%)	(A - NA)/NA (%)
# trees	-85	-85	0
# states	-86	-93	-50

Figure 7: *Empirical evaluation of the two-pass strategy*

actual experiments are difficult to perform since statistical data on these parameters are hard to obtain. We hope to perform some limited experiments along those lines.

3.4 LR-style parsing

In LR-style parsing, a machine is driven by a parsing table built before run-time. Although this type of algorithm behaves purely bottom-up, it uses pre-compiled top-down information provided by the parsing table. However, in case of LR-parsing of lexicalized TAGs even when the table is built for the entire grammar before run-time, this strategy may be useful to solve conflicts of moves if they occur.

LR-type parsers for TAGs have been recently proposed by Schabes and Vijay-Shanker (1990). The algorithm drives a machine called Bottom Up Embedded Automaton (a stack of stacks) in a bottom-up fashion. At run-time, before the point where a tree is reduced, two partial reductions can be performed (those partial reductions correspond to partial recognition of auxiliary trees). These partial reductions may occur at a point where the anchor has not yet be scanned. Therefore, the grammar filtering (and also the non-local bottom-up information) can be used to solve at run-time some conflicts of moves. We plan in the near future to collect empirical data to determine to what extent this two-step strategy is useful in solving conflicts of moves in parsing natural language grammars.

3.5 Parsing Feature Based Lexicalized TAGs

As defined by Vijay-Shanker (1987) and Vijay-Shanker and Joshi (1988), feature-based TAGs attach two feature structures to each adjunction node in an elementary tree: a top and a bottom feature structure. When the derivation is completed, the top and bottom feature structures of all nodes are unified simultaneously. If the top and bottom feature structures of a node do not unify, then a tree must be adjoined at that node. This definition can be easily extended to substitution nodes. To each substitution node we attach one feature structure which acts as a top feature structure. The updating of feature structures in the cases of adjunction and substitution is shown in Figure 8.

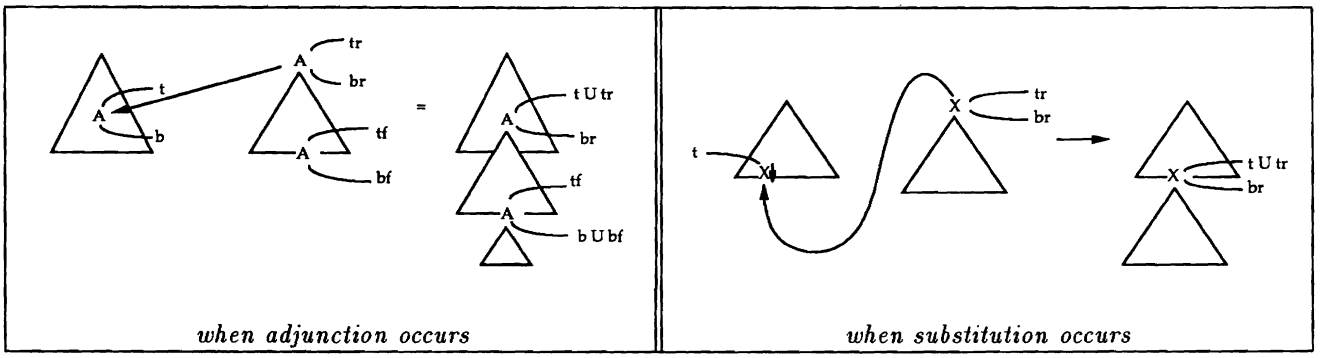


Figure 8: *Updating of feature structures*

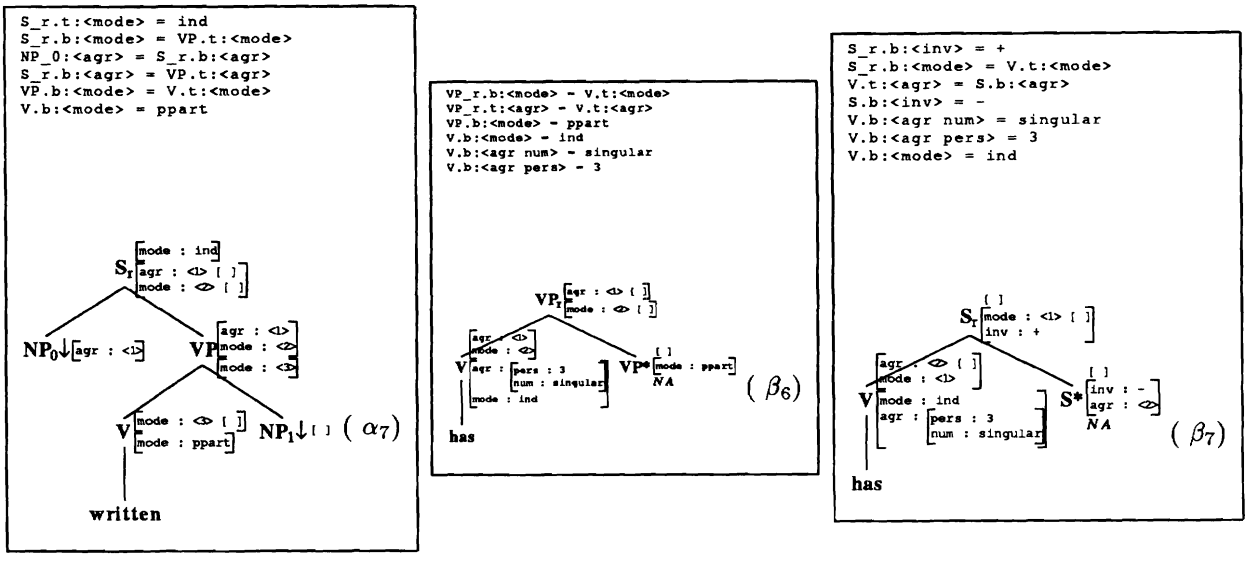


Figure 9: *Examples of unification equations*

3.5.1 Unification Equations

As in PATR-II (Shieber, 1984, 1986), we express with unification equations dependencies between DAGs[15] in an elementary tree. The extended domain of locality of TAGs allows us to state unification equations between feature structures of nodes that are not be at the same level in the tree.

The system consists of a TAG and a set of unification equations on the DAGs associated with nodes in elementary trees. An example of the use of unification equations in TAGs is given in Figure 9.[16]

The coindexing may occur between feature structures associated with different nodes in the tree. Top or bottom feature structures of a node are referred to by a node name (e.g. S_r)[17] followed by *.t* (for top) or *.b* (for bottom). The semicolon states that the following path specified in angle brackets is relative to the specified feature structure. The feature structure of a substitution node is referred to without *.t* or *.b*. For example, *VP_r.t:<agr num>* refers to the path *<agr num>* in the top feature structure associated with the adjunction node labeled by VP_r and *NP_0:<agr>* refers to the path *<agr>* of the substitution node labeled by NP_0.

The top and bottom feature structures of all nodes in the tree α_7 (Figure 9) cannot be simultaneously unified: if the top and bottom feature structures of S are unified, the *mode* will be *ind* which cannot unify with *ppart* (VP node). This forces an adjunction to be performed on S (e.g. adjunction of β_6 to derive a sentence like *Has John written a book?*) or on VP (e.g. adjunction of β_7 to derive a sentence like *John has written a book*). The sentence *John written a book* is thus not accepted.

In the tree α_7 agreement is checked across the nodes NP_0, S and VP. These equations handle the two cases of auxiliary : NP_0 *has written* NP_1 and *has* NP_0 *written* NP_1*?*. The corresponding derived trees are shown in Figure 10. α_8 derives sentences like *John has written a book.* It is obtained by adjoining β_7 on the VP node in α_7. α_9 derives sentences like *Has John written a book?.* It is obtained by adjoining β_6 on the S node in α_7. The obligatory adjunction imposed by the mode feature structure has disappeared in the derived trees α_8 and α_9. However, to be completed, α_8 and α_9 need NP-trees to be substituted in the nodes labeled by NP (e.g. *John* and *a book*).

[15]Directed Acyclic Graphs which represent the feature structures.

[16]In these examples we have merged the information stated on the trees and in the lexicon. We write unification equations above the tree to which they apply. We have also printed to the right of each node the matrix representation of the top and bottom feature structures.

[17]We implicitly require that each node have a unique name in an elementary tree. If necessary, subscripts differentiate nodes of the same category.

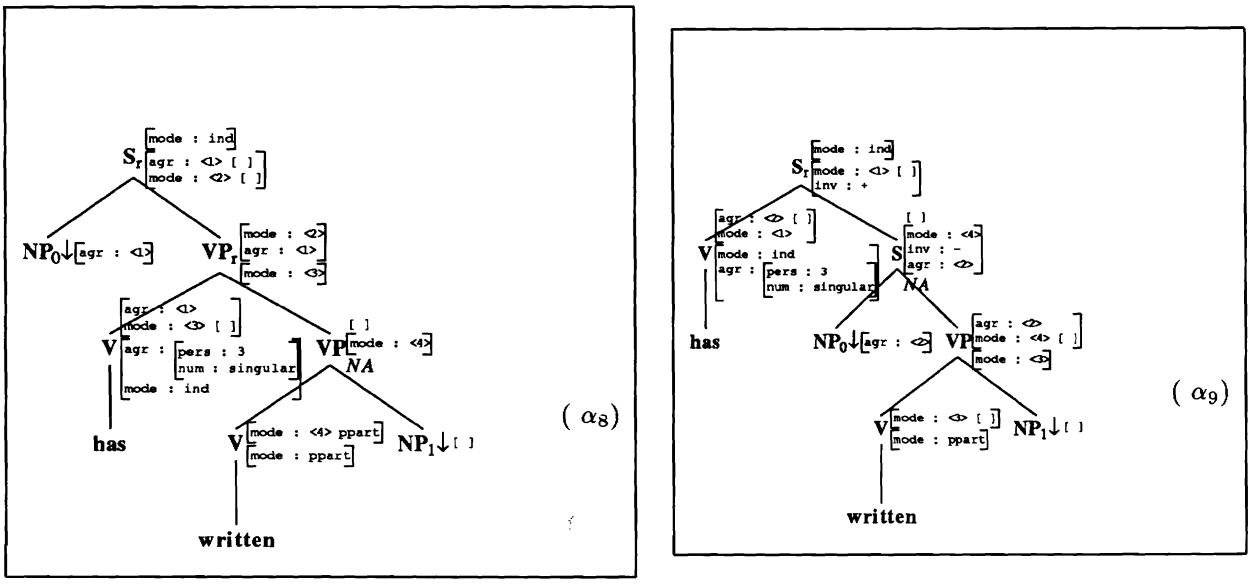


Figure 10: NP_0 *has written* NP_1 and *Has* NP_0 *written* NP_1*?*

3.5.2 Decidability of the recognition problem of Feature-based Lexicalized TAGs

As for context-free based unification based grammar, the problem of knowing whether a string is recognized by a unification-based TAG is undecidable. To make the system decidable, Vijay-Shanker and Joshi (1988) restrict the feature structures a node to be of bounded size. The resulting system is more expressive than TAGs and is still weakly equivalent to TAGs.

Lexicalization of feature-based TAGs guarantees by itself the decidability of the recognition problem. Since lexicalized grammars are finitely ambiguous, for a given sentence only a finite number of structures must be considered as possible parses. Furthermore, the Earley-type parsing algorithm can be extended to handle feature-based lexicalized TAGs in a simple way without the use of special mechanisms.

3.5.3 Extension to the Earley-type Parser

The Earley-type algorithm for TAGs (Schabes and Joshi, 1988) can be extended to parse Lexicalized TAG with unification equations on elementary trees. The extension is similar to the one proposed by Shieber (1985) for parsing the PATR-II formalism, but it does not require the use of restrictors. For the recognition of a substituted tree, we check that unification constraints are compatible at the prediction step and pass information only at the completion step. For the recognition of an adjunction, we check only that unification constraints are compatible at the Left Predictor, Left Completor and Right Predictor steps and we pass information only at the Right Completor step.

What follows is an informal explanation of the extension to the Earley-type parser. A new component D is added to the states manipulated by the Earley-type parser. D specifies a set of feature structures associated with the nodes of the tree represented by the state. The manipulation of the other components of a state remain the same. We will ignore these components of a state and focus our attention here on the manipulation of the set of feature structures D.

The Scanner, Move-dot-down and Move-dot-up processors behave as before and copy the DAG D to the new state.[18] The Left Predictor predicts all possible adjunctions and also tries to recognize the tree with no adjunction. In case no adjunction is left predicted, the Left Predictor adds the new state only if the top and bottom feature structures are compatible (see Figure 7). If they are compatible, a new state is added but top and bottom feature structures are not unified. They will be unified in the Right Predictor. Then,

[18]Identical states have identical components, identical feature structures D.

if no adjunction has been left predicted, the Right Predictor moves the dot up and unifies top and bottom feature structures (see Figure 7).

The recognition of an adjunction with feature structures is shown in Figure 7.[19] At each step of the recognition of an adjunction, the compatibility of the feature structures is checked. The information is passed only at the Right Completor step.

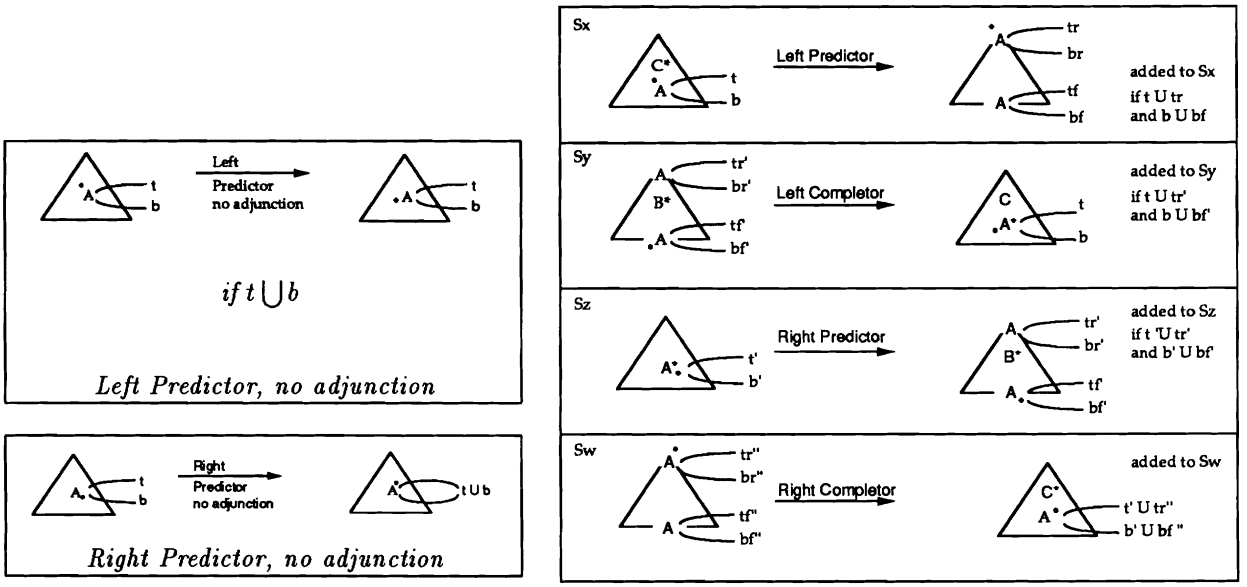


Figure 7: *No Adjunction* *Recognition of an adjunction*

For non-lexicalized TAGs, this approach does not guarantee that the algorithm halts (for similar reasons as CFG-based unification grammar). However for Lexicalized TAGs, even when recursion occurs, the termination of the algorithm is guaranteed since the recognition of a tree entails the recognition of at least one input token (its anchor) and since information is passed only when a tree is completely recognized. If information were passed before the Right Completor step (in case of adjunction), restrictors can be used to guarantee halting of the algorithm. However we believe that in practice (for the Lexicalized TAGs for French and English) passing information at an earlier step than the Right Completor step does not improve the performance.

4 Concluding Remarks

In 'lexicalized' grammars, each elementary structure is systematically associated with a lexical anchor. These structures specify extended domains of locality (as compared to the domain of locality in CFGs) over which constraints can be stated. The 'grammar' consists of a lexicon in which each lexical item is associated with a finite number of structures for which that item is the anchor.

We have seen that lexicalized grammars suggest a natural two-step parsing strategy. The first step selects the set of structures corresponding to each word in the sentence. The second step tries to combine the selected structures.

Lexicalized grammars allow us to make the search space of a pure top-down parser (as DCG-type parser for TAGs) finite and prevent looping on left-recursive rules. Experimental data show that the performance of the Earley-type parser is drastically improved. The first pass not only filters the grammar used by the parser to produce a relevant subset but also enables the parser to use non-local bottom-up information to guide its search. Current work done in collaboration with Vijay-Shanker indicates that some conflicts of move of LR-type parsers for TAGs can be solved at run-time with this strategy. We plan to study in practice to what extend the strategy improves the performance of an LR-parser for TAGs used in a pseudo-parallel way.

[19] A substituted tree is recognized in a similar way and is not explained here.

Finally, we showed that lexicalization makes the recognition problem for feature-based TAGs decidable. Then we explained how constraints over these structures expressed by unification equations can be parsed by a simple extension of the Earley-type parser.

The organization of lexicalized grammars and the simplicity and effectiveness of the two-pass strategy are therefore attractive from both a linguistic and for processing point of view.

Acknowledgments

This work is partially supported by Darpa grant N0014-85-K0018, ARO grant DAAL03-89-C-0031PRI NSF grant- IRI84-10413 A02.

We have benefited from our discussions with Anne Abeillé, Bob Frank, Lauri Karttunen, Bernard Lang, Mitch Marcus, Stuart Shieber, Mark Steedman and K. Vijayshanker. We would also like to thank Ellen Hays and Patrick Paroubek.

References

Abeillé, Anne, 1988. *A Lexicalized Tree Adjoining Grammar for French: the General Framework.* Technical Report MS-CIS-88-64, University of Pennsylvania.

Abeillé, Anne, 1988. Light Verb Constructions and Extraction out of NP in Tree Adjoining Grammar. In *Papers from the 24th Regional Meeting of the Chicago Linguistic Society.* Chicago.

Abeillé, Anne, August 1988. Parsing French with Tree Adjoining Grammar: some Linguistic Accounts. In *Proceedings of the* 12^{th} *International Conference on Computational Linguistics (COLING'88).* Budapest.

Abeillé, Anne, 1990. Lexical Constraints on Syntactic Rules in a Tree Adjoining Grammar. To appear in the proccedings of the 28^{th} Meeting of the Association for Computational Linguistics (ACL'90).

Abeillé, Anne and Schabes, Yves, 1989. Parsing Idioms in Tree Adjoining Grammars. In *Fourth Conference of the European Chapter of the Association for Computational Linguistics (EACL'89).* Manchester.

Abeillé, Anne and Schabes, Yves, 1990. Non Compositional Discontinuous Constituents in Tree Adjoining Grammar. In *Proceedings of the Symposium on Discontinuous Constituents.* Tilburg, Holland.

Abeillé, Anne; M., Bishop Kathleen; Cote, Sharon; and Schabes, Yves, forthcoming, 1990. *A Lexicalized Tree Adjoining Grammar for English.* Technical Report, Department of Computer and Information Science, University of Pennsylvania.

Chomsky, N., 1981. *Lectures on Government and Binding.* Foris, Dordrecht.

Gazdar, G.; Klein, E.; Pullum, G. K.; and Sag, I. A., 1985. *Generalized Phrase Structure Grammars.* Blackwell Publishing, Oxford. Also published by Harvard University Press, Cambridge, MA.

Gross, Maurice, 2-6 July 1984. Lexicon-Grammar and the Syntactic Analysis of French. In *Proceedings of the* 10^{th} *International Conference on Computational Linguistics (COLING'84).* Stanford.

Joshi, Aravind K., August 1969. Properties of Formal Grammars with Mixed Type of Rules and their Linguistic Relevance. In *Proceedings of the International Conference on Computational Linguistics.* Sanga Saby.

Joshi, Aravind K., 1973. A Class of Transformational Grammars. In M. Gross, M. Halle and Schutzenberger, M.P. (editors), *The Formal Analysis of Natural Languages.* Mouton, La Hague.

Joshi, Aravind K., 1985. How Much Context-Sensitivity is Necessary for Characterizing Structural Descriptions—Tree Adjoining Grammars. In Dowty, D.; Karttunen, L.; and Zwicky, A. (editors), *Natural Language Processing—Theoretical, Computational and Psychological Perspectives.* Cambridge University Press, New York. Originally presented in a Workshop on Natural Language Parsing at Ohio State University, Columbus, Ohio, May 1983.

Joshi, Aravind K., 1988. An Introduction to Tree Adjoining Grammars. In Manaster-Ramer, A. (editor), *Mathematics of Language.* John Benjamins, Amsterdam.

Joshi, A. K.; Levy, L. S.; and Takahashi, M., 1975. Tree Adjunct Grammars. *J. Comput. Syst. Sci.* 10(1).

Karttunen, Lauri, 1986. *Radical Lexicalism.* Technical Report CSLI-86-68, CSLI, Stanford University. Also in *Alternative Conceptions of Phrase Structure*, University of Chicago Press, Baltin, M. and Kroch A., Chicago, 1989.

Kroch, A. and Joshi, A. K., 1985. *Linguistic Relevance of Tree Adjoining Grammars.* Technical Report MS-CIS-85-18, Department of Computer and Information Science, University of Pennsylvania.

Lang, Bernard, 1990. Towards a Uniform Formal Framework for Parsing. In Tomita, Masaru (editor), *Current Issues in Parsing Technologies.* Kluwer Accademic Publishers.

Pollard, Carl and Sag, Ivan A., 1987. *Information-Based Syntax and Semantics. Vol 1: Fundamentals.* CSLI.

Schabes, Yves and Joshi, Aravind K., June 1988. An Earley-Type Parsing Algorithm for Tree Adjoining Grammars. In 26^{th} *Meeting of the Association for Computational Linguistics (ACL'88).* Buffalo.

Schabes, Yves and Vijay-Shanker, 1990. Deterministic Left to Right Parsing of Tree Adjoining Languages. To appear in the proceedings of the 28^{th} Meeting of the Association for Computational Linguistics (ACL'90).

Schabes, Yves; Abeillé, Anne; and Joshi, Aravind K., August 1988. Parsing Strategies with 'Lexicalized' Grammars: Application to Tree Adjoining Grammars. In *Proceedings of the* 12^{th} *International Conference on Computational Linguistics (COLING'88).* Budapest.

Shieber, Stuart M., July 1984. The Design of a Computer Language for Linguistic Information. In 22^{nd} *Meeting of the Association for Computational Linguistics (ACL'84).* Stanford.

Shieber, Stuart M., July 1985. Using Restriction to Extend Parsing Algorithms for Complex-feature-based Formalisms. In 23^{rd} *Meeting of the Association for Computational Linguistics (ACL'85).* Chicago.

Shieber, Stuart M., 1986. *An Introduction to Unification-Based Approaches to Grammar.* Center for the Study of Language and Information, Stanford, CA.

Steedman, M., 1987. Combinatory Grammars and Parasitic Gaps. *Natural Language and Linguistic Theory* 5:403–439.

Vijay-Shanker, K., 1987. *A Study of Tree Adjoining Grammars.* PhD thesis, Department of Computer and Information Science, University of Pennsylvania.

Vijay-Shanker, K. and Joshi, A. K., 1985. Some Computational Properties of Tree Adjoining Grammars. In 23^{rd} *Meeting of the Association for Computational Linguistics*, pages 82–93.

Vijay-Shanker, K. and Joshi, A.K., August 1988. Feature Structure Based Tree Adjoining Grammars. In *Proceedings of the* 12^{th} *International Conference on Computational Linguistics (COLING'88).* Budapest.