

Lexicalized Grammar 101

Matthew Stone
Department of Computer Science and Center for Cognitive Science
Rutgers University
Piscataway NJ 08854-8019

March 29, 2002

Paper ID:

Keywords: Syntax, lexicalized grammar

Contact Author: Matthew Stone

Under consideration for other conferences (specify)? No

Abstract

I propose a tree-rewriting grammar formalism, TAGLET, defined by the usual complementation operation and the simplest imaginable modification operation. TAGLET is context free and permits lexicalization of treebank parses (though TAGLET is only weakly equivalent in generative power to general CFGs), as well as straightforward exploration of rich hierarchical syntactic structures and detailed feature structures in the spirit of current linguistic syntax. It admits a dynamic-programming parser, with the $N \times N \times N$ chart search characteristic of CFGs and a clean “strong competence” implementation of combinatory operations. Derivations are lexicalized dependency structures: this invites the assignment of probabilities to structures based on bigram dependencies, and at the same time introduces the natural search space for natural language generation. In short, the formalism offers a simple and effective scaffold to bring the issues of current research into classroom learning.

Lexicalized Grammar 101

Paper ID:

Abstract

I propose a tree-rewriting grammar formalism, TAGLET, defined by the usual complementation operation and the simplest imaginable modification operation. TAGLET is context free and permits lexicalization of treebank parses (though TAGLET is only weakly equivalent in generative power to general CFGs), as well as straightforward exploration of rich hierarchical syntactic structures and detailed feature structures in the spirit of current linguistic syntax. It admits a dynamic-programming parser, with the $N \times N \times N$ chart search characteristic of CFGs and a clean “strong competence” implementation of combinatory operations. Derivations are lexicalized dependency structures: this invites the assignment of probabilities to structures based on bigram dependencies, and at the same time introduces the natural search space for natural language generation. In short, the formalism offers a simple and effective scaffold to bring the issues of current research into classroom learning.

1 Introduction

For most language researchers, grammar formalisms offer a means to some broader objective: perhaps the explanation of crosslinguistic universals; perhaps the characterization of human psychological processes; perhaps the construction of useful computational systems. Grammar formalisms come and go, while the objectives endure as a compelling draw for research in language.

Lexicalization is a philosophy that ties grammar formalisms particularly closely to these uses. In lexicalized approaches to grammar, syntactic and semantic specifications are associated directly with words, and grammatical operations are tightly integrated with these lexical representations. For linguistics, this philosophy invites a fine-grained description of sentence syntax, in which researchers document the diversity of linguistic constructions within and across languages, and at the same time uncover important generalizations among them. For computation, this philosophy suggests a particularly concrete approach to language processing, in which the information a system maintains and the decisions it takes ultimately always just concern

words. The fruits of this approach include simple representations and algorithms, as in (Sleator and Temperley, 1993); effective models of language, as in (Collins, 1997; Charniak, 1997); and powerful leverage on complex linguistic tasks, as in (Stone et al., 2001). Both linguistic and computational advantages combine in lexicalized approaches to the cognitive science of human language use; see, e.g., (Tanenhaus and Trueswell, 1995).

Important as they are, lexicalized grammars can be forbidding. Tree-adjoining grammars (TAG) (Joshi et al., 1975; Schabes, 1990) and combinatory categorial grammars (CCG) (Steedman, 2000) require complex bookkeeping for effective computation: when I wrote a CCG parser as an undergraduate, I found it ballooning into a semester course project; I still have never written a TAG parser or a CCG generator. Other formalisms come with linguistic assumptions that are hard to manage. Link grammar (Sleator and Temperley, 1993) and other pure dependency formalisms eschew traditional hierarchical structure altogether, while HPSG (Pollard and Sag, 1994) comes with a commitment to its complex, rather bewildering regime for formalizing linguistic information as feature structures.

In this paper, I sketch the mathematical, linguistic and computational aspects of a simple alternative that seems particularly suited to the classroom. It is a tree-rewriting grammar formalism like TAG, so I call it TAGLET.¹ TAGLET shares TAG’s substitution operation for complementation, but in place of general adjunction for modification uses the sister-adjunction operation defined in (Rambow et al., 1995); sister-adjunction just adds the modifier subtree as a child of an existing node in the head tree. I describe TAGLET formally in Section 2 and by example in Section 3. (Space precludes more background, motivation, or contrast with other lexicalized formalisms for computational syntax.)

TAGLET’s simplicity pays off in many ways:

¹If the acronym must stand for something, “Tree Assembly Grammar for LEXicalized Teaching” will do.

- TAGLET nodes can be easily decorated with grammatical feature structures (without the hassle of TAG top and bottom features or CCG functor-argument contravariance, yet with broad enough locality that features need not snowball as in HPSG); see Section 3.
- TAGLET languages are context free (though TAGLET is only weakly equivalent in generative power to general CFGs); see Section 4.
- TAGLET admits a dynamic-programming parser, with the $N \times N \times N$ chart search characteristic of CFGs and a clean “strong competence” implementation of combinatory operations; see Section 5.

TAGLET still retains the benefits of lexicalization. TAGLET permits lexicalization of treebank parses, as well as straightforward exploration of rich hierarchical syntactic structures in the spirit of current linguistic syntax; see Section 6. TAGLET derivations are lexicalized dependency structures: this invites the assignment of probabilities to structures based on bigram dependencies, and at the same time introduces the natural search space for natural language generation; see Section 7.

2 Definitions

I define TAGLET in terms of *primitive trees*. The definitions require a set V_T of *terminal categories*, corresponding to our lexical items, and a disjoint set V_N of *nonterminal categories*, corresponding to constituent categories. TAGLET uses trees labeled by these categories both as representations of the syntactic structure of sentences and as representations of the grammatical properties of words:

- A *syntactic tree* is a tree whose nodes are each assigned a unique label in $V_N \cup V_T$, such that only leaf nodes are assigned a label in V_T .
- A *lexical tree* is a syntactic tree in which exactly one node, called the *anchor*, is assigned a label in V_T . The path through such a tree from the root to the anchor is called the *spine*.

A *primitive tree* is lexical tree in which every leaf is the child of a node on the spine. That is, in a primitive tree, each leaf’s parent dominates the anchor

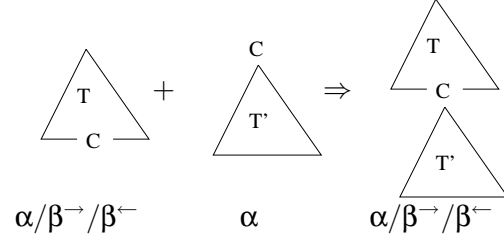


Figure 1: General schema of substitution (complementation).

(and equivalently again, each leaf c -commands the anchor). The restriction greatly simplifies parsing, at the cost of flexible treatment of idioms. Figures 3 and 4 illustrate primitive trees for individual lexical items with compositional semantics.

A TAGLET *element* is a pair $\langle T, O \rangle$ consisting of primitive tree together with the specification of the operation for the tree; the allowable operations are complementation, indicated by α ; premodification at a specified category $C \in V_N$, indicated by $\beta^{\rightarrow}(C)$ and postmodification at a specified category $C \in V_N$, indicated by $\beta^{\leftarrow}(C)$.

Formally, then, a TAGLET *grammar* is a tuple $G = \langle V_T, V_N, \Gamma \rangle$ where V_T gives the set of terminal categories, V_N gives the set of nonterminal categories, and Γ gives a set of TAGLET elements for V_T and V_N . Given a TAGLET grammar G , the set of *derived trees* for G is defined as the smallest set closed under the following operations:

- (Initial) Suppose $\langle T, O \rangle \in \Gamma$. Then $\langle T, O \rangle$ is a derived tree for G .
- (Substitution) Suppose $\langle T, O \rangle$ is a derived tree for G where T contains leaf node n with label $C \in V_N$; and suppose $\langle T', \alpha \rangle$ is a derived tree for G where the root of T' also has label C . Then $\langle T'', O \rangle$ is a derived tree for G where T'' is obtained from T by identifying node n with the root of T' . See Figure 1.
- (Premodification) Suppose $\langle T, O \rangle$ is a derived tree for G where T contains node n with label $C \in V_N$, and suppose $\langle T', \beta^{\rightarrow}(C) \rangle$ is a derived tree for G . Then $\langle T'', O \rangle$ is a derived tree for G where T'' is obtained from T by adding T' as the first child of node n . See Figure 2.
- (Postmodification) Suppose $\langle T, O \rangle$ is a derived

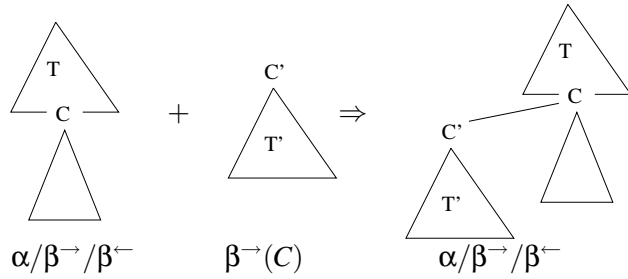


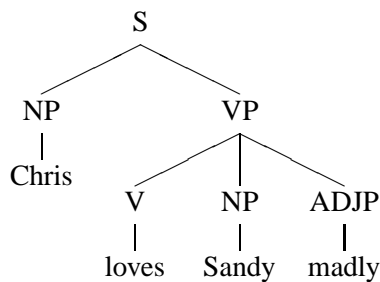
Figure 2: General schema of forward sister-adjunction (premodification.)

tree for G where T contains node n with label $C \in V_N$, and suppose $\langle T', \beta^{\leftarrow}(C) \rangle$ is a derived tree for G . Then $\langle T'', O \rangle$ is a derived tree for G where T'' is obtained from T by adding T' as the last child of node n . (The picture would be the mirror-image of Figure 2.)

A *derivation* for G is a derived tree $\langle T, \alpha \rangle$ for G , in which all the leaves of T are elements of V_T . The *yield* of a derivation $\langle T, \alpha \rangle$ is the string consisting of the leaves of T in order. A string σ is in the *language* generated by G just in case σ is the yield of some derivation for G .

3 Examples

With TAGLET, two kinds of examples are instructive: those where TAGLET can mirror TAG, and those where it cannot. For the first case, consider an analysis of *Chris loves Sandy madly* by the trees of Figure 3. The final structure is:



For the second case, consider the embedded question *who Chris thinks Sandy likes*. The usual TAG analysis uses the full power of adjunction. TAGLET requires the use of one of the familiar context-free filler-gap analyses, as perhaps that suggested by the trees in Figure 4, and their composition:

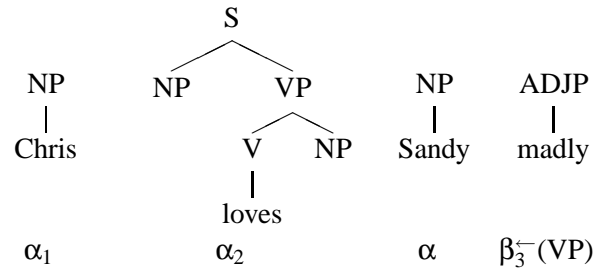


Figure 3: Parallel analysis in TAGLET and TAG.

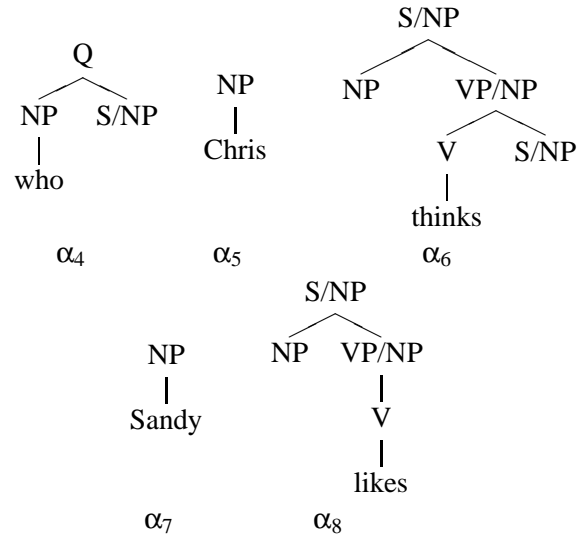
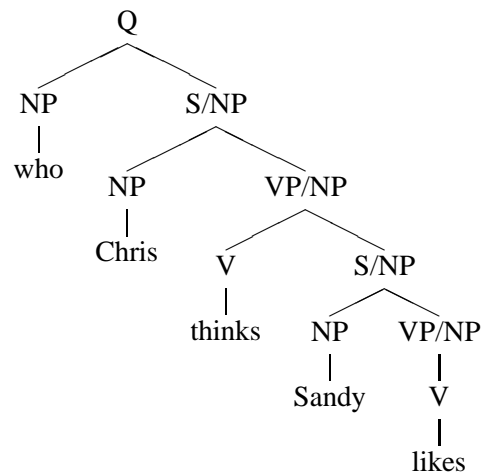
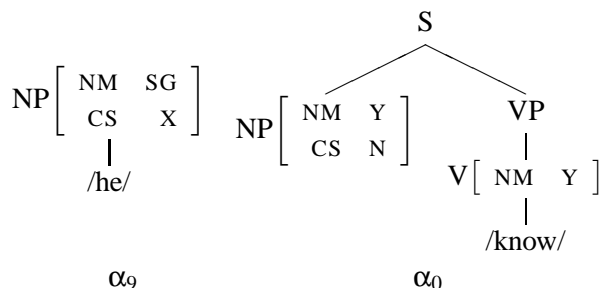


Figure 4: TAGLET requires a gap-threading analysis of extraction (or another context-free analysis).

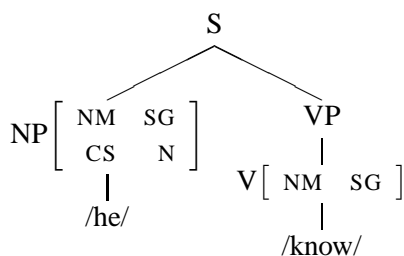


The use of syntactic features amounts to an intermediate case. In TAGLET derivations (unlike in TAG) nodes accrete children during the course of a derivation but are never rewritten or split. Thus, we can decorate any TAGLET node with a single set of

syntactic features that is preserved throughout the derivation. Consider the trees for *he knows* below:



When these trees combine, we can immediately unify the number Y of the verb with the pronoun's singular; we can immediately unify the case X of the pronoun with the nominative assigned by the verb:

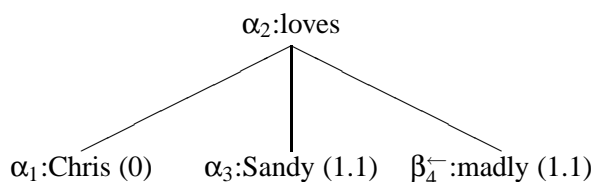


The feature values will be preserved by further steps of derivation.

4 Properties

4.1 Derivation Trees

Each node in a TAGLET derived tree T is first contributed by a specific TAGLET element, and so indirectly by a particular anchor. Accordingly, we can construct a lexicalized *derivation tree* corresponding to T . Nodes in the derivation tree are labeled by the elements used in deriving T . An edge leads from parent E to child E' if T includes a step of derivation in which E' is substituted or sister-adjoined at a node first contributed by E . To make the derivation unambiguous, we record the address of the node in E at which the operation applies, and we order the edges in the derivation tree in the same order that the corresponding operations are applied in T . For Figure 3, we have:

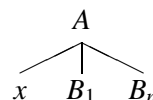


4.2 Any context-free language (CFL) has a TAGLET grammar.

Let L be a CFL. Then there is a grammar G for L in Greibach normal form (Hopcroft and Ullman, 1979), where each production has the form

$$A \rightarrow xB_1 \dots B_n$$

where $x \in V_T$ and $B_i \in V_N$. For each such production, create the TAGLET element which allows complementation with a tree as below:



An easy induction transforms any derivation in G to a derivation in this TAGLET grammar, and vice versa. So both generate the same language L .

4.3 Any TAGLET grammar generates a CFL.

Relabel all the internal nodes (except the root of α trees) of each TAGLET tree so that no two share the same internal nodes.

To handle sister-adjunction, create a new symbol N^+ and N^- for each internal node N in a tree. Add productions $N^+ \rightarrow \epsilon$ and $N^+ \rightarrow R_i$ where R_i is the root node of a premod element that could have sister-adjoined to the node corresponding to N in the original grammar. Add productions $N^- \rightarrow \epsilon$ and $N^- \rightarrow R_i$ where R_i is the root node of a postmod element that could have sister-adjoined to the node corresponding to N in the original grammar. Now for each internal node N in a TAGLET element with children $N_1 N_l \dots h N_{l+1} \dots N_k$ add the context-free production

$$N \rightarrow N^+ N_1 N^+ \dots N^+ N_l N^+ h N^- N_{l+1} N^- \dots N^- N_k N^-$$

Call this grammar G . An easy induction transforms any complete derivation in G to a complete TAGLET derivation, and vice versa. So both generate the same language.

4.4 There are CFGs without any strongly equivalent TAGLET.

In any TAGLET derivation, there is a bound on the depth of the closest lexical leaf to the root. This is because the initial anchor of the derivation has the

same depth as it has in its elementary tree, which is just some finite depth given by the grammar. In general, however, CFGs need not share this property. Such CFGs have no strongly equivalent TAGLET.

5 Parsing

The close analogy between TAGLET parsing and CFG parsing can be motivated through consideration of the TAGLET operations involved in assembling a derived tree. Suppose we make a bottom-up traversal of a TAGLET derivation tree. After we process any node (and all its children), we obtain a subtree of the final derived tree. This subtree represents a complete constituent that must appear in order in the final yield. Our parsing algorithms reproduce this hierarchical discovery of constituents; we run the assembly off a particular string. The only trick is to manage the insertion of complements and modifiers into the tree for the head while its tree is incomplete. To do that, we apply operations along what is known as the *frontier* of the tree. To avoid spurious ambiguities, we also require that operations to the left frontier must precede operations to the right frontier.

A node n in a TAGLET derived tree is on the *right frontier* if no node that follows in postorder traversal dominates a lexical item. It's on the *growing right frontier* if it's on the right frontier, it is a child of the spine, and no leaf nonterminal precedes it in postorder traversal. The *left frontier* and *growing left frontier* are defined *mutatis mutandis*; a node n is on the *frontier* if it is on either.

A TAGLET derived tree is an *open constituent* if every leaf nonterminal is on the frontier. It's a *completed constituent* if it has no leaf nonterminals.

Formally, then, the basic operations of TAGLET parsing are incorporating a completed constituent T_2 into an open constituent T_1 on the right along its growing right frontier, yielding T_3 ; and incorporating a completed constituent T_1 into an open constituent T_2 on the left along its growing left frontier, yielding T_3 . This gives a relation $\text{COMBINE}(T_1, T_2, T_3)$. Note that these operations are defined on ordinary TAGLET structures and are special cases of general TAGLET operations of complementation and sister-adjunction. The frontier restrictions just ensure the operations result in

constituents. Thus an implementation of these operations can respect a “strong competence” hypothesis, in which the knowledge and structures of the grammar are used directly for linguistic processes.

This leads to a CKY-style tabled parsing algorithm for TAGLETS. The parser analyses a string of length N using a dynamic-programming procedure to enumerate all the analyses that span contiguous substrings, shortest substrings first. We write $T \in (i, j)$ to indicate that object T spans position i to j . So we have:

```

for word  $w \in (i, i + 1)$ ,  $T$  with anchor  $w$ 
  add  $T \in (i, i + 1)$ 

for  $k \leftarrow 3..N$ 
  for  $i \leftarrow k - 2..1$ 
    for  $j \leftarrow i + 1..k - 1$ 
      for  $T_1 \in (i, j)$  and  $T_2 \in (j, k)$ 
        for  $T_3$  with  $\text{COMBINE}(T_1, T_2, T_3)$ 
          add  $T_3 \in (i, k)$ 

```

The point of this algorithm is simplicity and familiarity; it offers the same $N \times N \times N$ chart search that any CKY CFG parser has. Of course, any parser that delivers possible analyses exhaustively will be prohibitively expensive in the worst-case; analyses of ambiguities multiply exponentially. At the cost of a strong-competence implementation, one can imagine avoiding the complexity by maintaining TAGLET derivation forests. At that point, to establish $O(N^3)$ parsing, we would also need to show that the number of alternatives combinations of T_1 and T_2 does not depend on N . The reason for this is that TAGLET parsing operations apply within spans of the spine of single elementary trees. These spans just depend on how many arguments have been saturated, and in constituents, arguments are of course saturated in order, one after the other. Accordingly, a finite inventory of spans that define possible TAGLET operations for both simple and derived structures, can be identified ahead of time from the elementary trees of the TAGLET, independent of the string being parsed.

6 Modeling

As with a CFG, you can read a TAGLET straightforwardly off of an annotated parse tree. For TAGLET,

each internal node in the parse tree has to have exactly one child annotated as a head, and the rest annotated as complements or modifiers.

The following algorithm reconstructs the set of TAGLET elements required for a parse recursively. It starts at the root of a subtree of the parse and is given the operation with which this subtree is combined. It reconstructs the TAGLET element that contributes the root of this subtree, as follows. The head-path from the root to a lexical item determines the spine of the element. The complement children of this spine determine the other nodes in the element. Other children of the spine are modifiers which sister-adjoin to the main tree.

Now comes the recursive part of the algorithm. Within the element we have found, all children of the spine other than the head node have a full parse tree associated with them. We just treat each of these subtrees in turn.

We can apply this algorithm to a *set* of parse trees too. This gives a procedure to derive a TAGLET grammar from a treebank of parsed sentences. We can also tabulate the TAGLET operations that are used in the treebank, for example in terms of the bigram dependencies encoded in derivation trees. Such a table provides the basis for scoring TAGLET parses by probabilities of their lexicalized dependencies, as estimated from a treebank.

7 Generation

TAGLET retains from TAG the two most important features of grammar from the point of view of generation. First, because the grammar contains lexicalized elementary structures, derivations in TAGLET outline a space of consistent, meaningful choices. Searching through grammatical derivations does not require the system to manage interrelated decisions, or to make syntactic or lexical commitments without specific motivation from semantic and pragmatic considerations. Thus, there is no reason for the generator to work with abstract levels of syntactic structure—it can again adopt a “strong competence” implementation in which grammatical knowledge is used directly.

Second, because complementation and modification are independent, there is no tension between providing required material in a derivation, and

elaborating a derivation with supplementary information: a generator can provide required material first, then elaborate it. This is essential for using the grammar in high-level tasks such as the planning of referring expressions or the “aggregation” of related semantic material into a single complex sentence.

The semantic interpretation and search strategy of SPUD can be naturally implemented using TAGLET syntax rather than full TAG. Doing so makes it simpler to build reversible resources and architectures for dialogue applications (since TAGLET has such simple parsing) and to acquire resources for generation (since TAGLET has such a direct relationship to treebank parses).

References

- Eugene Charniak. 1997. Statistical parsing with a context-free grammar and word statistics. In *AAAI*.
- Michael Collins. 1997. Three generative, lexicalized models for statistical parsing. In *ACL*, pages 16–23.
- John E. Hopcroft and Jeffrey D. Ullman. 1979. *Introduction to automata theory, languages and computation*. Addison-Wesley.
- Aravind K. Joshi, L. Levy, and M. Takahashi. 1975. Tree adjunct grammars. *Journal of the Computer and System Sciences*, 10:136–163.
- Carl Pollard and Ivan A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. University of Chicago Press, Chicago.
- Owen Rambow, K. Vijay-Shanker, and David Weir. 1995. D-Tree grammars. In *ACL*, pages 151–158.
- Yves Schabes. 1990. *Mathematical and Computational Aspects of Lexicalized Grammars*. Ph.D. thesis, Computer Science Department, University of Pennsylvania.
- Daniel Sleator and Davy Temperley. 1993. Parsing English with a link grammar. In *Third International Workshop on Parsing Technologies*.
- Mark Steedman. 2000. *The Syntactic Process*. MIT.
- Matthew Stone, Christine Doran, Bonnie Webber, Tonia Bleam, and Martha Palmer. 2001. Microplanning with communicative intentions: The SPUD system. *Under review*.
- Michael Tanenhaus and John Trueswell. 1995. Sentence processing. In P. Eimas and J. L. Miller, editors, *Handbook in perception and cognition: Speech Language and Communication*. Academic Press.