ELSEVIER

Cognitive Science 28 (2004) 637–668

COGNITIVE SCIENCE

http://www.elsevier.com/locate/cogsci

Starting with complex primitives pays off: complicate locally, simplify globally

Aravind K. Joshi

Department of Computer and Information Science, Institute for Research in Cognitive Science, University of Pennsylvania, Philadelphia, PA 19104, USA

Received 6 August 2003; received in revised form 4 March 2004; accepted 16 March 2004

Abstract

In setting up a formal system to specify a grammar formalism, the conventional (mathematical) wisdom is to start with primitives (basic primitive structures) as simple as possible, and then introduce various operations for constructing more complex structures. An alternate approach is to start with complex (more complicated) primitives, which directly capture some crucial linguistic properties and then introduce some general operations for composing these complex structures. These two approaches provide different domains of locality, i.e., domains over which various types of linguistic dependencies can be specified. The latter approach, characterized as *complicate locally, simplify globally* (CLSG), pushes non-local dependencies to become local, i.e., they arise in the basic primitive structures to start with.

The CLSG approach has led to some new insights into syntactic description, semantic composition, language generation, statistical processing, and psycholinguistic phenomena, all these with possible relevance to the cognitive architecture of language. In this paper, we will describe these results in an introductory manner making use of the framework of lexicalized tree-adjoining grammar (LTAG), a key example of the CLSG approach, thereby describing the interplay between formal analysis on the one hand and linguistic and processing issues on the other hand.

Keywords: Domain of locality; Non-local dependencies; Lexicalized tree-adjoining grammar; Nested and crossed dependencies; Word order variation and scope ambiguities; Grammar architecture and psycholinguistic properties

E-mail address: joshi@linc.cis.upenn.edu (A.K. Joshi).

0364-0213/$ – see front matter
doi:10.1016/j.cogsci.2004.03.007

1. Introduction

The conventional (mathematical) wisdom in specifying a grammar formalism is to start with basic primitive structures as simple as possible and then introduce various operations for constructing more complex structures. These operations can be simple or complex and the number of operations (although finite) need not be limited. New operations (simple or complex) can be introduced in order to describe more complex structures.

An alternate approach is to start with complex (more complicated) primitives, which capture directly some crucial linguistic properties and then introduce some general operations for composing these complex structures (primitive or derived). What is the nature of these complex primitives? In the conventional approach the primitive structures (or rules) are kept as simple as possible. This has the consequence that information (e.g., syntactic and semantic) about a lexical item (word) is distributed over more than one primitive structure. Therefore, the information associated with a lexical item is not captured locally, i.e., within the domain of a primitive structure. We will illustrate this in Section 1.1 in terms of the well-known context-free grammar (CFG) framework.

In contrast, in the alternate approach described in this paper, we allow the primitive structures to be as complex as necessary to capture all the relevant information about a lexical item in the local domain of a primitive structure . The kinds of information that need to be localized are as follows: (a) a lexical item taken as a predicate has zero or more arguments, (b) the arguments need to satisfy certain syntactic and semantic constraints, which are determined by the lexical item, and (c) the arguments will occupy different positions relative to the position of the lexical item. Hence, in the alternate approach, all the pieces of information associated with a lexical item have to be represented in the local domains of the primitive structures of the formal system. In Section 1.3, we will illustrate these ideas in terms of the lexicalized tree-adjoining grammar (LTAG), a class of grammars that illustrates this alternate approach by adopting it in its extreme form.

In this alternate approach, although the primitives are complex and there may be more than one primitive structure associated with a lexical item, the number of primitives is finite. Further, the combining operations are kept to the minimum and they are language independent (i.e., *universal*). It will be shown later that the alternate approach, which starts with complex primitives and can be characterized as *complicate locally, simplify globally* (CLSG), pushes non-local dependencies to become local, i.e., they are located in the basic primitive structures to start with. In the conventional approach they are spread out over one or more primitive structures. The architecture resulting from the CLSG approach has important implications for linguistics, computational linguistics, and psycholinguistics, including generation and acquisition. Some of these implications will be illustrated later by examples in Section 4.

We will discuss the various issues arising out of the CLSG approach in the context of the formal system known as the LTAG.[1] LTAG and some of its extensions have been investigated both formally and computationally for over twenty-five years.[2] LTAG represents a class of grammars that illustrates the CLSG approach by adopting it in an extreme form and thus serves to bring out the payoffs of the CLSG strategy. These payoffs are at least in the following key dimensions.[3]

- An extension of the notion of lexical ambiguity making it possible to unify some ambiguities, conventionally treated as structural ambiguities, with the usual lexical ambiguities, making them both lexical and leaving the other ambiguities as the only structural ambiguities (pure attachment ambiguities).
- Representation of scope ambiguities as attachment ambiguities in the syntax itself, resulting in syntactic derivations that are underspecified by just the right amount.
- Unification of even putative non-lexical information (which, in the conventional approaches, is scattered across different primitive structures, structure transformation rules, and logical forms) in terms of the finite set of primitive lexicalized structures of LTAG and the two combining operations of substitution and adjoining.

This unification is the result of a notion of lexicalization. In the LTAG conception of lexicalization, each primitive structure of LTAG is associated with a lexical item and the structure not only encapsulates all and only the arguments of the lexical anchor but crucially, also provides a structural slot for each one of the arguments. It is this notion of lexicalization that takes LTAG beyond other lexicalized grammars[4] and provides a notion of locality that results in these payoffs.

By using simple examples to illustrate the formal aspects that are needed to discuss various issues, I will try to communicate the interplay between formal analysis on the one hand and linguistic and processing issues on the other hand, something that, I hope, is consistent with the criteria for the David E. Rumelhart Prize. The work reported here is by no means a survey of all the formal work related to language, computation, and processing; it is only a slice from my particular perspective.[5]

The plan of the paper is as follows: in the remainder of Section 1 we will introduce the notions of domain of locality and lexicalization in the context of the well-known CFG and then show how LTAG arise in the process of lexicalizing CFGs and extending the domain of locality. In Section 2, we will also show how the architecture of the building blocks of LTAG directly predicts many complex dependency patterns and then summarize some important properties of LTAG. In Section 3, we will introduce two alternate perspectives for LTAG: (a) supertagging and (b) flexible composition, and discuss their implications for language description and language processing, and more importantly, their psycholinguistic relevance in, Section 4. In this section, we will also briefly mention some other implications of the LTAG architecture from the linguistic, computational, and psycholinguistic perspectives. Some of these issues are discussed by other authors in this issue of the journal. Finally, in Section 5, we will summarize the main issues concerning the CLSG approach.

1.1. Domain of locality of CFGs

In a CFG, the domain of locality is the one level tree corresponding to a rule in a CFG (Fig. 1) . It is easily seen that the arguments of a predicate (for example, the two arguments of *likes*) are not in the same local domain. The two arguments are distributed over the two rules (two domains of locality)—S $\rightarrow$ NP VP and VP $\rightarrow$ V NP. They can be brought together by introducing a rule S $\rightarrow$ NP V NP. However, then the structure provided by the VP node is lost. We should also note here that not every rule (domain) in the CFG in (Fig. 1) is lexicalized.

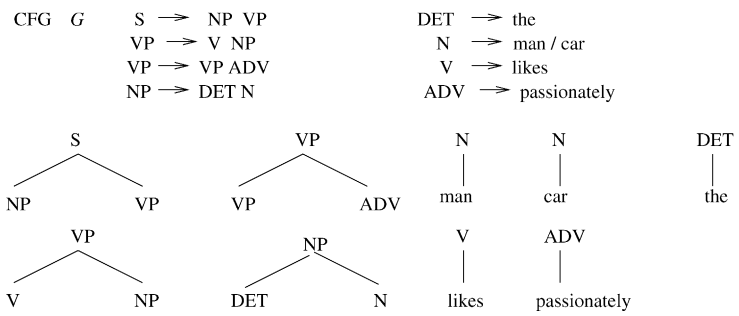


Fig. 1. Domain of locality of a context-free grammar.

The five rules on the right are lexicalized, i.e., they have a lexical anchor. The rules on the left are not lexicalized. The second, the third and the fourth rule on the left are almost lexicalized, in the sense that they each have at least one preterminal category (V in the second rule, ADV in the third rule, and DET and N in the fourth rule), i.e., by replacing V by *likes*, ADV by *passionately*, and either DET by *the* or N by *man*, these three rules will become lexicalized. However, the first rule on the left (S → NP VP) cannot be lexicalized, not certainly by *man*.

Can a CFG be lexicalized, i.e., given a CFG, G, can we construct another CFG, G', such that every rule in G' is lexicalized and $T(G)$, the set of (sentential) trees (i.e., the tree language of G) is the same as the tree language $T(G')$ of G'? Of course, if we require that only the string languages of G and G' be the same (i.e., they are weakly equivalent) then any CFG can be lexicalized. This follows from the fact that any CFG can be put in the Greibach normal form (see Linz, 2001) where each rule is of the form $A \rightarrow w\,B1\,B2\,\ldots\,Bn$ where w is a lexical item and the B's are non-terminals.[6] We call this *weak* lexicalization. The lexicalization we are interested in requires the tree languages (i.e., the set of structural descriptions) to be the same (i.e., strong equivalence). We call this *strong* lexicalization. It is easily seen, even from the example in (Fig. 1), that a non-lexicalized CFG cannot be necessarily strongly lexicalized by another CFG. Basically this follows from the fact that the domain of locality of CFG is a one level tree corresponding to a rule in the grammar (for detail see Joshi & Schabes, 1997). In Section 1.2 we will consider lexicalization of CFG by larger (extended) domain of locality.

Before proceeding further, it would be helpful to review certain definitions. The primitive structures of a formalism (also called elementary structures or elementary trees, as special cases) provide a *local* domain for specifying linguistic constraints (pieces of linguistic theory) in the sense that if the constraints are specifiable by referring to just the structures that are associated with the elementary structures then it is specifiable over the domain of these elementary structures. Therefore, we refer to the domains corresponding to the elementary structures as *domains of locality*. Formalism A is said to provide an *extended domain of locality* as compared to a formalism B if there is a linguistic constraint which is not specifiable in the local domains associated with B but which is specifiable in the local domains associated with A. The goal of the CLSG approach is to look for a formalism which provides local domains large enough so that, in principle, *all* linguistic constraints (pieces of linguistic theory) can be specified over these local domains. In the conventional approach (e.g., CFG-based) the specification of a constraint is often spread out over more than one local domain and thus the specification of

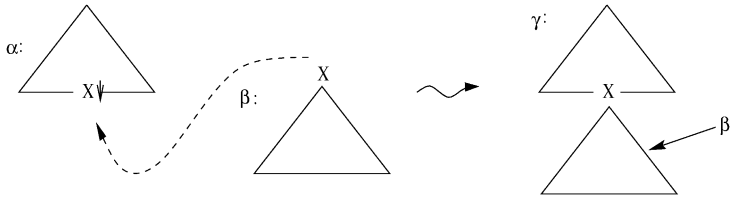


Fig. 2. Substitution.

a constraint is intertwined with the how the local domains are composed by the grammar, in other words, specification of a constraint will require specification of recursion, resulting in an effectively unbounded domain. In contrast, in the CLSG approach, we seek a system with extended (but still finite) domains of locality capable of specifying the linguistic constraints over these extended domains. Thus, recursion does not enter into the specification of the constraints. We call this property as *factoring recursion away from the domains of locality*.

1.2. Lexicalization of CFGs by grammars with larger domains of locality

Now we can ask the following question. Can we strongly lexicalize a CFG by a grammar with a larger domain of locality? Figs. 2 and 3 show a tree substitution grammar where the elementary objects (building blocks) are the three trees in Fig. 3 and the combining operation is the *tree substitution* operation shown in Fig. 2. The down arrows in Figs. 2 and 3 denote the substitution sites. Note that each tree in the tree substitution grammar (TSG), G' is lexicalized, i.e., it has a *lexical anchor*. It is easily seen that G' indeed strongly lexicalizes G. However, TSGs fail to strongly lexicalize CFGs in general. We show this by an example. Consider the CFG, G, in Fig. 4 and a proposed TSG, G'. It is easily seen that although G and G' are weakly equivalent they are not strongly equivalent. In G', suppose we start with the tree α_1 then by repeated substitutions of trees in G' (a node marked with a vertical arrow denotes a substitution site) we can grow the right side of α_1 as much as we want but we cannot grow the left side. Similarly for α_2 we can grow the left side as much as we want but not the right side. However, trees in G can grow on both sides. In order for a tree to grow on both sides, the distance between the lexical anchor of a tree, a, and the root of the tree, S, must become arbitrarily

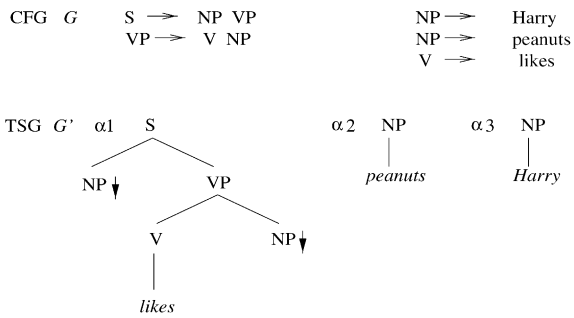


Fig. 3. Tree substitution grammar.

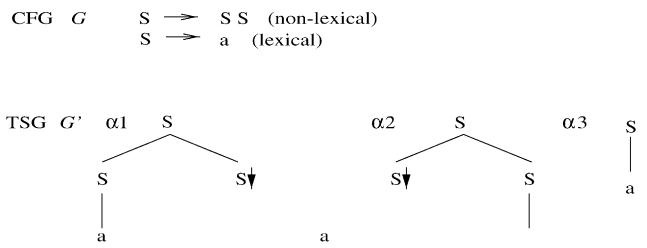


Fig. 4. A tree substitution grammar.

large. Substitution makes a tree grow only at the leaves of the tree and cannot make it grow internally. Hence, the TSG, G', cannot strongly lexicalize the CFG, G (Joshi & Schabes, 1997). Thus, even with the extended domain of locality of TSGs we cannot strongly lexicalize CFGs as long as substitution is the only operation for putting trees together.

We now introduce a new operation called *adjoining* as shown in Fig. 5. Adjoining involves splicing (inserting) one tree into another. More specifically, a tree β as shown in Fig. 5 is inserted (adjoined) into the tree α at the node X resulting in the tree γ. The tree β, called an *auxiliary tree*, has a special form. The root node is labeled with a non-terminal, say X and on the frontier there is also a node labeled X called the foot node (marked with *). There could be other nodes (terminal or non-terminal) on the frontier of β, the non-terminal nodes marked as substitution sites (with a vertical arrow). Thus, if there is another occurrence of X (other than the foot node marked with *) on the frontier of β it will be marked with the vertical arrow and that will be a substitution site. Given this specification, adjoining β to α at the node X in α is uniquely defined. Adjoining can also be seen as a pair of substitutions as follows: the subtree at X in α is detached, β is substituted at X and the detached subtree is then substituted at the foot node of β. A tree substitution grammar when augmented with the adjoining operation is called a tree-adjoining grammar (lexicalized tree-adjoining grammar because each elementary tree is lexically anchored). In short, LTAG consists of a finite set of *elementary trees*, each lexicalized with at least one lexical anchor. The elementary trees are either initial or auxiliary trees. Auxiliary trees have been defined already. *Initial* trees are those for which all non-terminal nodes on the frontier are substitution nodes. It can be shown that any CFG can be strongly lexicalized by an LTAG (Joshi & Schabes, 1997).

In Fig. 6, we show a TSG, G', augmented by the operation of adjoining, which strongly

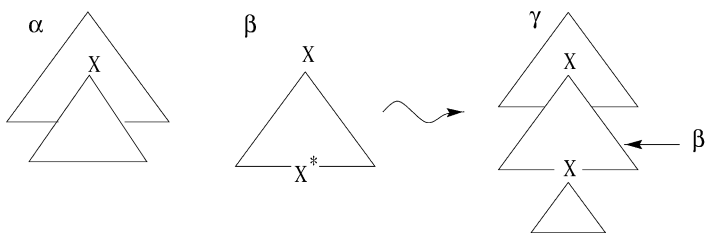


Fig. 5. Adjoining.

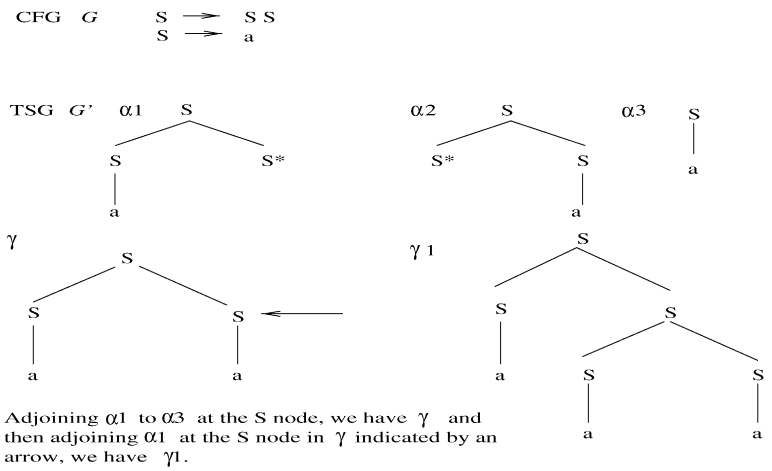


Fig. 6. Adjoining arises out of lexicalization.

lexicalizes the CFG, G. Note that the LTAG looks the same as the TSG considered in Fig. 4. However, now trees α_1 and α_2 are auxiliary trees (foot node marked with *) that can participate in adjoining. Since adjoining can insert a tree in the interior of another tree it is possible to grow both sides of the tree α_1 and tree α_2, which was not possible earlier with substitution alone. In summary, we have shown that by increasing the domain of locality we have achieved the following: (1) lexicalized each elementary domain, (2) introduced an operation of adjoining, which would not be possible without the increased domain of locality (note that with one level trees as elementary domains, adjoining becomes the same as substitution since there are no interior nodes to be operated upon), and (3) achieved strong lexicalization of CFGs.

1.3. Lexicalized tree-adjoining grammars

Rather than giving formal definitions for LTAG and derivations in LTAG, we will give a simple example to illustrate some key aspects of LTAG.[7] We show some elementary trees of a toy LTAG grammar for English. Fig. 7 shows two elementary trees for a verb such as *likes*. The tree α_1 is anchored on *likes* and encapsulates the two arguments of the verb. The tree α_2 corresponds to the object extraction construction. Since we need to encapsulate all the arguments of the verb in each elementary tree for *likes*, for the object extraction construction, for example, we need to make the elementary tree associated with *likes* large enough so that the extracted argument is in the same elementary domain. Thus, in α_2, the node for NP(wh) (the extracted argument) has to be in the tree for *likes*. Further, there is a dependency between the NP(wh) node and the NP node which is the complement of *likes* (i.e, to the right of V dominating *likes*) and this dependency is local to α_2. The tree α_2 not only shows that NP(wh) is an argument of *likes* but also that it is large enough to indicate a specific structural position for that argument.

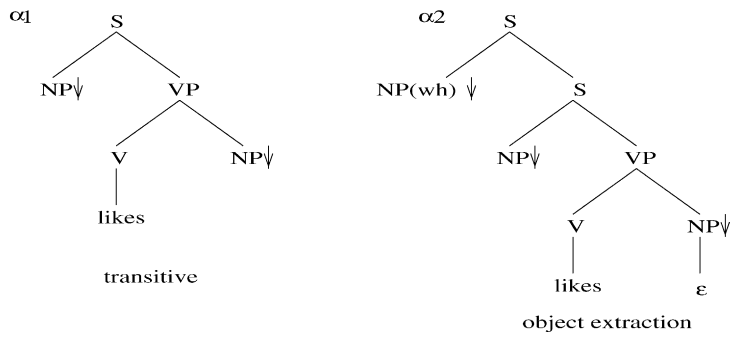


Fig. 7. LTAG: Elementary trees for *likes*.

Therefore, in principle, for each 'minimal' construction in which *likes* can appear (for example, subject extraction, topicalization, subject relative, object relative, passive, etc.) there will be an elementary tree associated with that construction. By *minimal* we mean that all recursion has been factored away. This factoring of recursion away from the domain over which the dependencies have to be specified is a crucial aspect of LTAGs, as they are used in linguistic descriptions. This factoring allows all dependencies to be localized in the elementary domains. In this sense, there will, therefore, be no long distance dependencies as such. They will all be local and will become long distance on account of the composition operations, especially adjoining. This will become clear as soon as we describe the derivation in Fig. 9.

Fig. 8 shows some additional elementary trees—trees α_3, α_4, and α_5 and trees β_1 and β_2. The β trees with foot nodes marked with * will enter a derivation by the operation of adjoining. The α trees enter a derivation by the operation of substitution.[8] A derivation using the trees α_2, α_3, α_4, α_5, β_1, and β_2 is shown in Fig. 9. The trees for *who* and *Harry* are substituted in the tree for *likes* at the respective NP nodes, at node addresses 1 and 2.1 in α_2. The tree for *Bill* is substituted in the tree for *think* at the NP node at the node address 1 in β_1. the tree for *does* is adjoined to the root node (address 0) of the tree for *think* tree (adjoining at the root node is a special case of adjoining), see Fig. 10 for this intermediate derived tree.

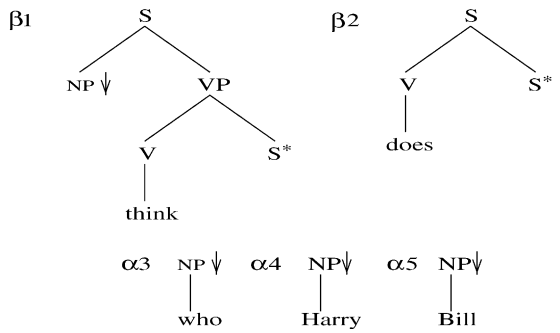


Fig. 8. LTAG: Sample elementary trees.

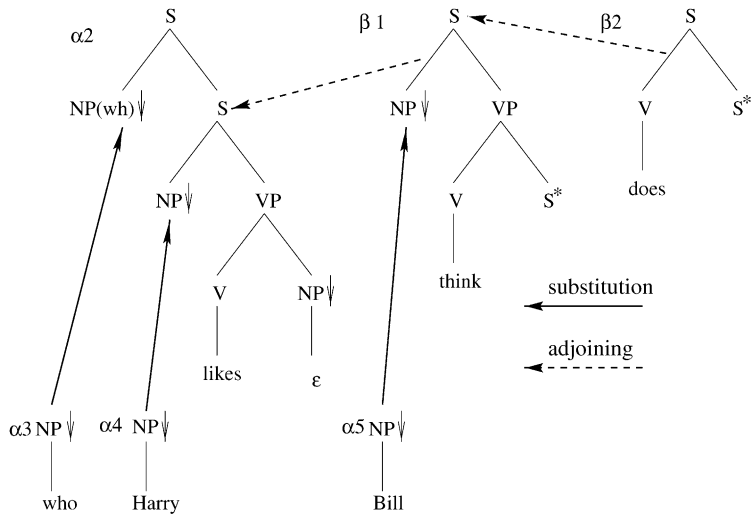


Fig. 9. LTAG derivation for *who does Bill think Harry likes*.

Finally, the derived auxiliary tree (after adjoining β_2 to β_1) is adjoined to the indicated interior S node of the tree α_2 at the address 2 in α_2. This derivation results in the *derived tree* for

Who does Bill think Harry likes

as shown in Fig. 11. Note that the dependency between *who* and the complement NP in α_2 (local to that tree) has been stretched in the derived tree in Fig. 11. It has become *long distance*. However, it started out as a local dependency. A key property of LTAGs is that all dependencies are local, i.e., they are specified in the elementary trees. They can become long distance as a result of the composition operations. Fig. 11 is the conventional tree associated with the sentence.

However, in LTAG, there is also a *derivation tree*, the tree that records the history of composition of the elementary trees associated with the lexical items in the sentence. This derivation tree is shown in Fig. 12. The nodes of the tree are labeled by the tree labels such as α_2 together with its lexical anchor *likes*.[9] The number on an edge of a derivation tree refers to the node address in a tree into which either a substitution or adjoining has been made. Thus, for example,

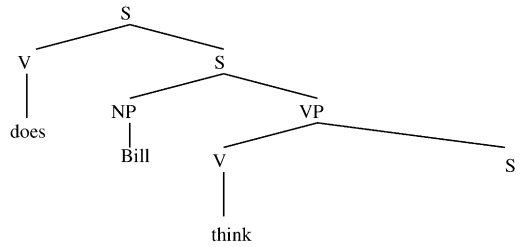


Fig. 10. Intermediate derived tree for β_2 adjoined to β_1.

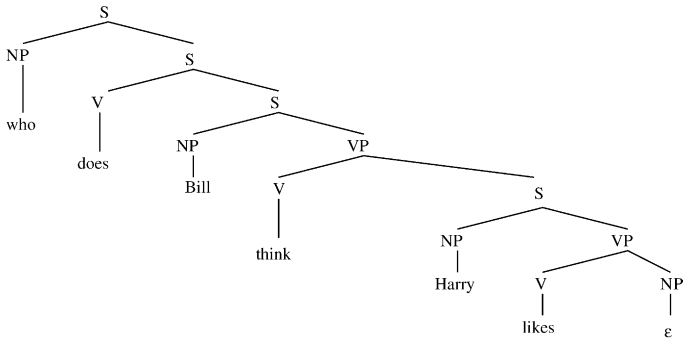


Fig. 11. LTAG derived tree for *who does Bill think Harry likes*.

in Fig. 12 the $\alpha_3(who)$ tree is substituted at the node with address 1 in the tree $\alpha_2(likes)$, the tree $\beta_1(thinks)$ is adjoined at the address 2 in the tree $\alpha_2(likes)$, etc. Solid edges denote substitution and dotted edges denote adjoining.

The derivation tree is the crucial derivation structure for LTAG. It records the history of composition in terms of the elementary trees (primitive building blocks) of LTAG. The derived tree in Fig. 11 does not indicate what the component elementary trees are for the final derived tree. It should be clear that from the derivation tree we can always obtain the derived tree by performing the substitutions and adjoinings indicated in the derivation tree. So, in this sense, the derived tree is redundant.

Further, for semantic computation, the derivation tree (and not the derived tree) is the crucial object. Compositional semantics is defined on the derivation tree. The idea is that for each elementary tree there is a semantic representation associated with it and these representations are composed using the derivation tree. Since the semantic representation for each elementary tree is directly associated with the tree, there is no need to reproduce necessarily the internal hierarchy in the elementary tree in the semantic representation (Kallmeyer & Joshi, 1999; Joshi & Vijay-Shanker, 1999; Joshi, Kallmeyer, & Romero, 2003). This means that the hierarchical structure internal to each elementary tree need not be reproduced in the semantic representation. This leads to the so-called *flat* semantic representation. i.e., the semantic expression associated with the sentence is essentially a conjunction of semantic expressions associated with each elementary tree.[10] Of course, relevant machinery has to be provided for scope information

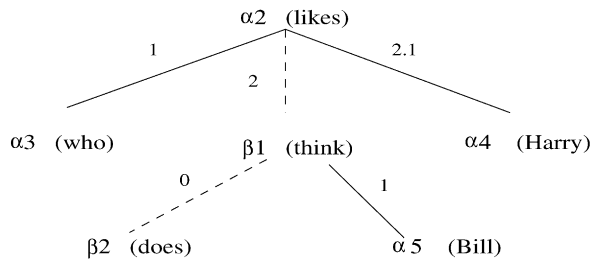


Fig. 12. LTAG derivation tree.

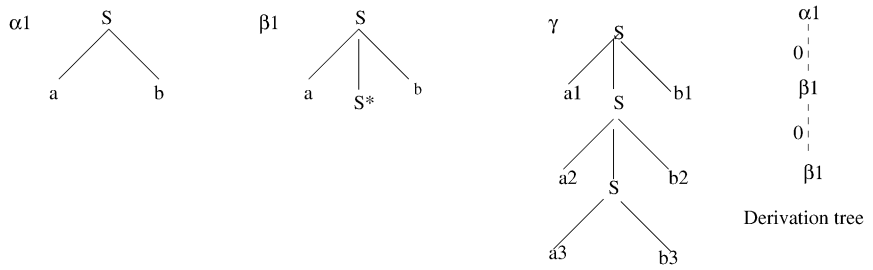
(for details see Kallmeyer and Joshi, 1999). The semantics need not be compositional at the level of the elementary trees. It is, however, compositional at the level of the derivation tree, i.e., at the level at which the elementary trees are assembled. This aspect of the architecture is also helpful in dealing with some of the non-compositional aspects, as in the case of rigid and flexible idioms (see Abeille, 2002; Stone & Doran, 1999).

2. Some important properties of LTAG

2.1. Nested and crossed dependencies

The two key properties of LTAG are (1) extended domain of locality (EDL; for example, as compared to CFG), which allows (2) factoring recursion from the domain of dependencies (FRD), thus making all dependencies local. All other properties of LTAG (mathematical, linguistic, and even psycholinguistic) follow from EDL and FRD. Roughly speaking, this follows from the fact that since all dependent elements are encapsulated in each elementary tree, all *interesting* properties of LTAG follow directly from the architecture of the elementary trees. We will illustrate this by considering nested and crossed dependencies. Fig. 13 shows an LTAG for nested dependencies. This is, of course, a CFG represented as an LTAG. The elementary trees α_1 and β_1 are both one level (depth one) trees and each tree encapsulates the two dependent elements a and b, which are at the same level (depth). The tree γ is obtained by starting with α_1, then adjoining the tree β_1 at the root node of α_1, and then adjoining another instance of β_1 at the root node of the previously derived tree. The indices on the dependent elements show the nested dependencies in the linear string associated with γ. The rightmost tree in Fig. 13 is the *derivation tree*. The nodes are labeled by the names of the elementary trees, α_1 and β_1. Going bottom–up on the derivation tree, β_1 adjoins to another instance of β_1 at the address 0 in β_1. Then, the derived β_1 adjoins to α_1 at the address 0 in α_1. Once we have the derivation tree, the derived tree is redundant, because it is the result of carrying out the compositions (in our case, by adjoining) as depicted in the derivation tree.

Fig. 14, on the other hand, shows an LTAG for crossed dependencies. Here the architecture of the elementary trees is different from the trees in Fig. 13. Each elementary tree is of depth



a1 a2 a3 b3 b2 b1 (indices, shown for convenience, indicate nested dependencies)

Fig. 13. An LTAG for nested dependencies.

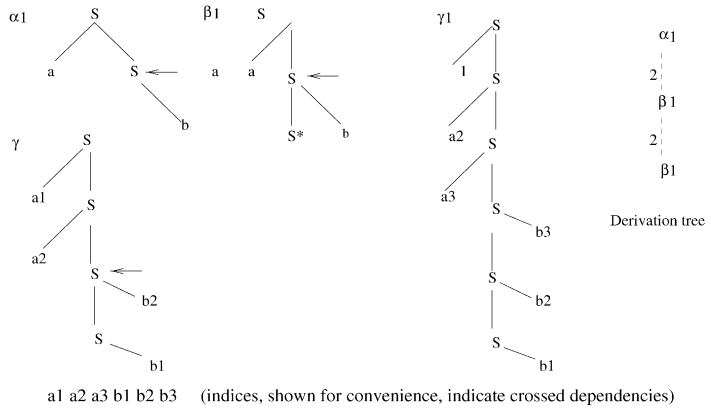


Fig. 14. An LTAG for crossed dependencies.

two and the two-dependent elements *a* and *b* in each elementary tree are not at the same depth, *b* is one level below *a*, although both are in the same tree. The derived tree γ_1 is obtained as follows. Starting with α_1 and then adjoining β_1 to the interior S node (as indicated by an arrow) of α_1, we get γ. Then, adjoining another instance of β_1 at the interior S node (as indicated by an arrow) of γ, we getγ_1. The indices on the dependent elements show the crossed dependencies in the linear string corresponding to γ_1. The rightmost tree in Fig. 14 is the *derivation tree*. This derivation tree is identical to the derivation tree in Fig. 13, except that β_1 in Fig. 14 is different from β_1 in Fig. 13. Further, going bottom up on the derivation tree, β_1 is adjoined to another instance of β_1 at the address 2 in β_1 and then the derived β_1 is adjoined to α_1 at the address 2 in α_1.

Note that the composition operation of adjoining is the same in both the examples. Whether we get nested dependencies or crossed dependencies is directly predicted from the architecture of the elementary trees. Nested dependencies arise due to center embedding in English as in (1) and due to complement embedding in German as in (2), (2′ shows the corresponding English word order).

(1) The rat_1 the cat_2 chased_2 ate_1 the cheese
(2) Hans_1 Peter_2 Marie_3 schwimmen_3 lassen_2 sah_1
(2′) Hans saw Peter make Marie swim

Both (1) and (2) are examples of nested dependencies. However, they arise for different reasons as we have stated above. This difference can be articulated in the architecture of the respective elementary trees for (1) and (2). We will not discuss this detail here but will comment on this issue in the context of some processing issues discussed in Section 4.

Crossing dependencies arise due to complement embedding in Dutch (under certain conditions) as in (3), (3′ shows the corresponding English word order).

(3) Jan_1 Piet_2 Marie_3 zag_1 laten_2 zwemmen_3
(3′) Jan saw Piet make Marie swim

It is possible to obtain a wide range of complex dependencies by manipulating the architecture of the elementary trees of LTAG, which result in combining nested and crossed dependencies in complex manners. Such dependencies arise due to complex word order phenomena such as *scrambling* and *clitic movement* and also due to scope ambiguities. We will briefly describe these issues in Section 4.

2.2. Some formal properties

We will list some important properties and describe them very briefly.

1. TAGs are more powerful than CFGs, both weakly and strongly, i.e., TAGs not only generate more languages (strings) than CFGs (weak sense) but also generate more structural descriptions (trees) than CFGs (strong sense). It is the strong power that is linguistically more relevant. Even if a language is context-free, a TAG for that language can assign structural descriptions not assignable by a CFG. Many complex word orders can be described as CFGs if we are concerned with strings only. However, if we want to make sure that for each string we have the correct structural description (i.e., correct semantics) then we may need more power, in the strong sense, than that provided by CFGs. We will pursue this issue in Section 4 in the context of a discussion of competence and performance properties.
2. TAGs carry over all formal properties of CFGs, modified in an appropriate manner, Thus, for example, the (worst case) parsing complexity of CFGs is $O(n^3)$, polynomial in n, where n is the length of the sentence, while it is $O(n^6)$ for TAGs.
3. CFGs are associated with pushdown automata (PDA) in the sense that for every CFG there is an associated PDA which recognizes precisely the same language which the corresponding CFG generates. Similarly TAGs are associated with the so-called Embedded Pushdowm Automata (EPDA) (Vijay-Shanker, 1987).

   EPDAs are a generalization of PDAs. In a PDA a move of the automaton is determined by (a) the input symbol, (b) the current state of the automaton, and(c) the topmost symbol on the stack. The result of the move takes the automaton to a new state, pushing a new sequence of symbols on the top of the stack or popping the topmost symbol from the stack.

   In an EPDA, a move may add a specified number of stacks to the left or right of the current stack. More specifically, a move of an EPDA is determined by (a) the input symbol, (b) the current state of the automaton, and (c) the topmost symbol on the (current) stack. The result of the move takes the automaton to a new state, pushing a new sequence of symbols on the (current) stack or popping the topmost symbol from the (current) stack, adding a specified (by the move) number of stacks to the left and right of the (current) stack, and pushing specified information on the newly introduced stacks. At the end of the move, crucially, the stack pointer points to the topmost symbol of the rightmost stack. It is this last requirement that makes an EPDA behave much like a PDA. The rightmost stack after a move becomes the current stack. During the moves of the automaton, if a stack becomes empty, we assume that it is eliminated from the sequence of stacks. In Section 4, we will explore some implications of the TAG–EPDA correspondence and the EPDA architecture for certain complex dependencies for human sentence processing.

4. TAGs (more precisely, the languages of TAGs) belong to a class of languages called *mildly context-sensitive languages* (MCSL). This class was proposed by Joshi (1985) with a suggestion that natural languages lie in this class. This class was characterized by the following properties: (a) worst case parsing complexity of a language in this class is polynomial, i.e., it is proportional to n^k, for some integer k and n is the length of the sentence. For CFGs $k = 3$ and for TAGs, $k = 6$, (b) grammars for languages in this class can characterize a limited set of patterns of nested and crossed dependencies and their combinations, (c) languages in this class have the constant growth property, i.e., if the sentences of language are arranged in increasing order of length then the difference between the lengths of successive sentences is bounded by a constant, and (d) context-free languages are properly included in this class.

   Property (a) just says that languages in MCSL are not too hard to parse. Property (b) says that complex dependency patterns are possible but completely arbitrary patterns of dependencies are not permitted. Property (c) captures the intuition that clauses have minimal lengths and a clause can be built further by adding a bounded amount of material to it. A sentence is made up of clauses and can be lengthened by adding a clause at a time. Property (d) says that MCSL is a proper extension of CFLs. CFGs have already been extensively used in formulating linguistic and computational theories, as well as psycholinguistic processing accounts.

   Over the past 20 years or so the MCSL hypothesis has generated very fruitful research in comparing different linguistic and formal proposals, and discovering some equivalences among formalisms, leading to a proper interplay of formal frameworks, substantive linguistic theories, and computational and processing paradigms.
5. Large scale wide coverage grammars have been built using LTAG, the XTAG system (LTAG grammar and lexicon for English and a parser) being the largest so far (for further details see XTAG, 2002).[11] In the XTAG system, each node in each LTAG tree is decorated with two feature structures (top and bottom feature structures), in contrast to the CFG-based feature structure grammars. This is necessary because adjoining can augment a tree internally, while in a CFG-based grammar a tree can be augmented only at the frontier. It is possible to define adjoining and substitution (as it is done in the XTAG system) in terms of appropriate unifications of the top and bottom feature structures. Because of factoring recursion from the domain of dependencies, there is no recursion in the feature structures. Therefore, in principle, feature structures can be eliminated. However, they are crucial for linguistic descriptions. Constraints on substitution and adjoining are modeled via these feature structures (Vijay-Shanker, 1987). This method of manipulating feature structures is a direct consequence of the extended domain of locality of LTAG.

3. Two alternate perspectives on LTAG

3.1. Supertagging

We will now describe a completely different perspective on LTAG. The elementary trees associated with a lexical item can be treated as if they are super parts-of-speech (super POS

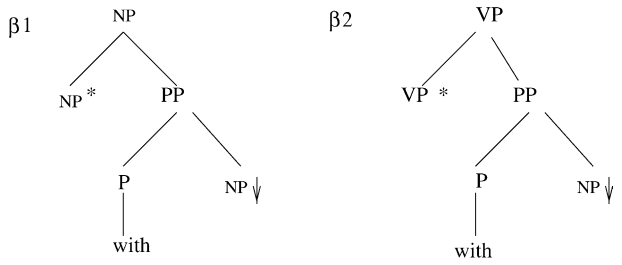


Fig. 15. Two supertags for *likes*.

(parts-of-speech) or supertags), in contrast to the standard POS such as V (verb), N (noun) etc. Now, it is well known that local statistical techniques can lead to remarkably successful disambiguation of standard POS. Can we apply these techniques for disambiguating supertags, which are very rich descriptions of the lexical items? If we can, then, indeed, this will lead to *almost* parsing. This approach is called supertagging (Joshi & Srinivas, 1994; Srinivas & Joshi, 1998).

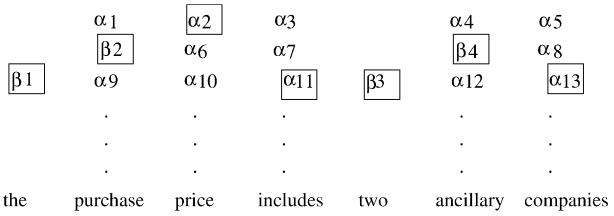
In Fig. 15, two elementary trees associated with the lexical item *likes* are shown. These are the same trees we have seen before. However, now we are going to regard these trees as super part-of-speech (supertags) associated with *likes*. Given a corpus parsed by an LTAG grammar, we can compute the statistics of supertags, statistics such as unigram, bigram, and trigram frequencies. Interestingly, these statistics combine not only lexical statistics but the statistics of constructions (as represented by the elementary trees) in which the items appear, thus combining lexical statistics with the statistics of the linguistic environments in which the lexical items appear. Thus, for example, consider the string as

The purchase price includes two ancillary companies

as shown in Fig. 16. The supertags associated with each word appear on top of that word. Some words have only only one supertag associated with them and others have more than one. In the current system there are about 15–20 supertags per word on the average, so there is a very high level of local ambiguity. In Fig. 17, the same supertags are shown for each word; however, for each word one supertag has been identified (in a box). This is the *correct* supertag for this word, in the sense that this is the supertag associated with this word in the correct parse of this sentence. Suppose we are able to find the correct supertag for each word in this sentence by applying local statistical disambiguation techniques, then for all practical purposes we will

	α1	α2	α3		α4	α5
	β2	α6	α7		β4	α8
β1	α9	α10	α11	β3	α12	α13
	.	.	.		.	.
	.	.	.		.	.
	.	.	.		.	.
the	purchase	price	includes	two	ancillary	companies

Fig. 16. A sentence with supertags for each word.



- Select the correct supertag for each word -- shown boxed
- Correct supertag for a word means the supertag that corresponds to that word in the correct parse of the sentence

Fig. 17. A sentence with the correct supertag for each word.

have parsed the sentence. It is not a complete parse because we have not put the supertags together, hence we call it an *almost* parse.

A supertagging experiment was carried out using trigrams of supertags[12] and techniques similar to the standard POS disambiguation techniques (Joshi & Srinivas, 1994; Srinivas & Joshi, 1998). The corpus used was the Wall Street Journal Corpus (WSJ). With a training corpus of 1 million words and a test corpus of 47,000 words, the baseline performance was 75% (i.e., 75% of the words received the correct supertag). The baseline corresponds to the case when the supertag chosen for a word is just the most frequent supertag for this word. We know from the performance of disambiguators for the standard POS that the baseline performance is 90% or better. The low baseline performance for supertagging is due to the fact that the local ambiguity is very high (about 15–20 on the average) in contrast to the local ambiguity of standard POS, which is about 1.5 for English. The performance of the trigram supertagger, on the other hand, is 92%. The improvement from 75% to 92% is indeed very remarkable. This means that 92% of the words received the correct supertag. More recent experiments based on other machine learning techniques have pushed the performance to about 93% Chen & Vijay-Shanker, 2000; Shen & Joshi, 2003).

Of course, more can be said about this supertagging approach. There are techniques to improve the performance and to make the output look more like a complete parse. We will not discuss these aspects; rather, we will talk about the abstract nature of supertagging and its relevance to the use of the CLSG approach. In supertagging we are working with complex (richer) descriptions of primitives (lexical items in our case). The descriptions of primitives are complex because we try to associate with each primitive all information relevant to that primitive. Making descriptions more complex has two consequences: (1) local ambiguity is increased, i.e., there are many more descriptions for each primitive, however, (2) these richer descriptions of primitives *locally* constrain each other. There is an analogy here to a jigsaw puzzle—the richer the description of each piece the better, in the sense that there are stronger constraints on what other pieces can go together with a given piece. Making the descriptions of primitives more complex allows us to compute statistics over these complex descriptions but, more importantly, these statistics are more meaningful because they capture the relevant dependencies directly (for example, word-to-word dependencies, where each word is the lexical anchor of some supertag, and word-to-construction dependencies). Local statistical computations over

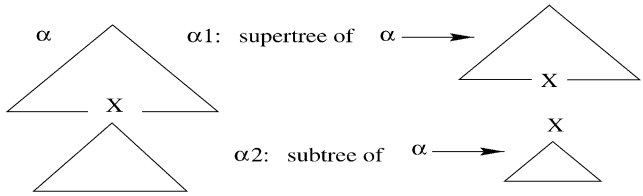


Fig. 18. Adjoining as wrapping 1.

these complex descriptions lead to robust and efficient processing. Supertagging by itself is not full parsing. However, parsing a sentence already supertagged is far more efficient (faster), on the average, as compared to parsing without supertagging. Supertagging is thus an example of a local computation on complex descriptions. Psycholinguistic relevance of supertagging is described in Section 4.

These considerations are directly relevant to other domains, such as AI. We can illustrate this by pointing out interesting relationships to the well known algorithm in Waltz (1975) for interpreting line drawings. What Waltz did was to make the descriptions of vertices more complex by adding information about the number and types of edges incident on a vertex. Again there is an analogy here to a jigsaw puzzle: the richer the description of a piece the better. By making the descriptions of vertices more complex, the local ambiguity was increased, for example, an L junction (a particular kind of junction in the taxonomy of junctions of edges) has about 92 physically possible labelings. However, local computations on these complex descriptions are adequate to rapidly disambiguate these descriptions, leading to efficient computation.

3.2. Flexible composition

In this Section, we will take a different perspective on the operation of adjoining and explore its implications for linguistic description. Psycholinguistic implications are described in Section 4. In adjoining, we insert an auxiliary tree, with the root and foot nodes labeled with X, into a tree at a node with label X. In Figs. 18 and 19, we present an alternate perspective on adjoining, leading to the notions of multi-component LTAGs and flexible composition.

The tree α which receives adjunction at X can be viewed as made up of two trees, the supertree at X and the subtree at X as shown in Fig. 18. Now, instead of the auxiliary tree β adjoined to the tree α at X, we can view this composition as a wrapping operation—the

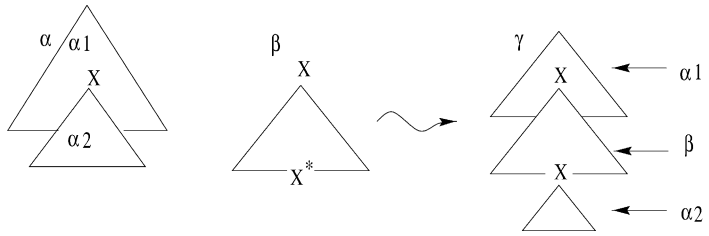


Fig. 19. Adjoining as wrapping 2.

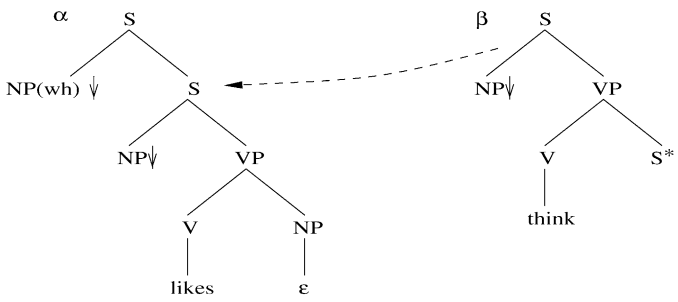


Fig. 20. Wrapping as substitution and adjunction 1.

supertree of α and the subtree of α are wrapped around the auxiliary tree β, as shown in Fig. 19. The resulting tree γ is the same as before. Wrapping of the supertree at the root node of β is like adjoining at the root (a special case of adjoining) and the wrapping of the subtree at the foot note of β is like substitution. Hence, this wrapping operation can be described in terms of substitution and adjoining. This is clearly seen in the linguistic example in Figs. 20 and 21. The auxiliary tree β can be adjoined to the tree α at the indicated node in α as shown in Fig. 20. Alternatively, we can view this composition as adjoining the supertree α_1 (the *wh* tree) at the root node of β and substitution of the subtree α_2 (the *likes* tree) at the foot node of β as shown in Fig. 21. The two ways of composing α and β are semantically coherent, in the sense that both ways of composing lead to the same semantics. The difference is that, in one case, β serves as the function and α as the argument and, in the other case, the two components of α (the supertree and the subtree of α at X) serve as the function and β serves as the argument.[13]

The wrapping perspective[14] can be formalized in terms of the so-called *multi-component LTAGs* (MC-LTAGs). They are called multi-component because the elementary objects can be sets of trees; in our examples, we have two components (in which α was split). When we deal with multi-components, we can violate the locality of the composition very quickly because the different components may be attached[15] (by adjoining or substitution) to different

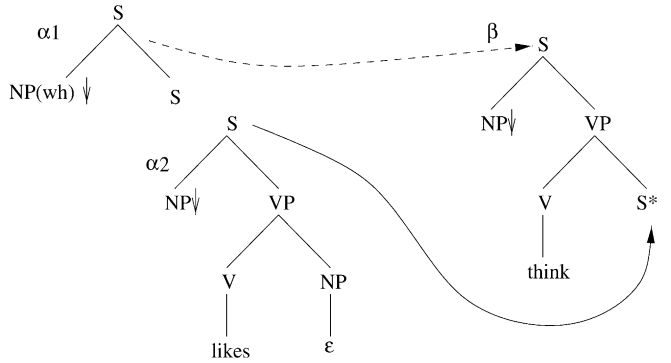


Fig. 21. Wrapping as substitution and adjunction 2.

nodes of a tree and these nodes may or may not be part of an elementary tree, depending on whether the tree receiving the multi-component attachments is an elementary or a derived tree. The obvious and natural way to preserve locality is to require that when the components of a multi-component tree (say, $\alpha_1 1$ and $\alpha_1 2$) attach to a tree (say, β) then β must be an elementary tree, i.e., a local domain. This requirement leads to what are known as tree-local MC-LTAGs. It is known that *tree-local MC-LTAGs* are weakly equivalent to LTAGs, i.e., we do not get increased weak generative power beyond LTAG. However, they can give rise to structural descriptions which are not obtainable by LTAGs, i.e., they are more powerful than LTAGs, in the sense of strong generative capacity (Weir, 1988). Thus, this alternate perspective leads to greater strong generative capacity without increasing the weak generative capacity.

We will now present an example illustrating the use of this alternate perspective in characterizing the *scope ambiguity* in:

Some student hates every course

This ambiguity arises because in the above example, either *some* scopes over *every* or vice versa, i.e., either 'there is some student (say, *s*) who hates every course' or 'for each course (say, *c*) there is some student (say, *t*) who hates that course'; two different courses need not be hated by the very same student.' This is illustrated in Figs. 22–24 (Kallmeyer & Joshi, 1999). In Fig. 22, we show a tree-local MC-LTAG for our example.[16] The trees for *hates*, *student*, and *course* are standard LTAG trees. The trees for *some* and *every* are multi-component trees. For example, the tree α_1 for *every* has two components, α_{11} and α_{12}, one of the components α_{11} is a degenerate tree in this special case. The multi-component tree, α_1, is lexically anchored by *some*. Similarly, for the tree α_2 for *every*. The main idea here is that the α_{12} component corresponds to the contribution of *some* to the predicate-argument structure of the tree for *hates* and the α_{11} component contributes to the scope structure. Similarly for the two components of α_2.

Fig. 23 shows the derivation. The main point to note here is that the two components of α_1 are attached (by substitution or adjoining) to α_3 at the appropriate nodes simultaneously. This composition is tree local as α_3 is an elementary tree. Similarly for the tree α_2. In this example, the two top components α_{11} and α_{21} are attached to the same node (the root node) of α_3.[17] This may give the impression that the composition is non-local because once α_1 is attached to α_3

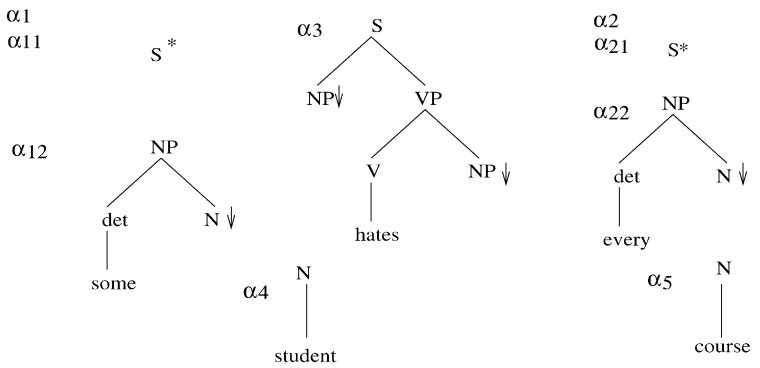


Fig. 22. Scope ambiguity: an example.

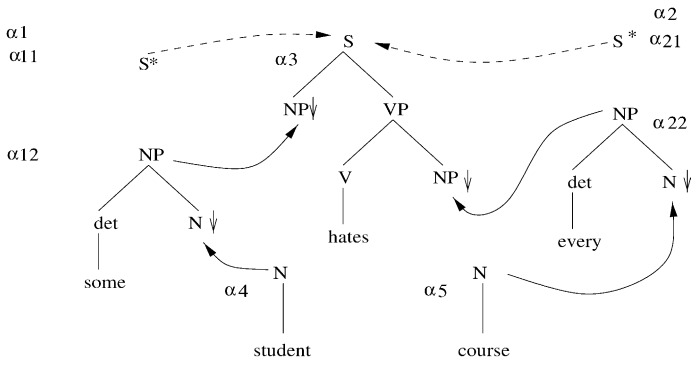


Fig. 23. Derivation with scope information.

we have a derived tree to which α_2 is attached. However, the two components, α_{11} and α_{21} are degenerate and it can be shown that in this case the composition of α_2 with α_3 (after α_1 has been composed with α_3) is still effectively tree-local (Kallmeyer & Joshi, 1999).

It is clear in this example that α_2 could have been attached to α_3 first and then α_1 attached to α_3. Fig. 24 shows the derivation tree for the derivation in Fig. 23. The numbers on the edges of the tree refer to the addresses for the attachments. Note that both α_{11} and α_{21}, the scope information carrying components, are attached to α_3 at the same node. Thus, they could be attached in any order (strictly speaking, α_1 and α_2 could be attached to α_3 in any order). Hence, α_{11} will outscope α_{21} if α_{21} is attached first and then α_{11} and vice versa. The scope ambiguity is thus directly reflected in the derivation tree for

Some student hates every course

This is in contrast to all other approaches (which are essentially CFG-based) where the scope ambiguity is represented at another level of representation. It is possible to represent in LTAG the scope ambiguity at the level of the derivation tree itself because of the alternate perspective on adjoining, which in turn is due to the extended domain of locality, a consequence of the CLSG approach.

More recently, similar ideas have been explored in the context of other linguistic phenomena such as scrambling and clitic climbing, both with respect to linguistic coverage and certain psycholinguistic implications. A particularly interesting result is that all word order variations

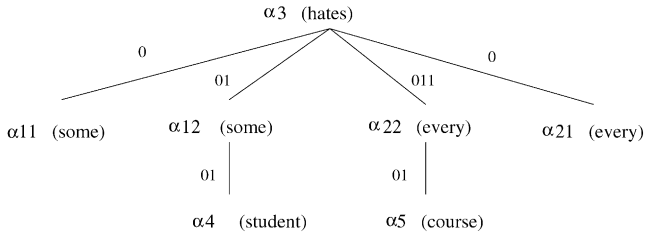


Fig. 24. Derivation tree with scope underspecification.

up to two levels of embedding (i.e., three clauses in all) can be correctly described by tree-local MC-LTAGs, correctly in the sense of providing the appropriate structural descriptions (correct semantics). Beyond two levels of embedding, not all patterns of word order variation will be correctly described (see (Joshi Becker, & Rambow, 2002) for details). Here, we will describe the main idea informally. Let us assume that we have three clauses, say, C_1, C_2, and C_3, each clause can be a single LTAG elementary tree or a multi-component TAG (MC-LTAG) tree set with two components. We assume that the tree which involves a verb in C_1 takes as an argument the tree containing a verb in C_2, i.e., C_1 embeds C_2, Similarly, we assume that C_2 embeds C_3. Thus, we have here a situation of two levels of center embeddings of complements. where C_1 embeds C_2 and C_2 embeds C_3. Now, it is possible to show (essentially by exhaustive enumeration) that, using flexible composition, all word orders can be generated preserving the embedding relationships, i.e., with correct semantics. Flexible composition allows us to compose the clauses in three ways, preserving semantics, (1) C_3 with C_2 and then the result with C_1, (2) C_1 with C_2 and then the result with C_3, or (3) both C_1 and C_3 into C_1. This third mode of composition is crucial to complete the proof. However, if we take four clauses, i.e., we have three levels of center embeddings of complements, then it can be shown (again by exhaustive enumeration) that there is at least one word order that cannot be generated without violating the semantics. The "trick" of using the third mode of composition described above does not work beyond two levels of center embeddings of complement clauses. Its use leads to violation of semantics for some cases. Some psycholinguistic implications of this result are discussed in Section 4 (see also Joshi et al., 2002; Kulick, 2000).

4. Processing issues

In this Section, we will discuss the implications of the TAG architecture for certain processing issues. These pertain to: (1) using supertags to make fine grained distinctions between lexical and structural ambiguities and their relevance to processing, (2) the relative processing complexities of certain embedding constructions; and (3) a different perspective on competence performance distinction.

4.1. Supertags in psycholinguistic models

Recently, there has been increasing convergence of perspectives in the fields of linguistics, computational linguistics, and psycholinguistics, especially with respect to the representation and processing of lexical and grammatical information. More specifically, this convergence is due to a shift to lexical and statistical approaches to sentence parsing. The particular integration of lexical and statistical information proposed in Kim, Srinivas, and Trueswell (2002) is highly relevant from the perspective of the LTAG architecture. As we have seen before in Section 3.1, LTAG associates with each lexical item one or more elementary structures (supertags), which encapsulate the syntactic and associated semantic dependencies. The computational results in supertagging as described earlier (Section 3.1) show that much of the computational work of linguistic analysis, which is traditionally viewed as the result of structure building operations, can be viewed as lexical disambiguation, in the sense of supertag disambiguation. If the

supertagging model is integrated in a psycholinguistics framework then one would predict that many of the initial processing commitments of syntactic analysis are made at the lexical level, in the sense of supertagging. The model proposed in Kim et al. (2002) is an integration of the Constraint-Based Lexicalist Theory (CBL; MacDonald, Pearlmutter, & Seidenberg, 1994), where the lexicon is represented as supertags with their distributions estimated from corpora as in the supertagging experiments described earlier (Section 3.1).

For example, in this model, there is a distinction between the prepositional phrase attachment ambiguity (PP ambiguity) as in (1) below

(1) I saw the man in the park with a telescope

and the PP attachment ambiguity as in (2) below

(2) The secretary of the general with red hair

In the first case, the PP (*with a telescope*) either modifies a noun phrase, *the man* or a verb phrase VP, headed by *saw*. There are two supertags associated with the preposition *with*, as in Fig. 25, one with the foot and root nodes being NP (supertag β_1) or both being VP (supertag $\beta 2$). That is, the ambiguity is resolved if we pick the correct supertag for *with* anchored on the preposition *with*. Thus, this PP attachment ambiguity will be resolved at the lexical level. However, in the second case, in both readings of (2) the supertag associated with *with* is the one whose root and foot nodes are both NP. Thus, in this case, the ambiguity will not be resolved at the lexical level. It can only be resolved at the level when the attachment is computed.

The first PP attachment ambiguity is not really an attachment ambiguity. It should be resolved at an earlier stage of processing. In the second case, it will be resolved at a later stage. Similarly, the ambiguity associated with a verb such as *forgot*, because it can take either an NP complement as in (3) below

(3) *The student forgot her name*

or a VP complement as in (4) below

(4) *The student forgot that the homework was due today*

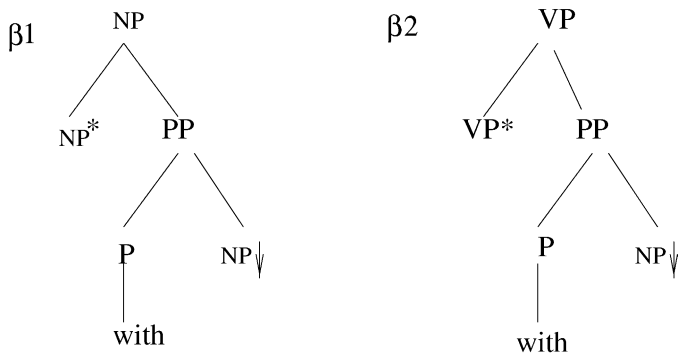


Fig. 25. Two supertags for *with*.

is a lexical (supertag) ambiguity and need not be viewed as a structural ambiguity. Kim et al. (2002) present a neural net-based architecture using supertags and confirm these and other related results.

4.2. Processing of crossed and nested dependencies

Context-free grammars and the associated automata, pushdown automata (PDA) have been extensively used in modeling several aspects of sentence processing. LTAGs are more powerful than CFGs both weakly, i.e., in terms of strings they produce and strongly, i.e., in terms of structures they produce. Further, LTAGs are associated with embedded pushdown automata (EPDA) (see Vijay-Shanker (1987)), an extension of pushdown automata, i.e., for every LTAG, G, there is an EPDA, M that recognizes the language of G and vice versa[18] TAGs and EPDAs provide a new perspective on the relative ease or difficulty of processing crossed and nested dependencies which arise in center embedding of complement constructions. We repeat here the examples of these constructions as discussed in Section 3.

(5) Hans_1 Peter_2 Marie_3 schwimmen_3 lassen_2 sah_1
(6) Jan_1 Piet_2 Marie_3 zag_1 laten_2 zwemmen_3
(7) Jan saw Piet make Marie swim

In (4) we have the nouns and the corresponding verbs in a nested order in a German construction and in (5) we have a crossed order in a related construction in Dutch. The indices on the nouns and verbs show these dependencies. The English word order is shown in (6). These are called complement embedding constructions because each verb is embedded in a higher verb of which it is a complement, except, of course, the matrix (top level and tensed) verb, i.e., *make* is embedded in *saw* and *swim* is embedded in *make*, *saw* is the matrix verb.

In (4) and (5) we have center embedding of the complements. In the corresponding English construction as in (6) we do not have center embedding, the complements are just iterated.

The main observation for our purpose is that Standard German prefers the nested order and Standard Dutch prefers the crossed order.[19] In the well-known study by Bach, Brown and Marslen-Wilson (1986) the authors investigated the consequences of these differences between German and Dutch for the processing complexity of sentences, containing either nested or crossed dependencies. Stated very simply, their results show that Dutch is *easier* than German. More specifically, in their study "German and Dutch subjects performed two tasks—ratings of comprehensibility and a test of successful comprehension—on matched sets of sentences which varied in complexity from a simple sentence to one containing three levels of embedding", three levels means three verbs and three nouns, as in our examples above. Their results show "no difference between Dutch and German for sentences within the normal range (up to one level), but with a significant preference emerging for the Dutch crossed order. These results in Bach et al. (1986) show that pushdown automaton (PDA) cannot be the universal basis for human parsing mechanism (as the authors themselves point out). They offer an explanation for the inadequacy of PDA, based on the kinds of partial interpretations that the nested and crossed dependencies allow the listener to construct. Their main suggestion was "that the most important variable in successful parsing and interpretation is not simply *when* information becomes available, but also *what* you can do with that information when

you get it." Thus in (4) (German example), when the deepest noun (noun phrase) and verb are reached (*Marie schwimmen*), we have a verb and its argument, however, we do not know at this stage where this structure belongs, i.e., we do not have a higher structure into which we can integrate this information. Hence, we must hold this information until a higher structure becomes available. The same consideration holds for the second noun and the second verb (*Peter lassen*).

In contrast, in (5) (Dutch example), we begin to build the matrix of higher verbs as soon as the verb cluster begins, i.e., beginning with the matrix verb *zag* and the corresponding noun argument which can be integrated, similarly for the second noun and the second verb, and for the third noun and the third verb. In this case, we do not create intermediate structures that do not have a place to fit into. Thus, we first build the structure for the matrix verb into which the next verb (with its structure) can be integrated, and similarly for the last verb.

The nested dependencies in German permit integration of structures (innermost to outermost) in a context-free manner (hence processed by PDA) but it is not possible to decide what to do with that information until the higher level verb(s) become available. Therefore, PDA will not serve as a processing model for the German case (even though the dependencies are nested) and, of course, it will not serve for the Dutch case, because PDAs cannot model crossed dependencies in the first place. Now, it turns out that EPDA can model both crossed and, of course, nested dependencies and the interpretations can be carried out in the way suggested in Bach et al. (1986), i.e., when a noun verb structure is integrated, the structure to which it belongs has already been created before (Joshi, 1990). We now introduce the following measure[20] of complexity—maximum number of items (say, m) from the input that have to be held back before the sentence parsing (interpretation) is complete. In Joshi (1990) it is further shown that on this measure Dutch is easier (smaller m) than German, in fact, approximately in the same proportion as in the experimental data in Bach et al. (1986), m for German is approximately twice that for Dutch. Further, linguistically relevant TAG grammars for (4) and (5) correspond (in the formal sense of the association between EPDAs and TAGs) to the EPDAs that correctly model the processing of (4) and (5).

4.3. A new twist to the competence performance distinction

How do we decide whether a certain property of language is a competence property or a performance property? This is an old question. Our main claim in this section is that the answer to this question is not given a priori. It depends on the formal devices available to us for describing language. If a formal device corresponding to the property is available then we are presented with a choice. We can claim that the property as either a competence property or a performance property. When there is no formal device available then we are left with no choice but to treat the property as a performance property.

In the context of the LTAG framework, especially, the multi-component LTAG (MC-LTAG), discussed in Section 3, this question has an interesting answer when we look at a variety of word order phenomena, such as scrambling, clitic climbing, and even scope ambiguities. Since the CLSG approach leads directly to LTAG, the claim here is that it is the CLSG approach that leads to this answer.[21]

As a particular example, consider the phenomenon of *scrambling* (in German, for example) as illustrated below. We repeat (4) in Section 4.2 as (7) below.

(8) $Hans_1\ Peter_2\ Marie_3\ schwimmen_3\ lassen_2\ sah_1$

The three verbs are in the standard order. However, it is possible, in principle, to have the three nouns in any order different from the order in (8), keeping the verbs in the same order as in (7). Thus (9) below is a scrambled version of (8).

(9) $Hans_1\ Marie_3\ Peter_2\ schwimmen_3\ lassen_2\ sah_1$

Thus, in general, we can have

(10) $P(N_1, N_2, \ldots N_k)\, V_k V_{k-1} \ldots V_1$

where $P(N_1, N_2, \ldots N_k)$ stands for some permutation of the k nouns. It is not surprising that sentences involving scrambling from more than two levels of embedding are indeed difficult to interpret and native speakers show reluctance to accepting these sentences. This situation is reminiscent of the difficulty of processing English sentences with more than two (or perhaps even more than one) center embedded relative clauses, as in (10) below.

(10) *The rat the cat the dog chased bit ate the cheese*

Such a difficulty is characterized as a performance property, say, P, where P is the property that beyond two levels[22] of embedding the human processing device, whatever it is, simply fails. Thus, in analogy to the center embedding of relative clauses in English, we could say that in the case of scrambling, processing difficulty (or failure to process) for sentences with center embedding of complement clauses beyond two levels of embedding is also a performance property, say Q. From this perspective, both P and Q are seen as performance properties and not properties of the respective associated grammars. We will point out below that there is a very interesting difference between these two cases.

Let us first consider the case of center embedding of relative clauses in English. Is it possible to claim the property P is a property of a *class* of grammars, say Γ such that for all grammars in this class, P holds, i.e., for each grammar G in Γ, up to two levels of embedding, G will assign correct (i.e, semantically coherent) structural descriptions to all sentences, however, beyond two levels of embedding, there is no guarantee that for all such sentences the grammar will assign correct structural descriptions. If we can exhibit such a class Γ then we could claim that P is a competence property, i.e., it is a property of each grammar in this *class*. In other words, for each grammar G in Γ the property P holds up to two levels of embedding only and then fails to hold, and the failure is due to the inability of G to assign correct structural descriptions to all sentences beyond two levels of embedding. The idea here is that this lack of ability to assign correct structural descriptions (i.e., correct semantics) is the reason for the processing difficulty.

Now, it turns out that we do not know whether such a *class* Γ exists. The class of finite state grammars F will not do because although there is a grammar in this class that will work up to two levels of embedding and then fail, there is clearly another grammar that will work up to three levels of embedding, another up to four levels, and so on. Thus, P does not hold for the class Γ. We could choose the class of CFG, the next one in the Chomsky hierarchy. In this

case, an arbitrary number of center embeddings will be permitted. This is because once we allow center embedding (thus nested dependencies, which characterizes the class CFG) then we cannot put a bound on the number of embeddings. Hence, the class CFG will not work for us also. In fact, as far as we know, at present, there is no class of grammars which has the property P.[23]

Since we cannot find a class of grammars with the property P we have no opportunity to claim that P is competence property. So in a way, we have no choice and we must conclude that P is a performance property.[24]

However, when we consider the case of scrambling from more than two levels of center embedded complement clauses (see (9) above) we can actually exhibit a class of grammars Δ which has the property that for each grammar in this class, say, G, the property Q (the property corresponding to the property P discussed above) holds, i.e., for each grammar G in this class, G will assign correct structural descriptions to all sentences with scrambling from up to two levels of embedding; however, beyond two levels of embedding, there is no guarantee that G will assign correct structural description (correct semantics) to all such sentences. In other words, syntax allows for arbitrary number of levels of embedding but beyond two levels of embeding, correct semantics is not guaranteed for all sentences with more than two levels of embedding. It is shown in Joshi et al. (2002) that the class of multicomponent LTAG (MC-LTAG) is such a class, i.e., it has the property Q.

So, now, we have an opportunity to claim that Q is a competence property, i.e., the difficulty of processing sentences with scrambling from more than two levels of embedding can be characterized as a property of the grammar. Note that the claim here is not that we *must* conclude that Q has to be a competence property. The claim is that we are presented with a *choice*. We can claim Q to be a competence property by adopting Δ (the class MC-LTAG) as the class of grammars for describing scrambling from center embedded complement clauses or we can continue to follow the traditional wisdom (as we did in the case of center embedding of relative clauses in English) and say that Q is a performance property. As far as we know, this is the first example of a situation where we have an opportunity to claim a particular processing difficulty as a competence property.

For the case of center embedding of relative clauses, we had no choice as we are unable to exhibit the relevant class of grammars, Γ. Structurally, center embedding of relative clauses in English is a different phenomenon as compared to the center embeddings of complement clauses in German. Whether this difference has something to do with the conclusion we have just reached is an open question. The class MC-LTAG does not work for the center embeddings of relative clauses in English, as can be seen in (7) above which, mathematically, looks the same as arbitrary number of center embeddings of relative clauses as in English.

To sum up, we have discussed the following issue: How do we decide whether a certain property of language is a competence property or a performance property? We showed that the answer to this question is not given a priori. It depends on the formal devices (formal grammars and machines) available to us for describing language. We showed that in the case of center embedding of relative clauses in English, we were forced to conclude that the property that sentences with more than two levels of embedding are difficult to interpret is a performance property and not a competence property. In contrast, for the case of scrambling (as in German), with respect to a corresponding property we can indeed exhibit a class of grammars such that, for

all grammars in this class, the corresponding property holds. Thus, we are given the opportunity to claim this property to be a competence property, rather than a performance property. To the best of our knowledge, this is the first such case where a choice of this kind is presented. The lack of such a choice before this case was discovered has led to the traditional assumption that all properties similar to the processing difficulty associated with center embedding of relative clauses in English should be treated as performance properties. Our main conclusion is that this assumption is not justified at all!

To avoid any misunderstanding, let us restate the main claim as follows. We are not claiming that by finding an appropriate class of grammars we can conclude that a property associated with processing difficulty automatically becomes a competence property. What we are claiming is that in this case we are presented with a *choice*. Without an associated class of grammars, we have no choice but to regard the property as a performance property. Until the appropriate class of grammars associated with the property discussed in this section (property Q) was discovered, we had no choice. Now we have a choice!

4.4. Related work on processing

Implications of the LTAG architecture have been explored in the areas of language acquisition and production (generation). Frank (1998) has studied structural complexity and the time course of grammatical development. He attempts to relate children's difficulties with certain constructions to the processing load and representational complexity in the context of a particular view of syntactic representation, which is derivable from the TAG architecture. Ferreira (2000) presents a perspective on production that relates the TAG architecture to a range of experiments about production. Insights from the LTAG theory have been extended to the domain of discourse structure, primarily by treating discourse connectives as lexical anchors of LTAG like trees at the discourse level, thereby, blurring the line between sentence level and discourse level descriptions. This work is described in a series of recent papers (Forbes et al., 2001; Webber, Joshi, Knott, & Stone, 1999; Webber, Stone, Joshi, & Knott, 2003).

5. Summary

We have described an approach to formal systems where we start with complex (rather than simple) primitive structures, complex enough to localize all relevant properties of lexical items. Under this approach (characterized as the CLSG approach—complicate locally, simplify globally)—the grammar is characterized by the (finite) set of primitive structures together with two combining operations (substitution and adjoining), which are universal. Specification of the finite set of primitives is the linguistic theory, in a sense. We have illustrated the formal, linguistic, and computational consequences of the CLSG approach in the context of the formal system known as the LTAG and some of its variants. We have also discussed the implications of this approach for several psycholinguistic issues. In general, under this approach, several important properties of language arise as corollaries of the formal system and therefore, they are not stipulative. Thus, this research is an example of productive interplay between formal analysis on the one hand and linguistic and processing issues on the other hand.

Notes

1. There is another dimension in which formal systems can be characterized. One could start with an unconstrained formal system (turing machine equivalent, for example) and then add linguistic constraints, which become in a sense, all stipulative. Alternatively, one could start with a formal system that is constrained and just adequate for describing language. The formal constraints then become universal, in a sense. All other linguistic constraints become stipulative and language specific. Now it turns out that the CLSG approach leads to constrained formal systems, LTAG in particular. This convergence is of interest in its own right. However, in this paper, we will not discuss constrained systems in their own right. Our focus will be on the CLSG approach and its implications for the architecture of the grammars and their processors.
2. See Joshi, Levy and Takahashi (1975); Joshi (1985), Kroch and Joshi (1985), Vijay-Shanker (1987), Weir (1988), Kroch (1989), Kroch and Santorini (1991), Schabes (1992), Rambow (1994), Resnik (1992), Schabes and Waters (1994), Chiang (2000), Sarkar (2002), Abeillé (2002), Prolo (2003).
3. Using the CLSG approach it is also possible to study directly many aspects of strong generative capacity, in terms of the set of structural descriptions (i.e., sentence structures), provided by a formal system and not just the strings (i.e., sentences). The former are more relevant to the linguistic descriptions (Joshi, 2003; Miller, 1999). This aspect is not pursued in this paper.
4. For example, in a categorial grammar (CG) each lexical item is also associated with an expression that encodes the arguments of that lexical item. However, in this representation, the different structural positions occupied by the different arguments are not shown in the primitive expression associated with the lexical item. These positions implicitly arise in the course of the derivation. In this sense LTAG can be viewed as a class of grammars emerging from the full use of the CLSG approach.
5. There are several formal systems that are clearly related to LTAG. Some examples are Combinatory Categorial Grammars (CCG) (Steedman, 1996), Stabler's version of minimalist grammars (Stabler, 1997), Lexical Functional Grammars (LFG) (Kaplan & Bresnan, 1983), Head Driven Phrase Structure Grammars (HPSG) (Pollard & Sag, 1994), for a constrained version of HPSG, see Kiefer, Netter and Vijay-Shanker (1995), Frank (2002). Linear Indexed Grammars (LIG) by Gazdar, and HeadGrammars (HG) by Pollard, CCG, and LTAG have all been shown to be weakly equivalent, i.e., in terms of the string sets they generate but not in terms of the structural descriptions. These relationships have been discussed extensively in Joshi, Vijay-Shanker and Weir (1991). CCGs are close to LTAGs, however, LTAGs are *fully lexicalized*, in the sense that each elementary tree of LTAG not only encodes the argument structure of the lexical anchor but also the structural positions the arguments would occupy in minimal clauses. In this sense, the elementary trees of LTAG can be considered as structured categories (see Section 3 for supertags), see also Joshi and Kulick (1997).
6. The Greibach form of the rule is related to the categories in a categorial grammar.
7. In the actual LTAG grammar each node in an elementary tree is decorated with attribute value structures (feature structures) which encode various linguistic constraints specified

over the domain of an elementary tree. There is no recursion in these feature structures. We omit these details here as they are not essential for our immediate purpose.

8. This distinction between the two types of elementary trees is characterized in LTAG in terms of *initial trees* (the α trees) and the *auxiliary trees* (the β trees).
9. The derivation trees of LTAG have a close relationship to dependency trees, although there are some crucial differences. The semantic dependencies are the same, however.
10. The general notion of flat semantics is related to the notion of minimal recursion semantics (MRS; Copestake, Flickinger, Sag, & Pollard, 1999). MRS has been used in the semantic computation in the HPSG framwork. In the LTAG framework the notion of elementary trees and the derivation tree which specifies the composition in terms of the elementary trees directly provides a representation for computing a *flat* semantics.
11. A large French grammar, FTAG is described in Abeillé & Candito (2002) and Abeillé (2002). Other sizable LTAG grammars exist for German, Hindi, and Korean. There may be some others that I am not aware of at this time.
12. These are trigrams of supertags and not just trigrams of words or even words with standard POS. These trigrams are trigrams of words, together with the supertags associated with these words.
13. Such flexibility is also available in a CFG-based system. Thus, in a CFG rule $A \rightarrow BC$, B can be treated as a function and C as the argument, resulting in A, or alternatively, C as the function and B as the argument, also resulting in A. However, it is easily seen that this flexibility does not lead to any increase in the strong generative capacity (no additional structural descriptions). This is so because a CFG rule is essentially a string rewriting rule and we are composing essentially flat structures. In an LTAG, we are composing larger structures (extended domains of locality) and therefore, flexibility of composition can provide greater strong generative capacity. In a categorial grammar (CG), A, B, and C are structured categories. Flexible composition (which is achieved by the operation of type raising) can give additional structural descriptions but not to the extent that is possible in LTAG. This is because, although both LTAG and CG encode the arguments of a lexical item, in the LTAG elementary structures, the different positions occupied by each argument are also encoded. It is this property of LTAG and the fact that in LTAG we compose trees and not strings as in a CFG or CG that leads to the increased strong generative power.
14. Readers familiar with categorial grammars will notice some similarity to the wrapping rule in (Bach, 1988), which he introduced to relate the structure of *persuaded John to leave* to the structure of *persuade to leave John*, the former derived from the latter by an operation of wrapping. The wrapping operation we have defined is much more general and involves structures far more articulated than those in CFG, CG, and Bach's wrapping rule.
15. We will use the term *attachment* as a cover term for both substitution and adjoining.
16. The component α_{11} is an auxiliary tree with just one node. Thus, in this case, the root node and the foot node of this auxiliary tree are the same. The $*$ on the S node has the same meaning as the $*$ on the foot node of an auxiliary tree.
17. In general, in a multi-component LTAG, multiple adjunctions to the same node are not allowed as this violates the tree-locality and also takes the system beyond LTAG.

However, we are dealing here with a special case. The components entering multiple adjunctions are degenerate. It can be shown that in this case the weak generative capacity is still the same as for LTAGs.

18. EPDAs are thus a precise computational model for the languages of LTAGs. For a brief description of EPDA, see item 3 in Section 2.2. Lack of space prevents us from giving full details of this processing model.
19. Strictly speaking this word order appears in subordinate clauses. See also Kroch and Joshi (1985), Kroch (1989), and Kroch and Santorini (1991).
20. Suggested by a measure used in Bach et al. (1986) for evaluating the complexity of comprehension.
21. The possibility of some other formal approach leading to a similar answer cannot be ruled out. However, to my knowledge, no such formal approach is known so far.
22. We could set this level to one but this is not essential for the rest of the argument of this section. I will keep the level at two in order to make it parallel to the scrambling situation.
23. Obviously, we cannot limit our search to the grammars in the Chomsky hierarchy. Any class of grammars defined in a formal framework is a possible candidate. However, to my knowledge, we do not have such a candidate at present.
24. Miller and Chomsky (1963) were the first to discuss the relationship between formal grammars (and associated machines) and center embedding of relative clauses in English. Since then a large number of publications on this issue have appeared.

Acknowledgments

This work was partially supported by the National Science Foundation Grant STC-SBR 8920230. My work on the formal analysis of language began a little over three decades ago. I want to take this opportunity to express my sincere thanks to all my past and present students, collaborators, colleagues, and mentors (too many to enumerate by name) for their wonderful insights, advice, and comments throughout this period. I also greatly appreciate all the constructive comments provided by the three reviewers and the action editor, which resulted in significant improvements in the final version of this paper.

References

Abeillé, A. (2002). *Une grammaire électronique du français, CNRS Éditions*. Paris.

Abeillé, A., & Candito, M. (2002). FTAG: A lexicalized tree adjoining grammar for French. In A. Abeillé & O. Rambow (Eds.), *Tree-adjoining grammar*, CSLI, Stanford.

Bach, E. (1988). Categorial grammars as theories of language. In R. T. Oehrle, E. Bach, & D. Wheeler (Eds.), *Categorial grammars and natural language structures* (pp. 17–34). Dordrecht: Reidel.

Bach, E., Brown, C., & Marslen-Wilson, W. (1986). Crossed and nested dependencies in German and Dutch: a psycholinguistic study. *Language and Cognitive Processes*, *1*(4), 249–262.

Chen, J. C., & Vijay-Shanker, K. (2000). Automated extraction of the TAGs from the Penn Treebank. In *Proceedings of the 6th International Workshop on Parsing Technology (IWPT)* (pp. 65–76).

Chiang, D. (2000) Statistical parsing with an automatically extracted tree adjoining grammar. In *Proceedings of the Association for Computational Linguistics (ACL) 2000 Meeting*.

Copestake, A., Flickinger, D., Sag, I., & Pollard, C. (1999). *Minimal recursion semantics: An introduction manuscript*. Stanford University.

Frank, R. (1998). Structural complexity and the time course of grammatical development. *Cognition*, *66*, 249–301.

Frank, R. (2002). *Phrase structure composition and syntactic dependencies*. Cambridge: MIT Press.

Ferreira, F. (2000). Syntax in language production: An approach using tree-adjoining grammars. In L. Wheeldon (Ed.), *Aspects of language production*.

Forbes, K., Miltsakaki, E., Prasad, R., Sarkar, A., Joshi, A. K., & Webber, B. (2001). D-LTAG system-discourse parsing with a lexicalized tree-adjoining grammar. In *ESSLI'2001. Workshop on information structure, discourse structure and discourse semantics*. Helsinki.

Joshi, A. K. (1985). Tree-adjoining grammars: How much context sensitivity is required to provide reasonable structural descriptions? In D. Dowty, L. Karttunen, & A. Zwicky (Eds.), *Natural language parsing*(pp. 206–250). Cambridge: Cambridge University Press.

Joshi, A. K. (1990). Processing crossed and nested dependencies: An automaton perspective on the psycholinguistic results. *Language and Cognitive Processes*, *5*, 1–27.

Joshi, A. K. (2003). A note on the strong and weak generative powers of formal systems. *Theoretical Computer Science*, *293*, 243–259.

Joshi, A. K., Becker, T., & Rambow O. (2002). Complexity of scrambling: A new twist to the competence-performance distinction. In A. Abeillé & O. Rambow (Eds.), *Tree-adjoining grammars*. Stanford: CSLI.

Joshi, A. K., Kallmeyer, L., & Romero, M. (2003). Flexible composition in LTAG: Quantifier scope and inverse linking. In *Proceedings of the International Workshop on Computational Semantics (IWCS-5)*. Tilburg.

Joshi, A. K., & Kulick, S. (1997). Partial proof trees as building blocks for a categorial grammar. *Linguistics and Philosophy*, *20*, 637–667.

Joshi, A. K., Levy, L. S., & Takahashi, M. (1975). Tree adjunct grammars. *Journal of Computer and System Sciences*, *10*, 1.

Joshi, A. K., Vijay-Shanker, K., & Weir, D. (1991). The convergence of mildly context sensitive grammatical formalisms. In P. Sells, S. Shieber, & T. Wasow (Eds.), *Foundational issues in natural language processing*. Cambridge: MIT Press.

Joshi, A. K., & Schabes, Y. (1997). Tree-adjoining grammars. In G. Rosenberg & A. Salomaa (Eds.), *Handbook of formal languages* (pp. 69–123). Berlin: Springer.

Joshi, A. K., & Srinivas, B. (1994). Disambiguation of super parts of speech (Supertags): Almost parsing, In *Proceedings of the 1994 International Conference on Computational Linguistics (COLING)*. Japan: Kyoto.

Joshi, A. K., & Vijay-Shanker, K. (1999). Compositional semantics with lexicalized tree-adjoining grammar (LTAG): How much underspecification is necessary? In H. C. Bunt & E. G. C. Thijsse (Eds.), *Proceedings of the Third International Workshop on Computational Semantics (IWCS-3)* (pp. 131–145). Tilburg.

Kallmeyer, L., & Joshi, A. K. (1999). Factoring predicate argument and scope semantics: Underspecified semantics with LTAG. In *Proceedings of the Twelfth Amsterdam Colloquium* (pp. 169–174). Amsterdam: University of Amsterdam (a revised version appears in *Language and Computation*, 2003).

Kaplan, R., & Bresnan, J. (1983). Lexical-functional grammar: A formal system of grammatical representation. In J. Bresnan (Ed.), *The mental representation of grammatical relations*. Cambridge, MA: MIT Press.

Kasper, R., Kiefer, B., Netter, K., & Vijay-Shanker, K. (1995). Compilation of HPSG to TAG. In *Proceedings of the Association for Computational Linguistics (ACL)* (pp. 92–99).

Kim, A., Srinivas, B., & Trueswell, J. C. (2002). The convergence of lexicalist perspectives in psycholinguistic and computational linguistics. In P. Merlo & S. Stevenson (Eds.), *Sentence processing and the lexicon: Formal, computational and experimental perspectives*. Philadelphia: John Benjamin Publishing.

Kroch, A. (1989). Asymmetries in long distance extraction in a tree-adjoining grammar. In M. Baltin & Kroch, A. (Eds.), *Alternative conceptions of phrase structure*. Chicago: University of Chicago Press.

Kroch, A., & Joshi, A. K. (1985). *Linguistic relevance of tree-adjoining grammars* (Technical Report). Department of Computer and Information Science, University of Pennsylvania.

Kroch, A., & Santorini, B. (1991). The derived constituent structure of the West Germanic verb raising constructions. In R. Freiden (Ed.), *Principles and parameters in comparative grammar* (pp. 269–338). Cambridge: MIT Press.

Kulick, S. (2000). *Constrained non-locality: Long-distance dependencies in TAG*. Ph.D. Dissertation, University of Pennsylvania, Philadelphia, PA, USA.

Linz, P. (2001). *An introduction to formal languages and automata*. Sudbury, MA: Jones and Bartlett Publishers.

MacDonald, M. C., Pearlmutter, N. J., & Seidenberg, M. S. (1994). Lexical nature of syntactic ambiguity resolution. *Psychological Review*, *101*, 676–703.

Miller, P. M. (1999). Strong generative capacity. Stanford CA: CSLI Publications, Stanford University.

Miller, G. A., & Chomsky, N. (1963). Finitary models of language users. In R. D. Luce., R. Bush, & E. Galanter (Eds.), *Handbook of mathematical psychology* (Vol. II). New York: Wiley.

Pollard, C., & Sag, I. A. (1994). *Head-driven phrase structure grammar*. Chicago: Chicago University Press.

Prolo, C. (2003). LR parsing for tree-adjoining grammars and its applications to corpus-based natural language parsing. Ph.D. Dissertation, University of Pennsylvania, Philadelphia.

Rambow, O. (1994). *Formal and computational aspects of natural language syntax*. Ph.D. Dissertation, University of Pennsylvania, Philadelphia.

Resnik, P. (1992). Probabilistic tree-adjoining grammars as a framework for statistical natural language processing. *In Proceedings of COLING '92, Nantes, 2*, 418–424.

Sarkar, A. (2002). Combining labeled and unlabeled data in statistical natural language processing. Ph.D. Dissertation. University of Pennsylvania, Philadelphia.

Schabes, Y. (1992). Stochastic lexicalized grammars. In *Proceedings of COLING '92* (pp. 426–432) Nantes.

Schabes, Y., & Waters, R. C. (1994). *Tree insertion grammars* (Technical Report TR-94-13). Cambridge: Mitsubishi Electric Research Laboratories.

Shen, L., & Joshi, A. (2003). A SNoW based supertagger the applications to NP Chunking. In *Proceedings of the Association for Computational Linguistics Meeting (ACL)*(pp. 89–96). Sapporo, Japan.

Srinivas, B., & Joshi, A. K. (1998). Supertagging: An approach to almost parsing. *Computational Linguistics*, *22*, 1–29.

Stabler, E. (1997). Derivational minimalism. In C. Retore (Ed.), *Logical aspects of computational linguistics* (pp. 68–95). Heidelberg: Springer Verlag (Lecture Notes in Artificial Intelligence 1328).

Steedman, M. (1996). *Surface structure and interpretation*. Cambridge, MA: MIT Press.

Stone, M., & Doran, C. (1999). Sentence planning as description using tree-adjoining grammar. In *Procceedings of the Association for Computational Linguistics (ACL) Meeting*. Madrid.

Vijay-Shanker K. (1987). *A study of tree-adjoining grammars*. Ph.D. Dissertation, University of Pennsylvania, Philadelphia.

Waltz, D. (1975). Understanding line drawings of scenes with shadows. In P. H. Winston (Ed.), *The psychology of computer vision*. New York: McGraw-Hill.

Webber, B., Joshi, A. K., Knott, A., & Stone, M. (1999). What are little texts made of? A structural and presuppositional account using lexicalized TAG. In *Proceedings of the International Workshop on Levels of Representation in Discourse (LORID '99)* (pp. 151–156). Edinburgh.

Webber, B., Stone, M., Joshi, A. K., & Knott, A. (2003). Anaphora and discourse structure. *Computational Linguistics*, *29*, 545–588.

Weir, D. (1988). *Characterizing mildly context-sensitive grammar formalisms*. Ph.D. Dissertation. University of Pennsylvania, Philadelphia.

The XTAG Research Group. (2002). *A lexicalized tree-adjoining grammar for english* (Technical Report 02-18). Institute for Research in Cognitive Science, University of Pennsylvania, Philadelphia.