Lexicalized TAGs, Parsing and Lexicons*

Anne Abeillé, Kathleen Bishop, Sharon Cote, Aravind K. Joshi, and Yves Schabes

Department of Computer and Information Science
University of Pennsylvania, Philadelphia PA 19104-6389 USA
abeille/bishop/cote/joshi/schabes@linc.cis.upenn.edu

Abstract

In our approach, each elementary structure is systematically associated with a lexical head. These structures specify extended domains of locality (as compared to a context-free grammar) over which constraints can be stated. These constraints either hold within the elementary structure itself or specify what other structures can be composed with a given elementary structure. The 'grammar' consists of a lexicon where each lexical item is associated with a finite number of structures for which that item is the head. There are no separate grammar rules. There are, of course, 'rules' which tell us how these structures are composed. A grammar of this form will be said to be 'lexicalized'. A 'lexicalized' grammar naturally follows from the extended domain of locality of TAGs.

A general parsing strategy for 'lexicalized' grammars is discussed. 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. An Earley-type parser for TAGs has been has been developed. It can be adapted to take advantage of the two steps parsing strategy. The system parses unification formalisms that have a CFG skeleton and that have a TAG skeleton.

Along with the development of an Earley-type parser for TAGs, lexicons for English are under development. A lexicons for French is also being developed. Subsets of these lexicons are being incrementally interfaced to the parser.

We finally show how idioms are represented in lexicalized TAGs. We assign them regular syntactic structures while representing them semantically as one entry. We finally show how they can be parsed by a parsing strategy as mentioned above.

1 Lexicalized Tree Adjoining Grammar

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 both in amount and complexity: 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 1988), Karttunen's version of Categorial Grammar (Karttunen 1986, 1988), some versions of GB theory (Chomsky 1981), and Lexicon-Grammars (Gross 1984).

We say that a grammar is 'lexicalized' if it consists of: [1]

- a finite set of structures associated with each lexical item, which is intended to be the head of these structures;
- an operation or operations for composing the structures. The finite set of structures define the domain of locality over which constraints are specified, and these are local with respect to their lexical heads.

Context free grammars cannot be in general be lexicalized. However TAGs are 'naturally' lexicalized because they use an extended domain of locality (Schabes, Abeillé and Joshi, 1988). TAGs were first introduced by Joshi, Levy and Takahashi (1975), Joshi (1983-1985) and Kroch and Joshi (1985). It is known

*This work is partially supported by the DARPA grant N0014-85-K0018

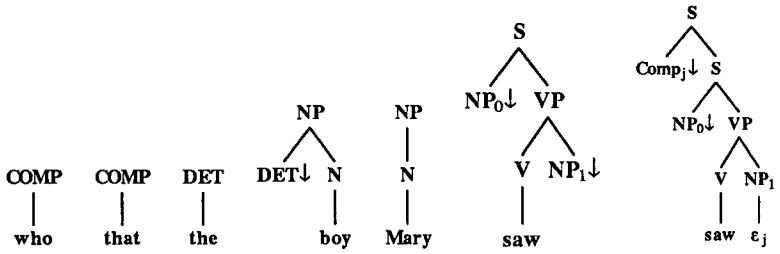
[1] By 'lexicalization' we mean that in each structure there is a lexical item that is realized. We do not mean just adding features (such as head) and unification equations to the rules of the formalism.

that Tree Adjoining Languages (TALs) are mildly context sensitive. TALs properly contain context-free languages.[2]

A basic component of a TAG is a finite set of elementary trees, each of which defines domain of locality, and can be viewed as a minimal linguistic structure. The elementary structures are projections of lexical items which serve as heads. We recall that tree structures in TAGs correspond to linguistically minimal but complete structures: the complete argument structure in the case of a predicate, the maximal projection of a category in the case of an argument or an adjunct. If a structure has only one terminal, the terminal is the head of the structure; if there are several terminals, the choice of the head for a given structure is linguistically determined, e.g. by the principles of $\overline{X}$ theory if the structure is of $\overline{X}$ type. The head of NP is N, that of AP is A. S also has to be considered as the projection of a lexical head, usually V. As is obvious, the head must always be lexically present in all of the structures it produces.

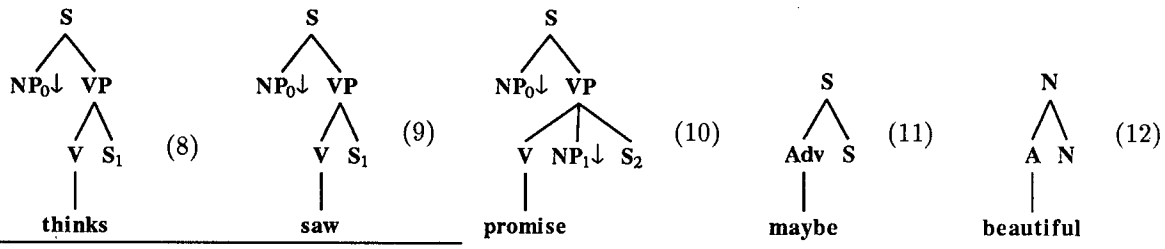
In the TAG lexicon each item is associated with a structure (or a set of structures), and that structure can be regarded as its category, linguistically speaking. Each lexical item has as many entries in the lexicon as it has possible category or argument structures. We will now give some examples of structures that appear in this lexicon.

Some examples of initial trees are (for simplicity, we have omitted the constraints associated with the nodes) :[3]



For $X \neq S$, X-type initial trees correspond to the maximal projection of the category X of the head. They are reduced to a pre-terminal node in the case of simple categories such as $COMP$ or DET (trees 1, 2 and 3) and are expanded into more complex structures in the case of categories taking arguments (tree 4). They correspond to the maximal projection of a category in the case of simple phrases and to trees which will be systematically substituted for one of the argument positions of one of the elementary structures. Trees 6-7 are examples of S-type initial trees: they are usually considered as projections of a verb and usually take nominal complements. The NP-type tree 'Mary' (tree 5), and the NP-type tree 'John' (similar to tree 5), for example, will be inserted by substitution in the tree 6 corresponding to 'NP_0 saw NP_1' to produce 'John saw Mary'.

Examples of auxiliary trees (they are predicates taking sentential complements, 8-10, or modifiers, 11-12):



[2] 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.

[3] We put indices on some non-terminals to express syntactic roles (0 for subject, 1 for first object, etc.). 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. We use the convention of marking substitution nodes by a down arrow ($\downarrow$).

2 Parsing Lexicalized TAGs

We assume that the input sentence is not infinite and that it cannot be syntactically infinitely ambiguous. 'Lexicalization' simplifies the task of a parser in the following sense. The first pass of the parser filters the grammar to a grammar corresponding to the input string. It also puts constraints on the way that adjunctions or substitutions can be performed since each structure has a head whose position in the input string is recorded. The 'grammar' of the parser is reduced to a subset of the entire grammar. Furthermore, since each rule can be used only once, recursion does not lead to the usual non-termination problem. Once a structure has been chosen for a given token, the other possible structures for the same token do not participate in the parse. Of course, if the sentence is ambiguous, there may be more than one choice.

If one adopts an off-line parsing algorithm, the parsing problem is reduced to the following two steps:

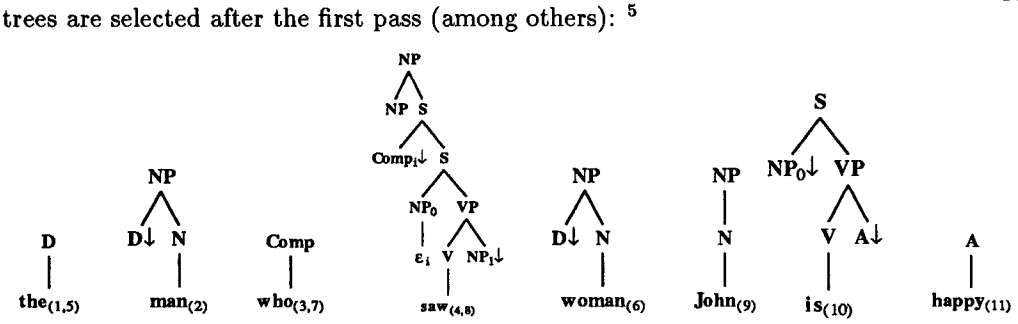
- In the first step the parser will select the set of structures corresponding to each word in the sentence. Each structure can be considered as an encoding of a set of 'rules'.
- Then the parser tries to determine whether these structures can be combined to obtain a well-formed structure. In particular, it puts the structures corresponding to arguments into the structures corresponding to predicates, and adjoins, if needed, the auxiliary structures corresponding to adjuncts to what they select (or are selected) for.

In principle, any parsing strategy can be applied in the second step, since the number of structures produced is finite, and since each of them corresponds to a token in the input string, the search space is finite and termination is guaranteed. In principle, one can proceed inside out, left to right or in any other way. Of course, standard parsing algorithms can be used, too. In particular, we can use the top-down parsing strategy without encountering the usual problems due to recursion.

By assuming that the number of structures associated with a lexical item is finite, since each structure has a lexical item attached to it, we implicitly make the assumption that an input string of finite length cannot be syntactically infinitely ambiguous.

An Earley-type parser for TAGs has been investigated by Schabes and Joshi (1988). The algorithm has a linear best time behavior and an $O(n^9)$ worst time behavior. This is the first practical parser for TAGs because as is well known for CFGs, the average behavior of Earley-type parsers is superior to its worst time behavior. We extended it to deal with substitution and feature structures for TAGs. By doing this, we have built a system that parses unification formalisms that have a CFG skeleton and also those that have a TAG skeleton.

The Earley-type parser for TAGs can be extended to take advantage of lexicalized TAGs. Once the first pass has been performed, a subset of the grammar is selected. The structures encode the value and positions of their head. Structures of same head value are merged together, and the list of head positions recorded.[4] This enables us to use the head position information while processing efficiently the structures. For example, given the sentence `The` $_1$ `man` $_2$ `who` $_3$ `saw` $_4$ `the` $_5$ `woman` $_6$ `who` $_7$ `saw` $_8$ `John` $_9$ `is` $_{10}$ `happy` $_{11}$, the following trees are selected after the first pass (among others): [5]



[4] Unlike our previous suggestions (Schabes, Abeillé and Joshi, 1988), we do not distinguish each structure by its head position since it increases unnecessarily the number of states of the Earley parser. By factoring recursion, the Earley parser enables us to process only once parts of a tree that are associated with several lexical items of same value but different positions. However, if termination is required for a pure top down parser, it is necessary to distinguish each structure by its head position.

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

Notice that there is only one tree for the relative clauses introduced by **saw** but that its head position can be 4 or 8. Similarly for **who** and **the**.

The head positions of each structure imposes constraints on the way that the structures can be combined (the head 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 **man** will not be predicted for substitution in any of the trees corresponding to **saw** since the head positions would not be in the right order.

3 Lexicalized TAG for English

A lexicalized TAG for English is under development (Bishop, Cote and Abeillé, 1988). Trees are gathered in tree families when an element of a certain type (e.g. a verb) is associated with more than one tree. We have 55 such tree families that correspond to most of the basic argument structures. There are 10 tree families for simple verb sentences, 17 families for sentences with verbs taking sentential complements, 11 families for light verb-noun constructions, 7 families for verb-particle combinations and 10 families for light verb-adjective constructions. A tree family consists on average of 12 trees, which makes approximately 700 trees total.

The grammar covers subcategorization (strictly lexicalized), wh-movement and unbounded dependencies, light verb construction, some idioms, transitivity alternations (such as dative shift or the so-called ergative alternation), subjacency and some island violations.

The current size of the lexicon is approximately 1200 words: 750 verbs; 350 nouns; 50 adjectives; 25 prepositions, adverbs and determiners.

Subsets are being extracted and interfaced to the parser. Each subset is being incrementally augmented as it is debugged. A similar lexicalized TAG for French is also under development.

4 Parsing Idioms in Lexicalized TAGs

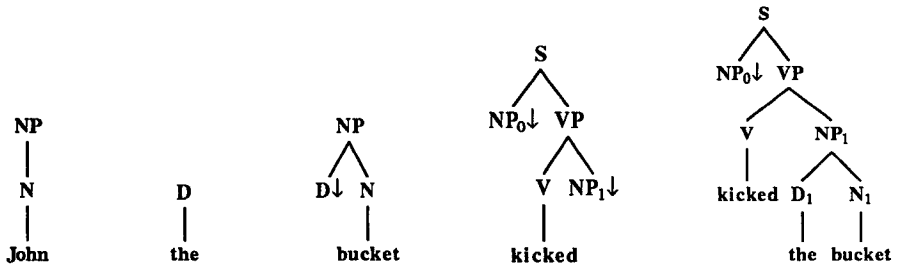
In lexicalized TAGs, idioms fall into the same grammar as 'free' sentences (Abeillé and Schabes, 1989). We assign them regular syntactic structures while representing them semantically as one entry. Transformations and modifiers thus can apply to them. Unlike previous approaches, their variability becomes the general case and their being totally frozen the exception.

Idioms are represented by extended elementary trees with multicomponent head. When an idiomatic tree is selected by its head, lexical items are attached to some nodes in the tree. Idiomatic trees are selected by a single head node; however the head value imposes lexical values of other nodes in the tree. This operation of attaching the head item of an idiom and its lexical parts is called **lexical attachment**. The resulting tree has the lexical items corresponding to the pieces of the idiom already attached to it.

The parser must be able to conjecture idiomatic and literal interpretation of an idiom. We propose to parse idioms in two steps which are merged in the two steps parsing strategy that we use. The first step performed during the lexical pass selects trees corresponding to the literal and idiomatic interpretation. However it is not always the case that the idiomatic trees are selected as possible candidates. We require that all basic pieces building the minimal idiomatic expression must be present in the input string (in the right order). This condition is a necessary condition for the idiomatic reading but of course it is not sufficient. The second step performs the syntactic analysis as in the usual case. During the second step, idiomatic reading might be rejected. Idioms are thus parsed as any 'free' sentences. Except during the selection process, idioms do not require any special parsing mechanism.

Furthermore, our representation allows us to recognize discontinuities in idioms that come from internal structures and from the insertion of modifiers.

Take as example the sentential idiom NP_0 **kicked the bucket**. We have, among others, the following



(αNPn[John]) (αD[the]) (αNPdn[bucket]) (αtn1[kicked]) (αtdn1[kicked-the-bucket])

Figure 1: Trees selected for the input: `John kicked the bucket`

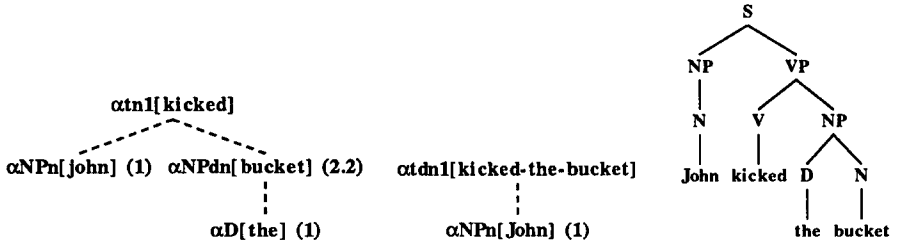
entries in the lexicon:[6]

kicked , V	: Tn1	(simple transitive verb)	(a)
kicked , V	: Tdn1[D_1 = the, N_1 = bucket]	(idiom: kicked the bucket)	(b)
the , D	: αD	(one node tree rooted by D)	(c)
bucket , N	: αNPdn	(NP tree expecting a determiner)	(d)
John , N	: αNP	(NP tree for proper nouns)	(e)

Suppose that the input sentence is `John kicked the bucket`. In the first pass, the trees in Figure 1 are be selected (among others).

The first entry for kicked (a) specifies that kicked can be attached under the V node in the tree αtn1 (See the tree αtn1[kicked] in Figure 1). However the second entry for kicked (b) specifies that `kicked` can be attached under the V node and that `the` must be attached under the node labeled by D_1 and that `bucket` must be attached under the node labeled N_1 in the tree αtn1 (See the tree αtdn1[kicked-the-bucket] in Figure 1).

The sentence can be parsed in two different ways. One derivation is built with the trees: αtn1[kicked] (transitive verb), αNPn[John], αD[the] and αNPn[bucket] . It corresponds to the literal interpretation; the other derivation is built with the trees: αtdn1[kicked-the-bucket] (idiomatic tree) and αNPn[John] (John). However, both derivations have the same derived tree:



5 Conclusion

We have presented a general parsing strategy based on 'lexicalized' grammars. TAGs are shown to be naturally 'lexicalized'. Lexicalization of a grammar suggests a two-step parsing strategy. The first step selects the set of structures corresponding to each word in the sentence. The second step puts the argument structures into predicate structures. In the first step, structures, rather than non-terminals, are associated with lexical items. The Earley-type parser for TAGs has been extended to take advantage of this strategy.

[6] The lexical entries are somewhat simplified for illustrating how idioms are handled.

We have briefly described the current state of the implementation and the size of the associated lexicon. Finally we show that in lexicalized TAGs idioms can be processed without defining special rules for processing them. We can access simultaneously frozen elements at different levels of depths in contrast to a CFG which either has to flatten the idiomatic structure (and loose the possibility of regular insertion of modifiers) or to use specific devices to check the presence of an idiom. The two pass parsing strategy we use, combined with the operation of direct attachment of lexical items in idiomatic trees, enable us to cut down the number of idiomatic trees that the parser takes as possible candidates. We easily get the possibly idiomatic and literal reading for a given sentence. The only distinctive property of idioms is the non compositional semantics of their frozen constituents.

References

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

Bishop, Kathleen M.; Cote, Sharon; and Abeillé, Anne, 1988. *A Lexicalized Tree Adjoining Grammar for English: some Basic Accounts.* 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, 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. To also appear in *New Approaches to Phrase Structures*, University of Chicago Press, Baltin, M. and Kroch A., Chicago, 1988.

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.

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.* Buffalo.

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.*

Steedman, M., 1988. Combinatory Grammars and Parasitic Gaps. To appear in *Natural Language and Linguistic Theory* .