Tree Adjoining Grammars

Anne Abeillé, Owen Rambow (editors)

CENTER FOR THE STUDY
OF LANGUAGE
AND INFORMATION

1

Tree Adjoining Grammar: An Overview

Anne Abeill, Owen Rambow

1.1 Introduction

Tree Adjoining Grammar (TAG) is a **constrained mathematical formalism** that supports the development of **lexicalized grammars** for natural languages. Let us examine the two key concepts contained in this statement in more detail.

A constrained mathematical formalism is a mathematical device for specifying sets of structures, such as sets of strings, sets of trees, sets of feature structures, or other such sets. By "constrained" we mean that the mathematical device cannot specify *all* possible sets. We say "mathematical formalism" because we are not (yet) talking about a linguistic theory, just about formal ways to generate sets of structures. By using the term "mathematical", we mean that the elementary structures and the way in which they are used to create larger structures are defined precisely.

We are interested in constrained mathematical formalisms for two reasons:

- They are linguistically appealing because the scope of the theory of grammar is restricted. We return to this issue in Section 1.3.
- They are computationally appealing because they allow us to define efficient processing models. We return to this issue in Section 1.2.5 and in Section 1.6.2.

We now turn to the other key concept, lexicalization. One of the most important developments in the study of natural language has been the increasing emphasis on the importance of the lexicon. There are two

types of motivation for this. First, many syntactic phenomena, including those at the interface to semantics, can easily be described or explained by referring to specific lexical items or classes of lexical items. For example, some verbs take three arguments (1a) while others do not (1b and 1c), and among those that do, some allow the double-object construction (1a), while others do not (1d), instead allowing only a prepositional object (1e).

(1) a. He told Mary a secret
    b. *He saw Mary (to/from/...) a secret
    c. *He saw a secret (to/from/...) Mary
    d. *He divulged Mary a secret
    e. He divulged a secret to Mary

The second type of motivation for studying lexicalized formalisms comes from applications. In natural language processing, the use of corpora has become popular and corpora typically consist of words. Thus, it is desirable to develop grammars in a formalism that directly supports the association of grammatical structure with words.

The best-known constrained mathematical formalism used for the description of natural language syntax is the Context-Free Grammar (CFG). Not coincidentally, CFGs were defined by (Chomsky 1957), laying the foundation for both the formal study of natural languages within linguistics and the study of formal languages within theoretical computer science. While CFGs continue to be important tools in computer science, their appeal to linguists has ebbed, for two reasons:

- CFGs appear to have the wrong descriptive level, since they do not allow us to easily state the kind of lexical information mentioned above.[1] We explore this issue more in Section 1.2.
- Starting shortly after the introduction of CFGs, there was considerable debate as to whether all natural languages can be described by a CFG. This debate appears to have ceased after (Shieber 1985) argued that data from Swiss German can be interpreted as showing that it cannot be generated by a CFG.

Thus, there has been a wide-spread conclusion that we need to go beyond CFG in defining a formalism for natural language syntax. There have been two ways in which this has been done. In the first approach, the CFG is maintained, but a formal extension is added. In transformational grammar (Chomsky 1957), an initial structure is generated using

[1] GPSG (Gazdar et al. 1985) represents an attempt to overcome the descriptive limitations of CFG.

a CFG, which is subsequently changed by transformations. In Lexical-Functional Grammar (LFG) (Kaplan and Bresnan 1982), the constituent (c-) structure is described by a CFG, but its derivation is controled by a separate functional (f-) structure. In Head-Driven Phrase Structure Grammar (HPSG) (Pollard and Sag 1994), the CFG is (essentially) retained as backbone, but it is augmented with unlimited feature structures. In all cases, the descriptive power is increased, as is the formal power, so that the two problems described above are both overcome. However, at the same time, the resulting formalisms are no longer constrained![2] This was shown by (Peters and Ritchie 1973) for transformational grammar, by (Maxwell and Kaplan 1993) for LFG, and by (Carpenter 1991) for HPSG.

In the second approach, CFG is not maintained, even as a component. What is maintained is the spirit of CFG: a simple formalism in which a sequence of context-free rewrite steps yields the result. This is the approach followed by TAG, in which the string rewriting of CFG is replaced by tree rewriting. As a result, TAG provides us with the right descriptive level for developing a lexically-oriented representation of natural language syntax, including the interface to semantics. Specifically, TAG allows us to develop grammars which are *lexicalized*, and whose derivations represent structural information which CFGs cannot represent. At the same time, TAG is a constrained mathematical formalism with appealing formal and computational properties.

In this chapter, we introduce TAG and describe some of the work done using TAG. We start out in Section 1.2 by motivating and defining TAG, and by comparing it further to CFG. We continue by discussing the relationship between TAG and linguistic grammars in Section 1.3. In Section 1.4 we present some variants of TAG. We then discuss the relation of TAG to other frameworks in Section 1.5. Finally, we discuss various applications of TAG in Section 1.6, including parsing, generation, and psycholinguistic modeling. For other overviews of TAG, see (Joshi et al. 1991, Joshi 1994, Frank 2000).

1.2 A Constrained Mathematical Formalism for Natural Language

In this section, we will motivate Tree Adjoining Grammar by showing how it arises from an attempt to lexicalize a context-free grammar.

[2] We observe that in the case of transformational grammar, later definitions ("move-α") no longer are mathematical formalisms in our sense, as the definition of the possible operations is somewhat vague. At the same time, the authors of transformational grammar explicitly abandoned the attempt to provide a mathematical formalism for natural language syntax.

1.2.1 The Anatomy of Formal Systems

If we want to analyze how formal systems can be used for linguistic theories, we must start by determining what sort of *elementary structures* the formalism provides, and how these elementary structures are combined using the *combining operations* defined in the formalism. We will illustrate these notions with some examples. First, consider CFGs. In a CFG, a grammar consists of a set of rewrite rules, which associate a single nonterminal symbol with a string of terminal and nonterminal symbols. Here is a sample context-free grammar:

(2) a. S $\longrightarrow$ NP VP
b. VP $\longrightarrow$ really VP
c. VP $\longrightarrow$ V NP
d. V $\longrightarrow$ likes
e. NP $\longrightarrow$ John
f. NP $\longrightarrow$ Lyn

Each of these rules is an elementary structure in this grammar. We combine these elementary structures by using one rule to rewrite a symbol introduced by a previous application of some rule. For example, when we use rule (2a), we introduce the nonterminal symbols (or "nodes") NP and VP. We may rewrite the VP node by using rule (2b) or (2c). This grammar generates, among others, the following string:

(3) John really likes Lyn

Derivations in CFGs can be represented as trees: for each nonterminal node in the tree, the daughters record which rule was used to rewrite it. The phrase-structure tree that corresponds to sentence (3) is given in Figure 1.

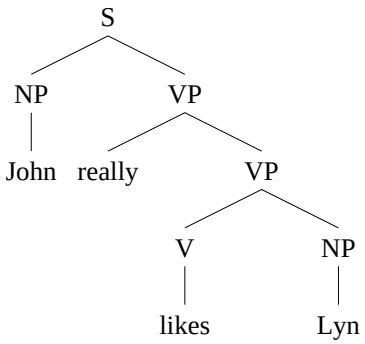

FIGURE 1 Phrase Structure Tree for John really likes Lyn

Now consider a different type of mathematical formalism, Tree Substitution Grammars (TSG). In a TSG, the elementary structures are phrase-structure trees. A sample grammar is given in Figure 2. It consists of three trees, one of which is rooted in S, and two of which are rooted in NP. Note that even though from the point of view of a CFG, a tree is a derivation tree, not an elementary object or a derived object, we have defined TSGs in such a way that a tree is now an elementary object of the grammar, as well as the derived object.

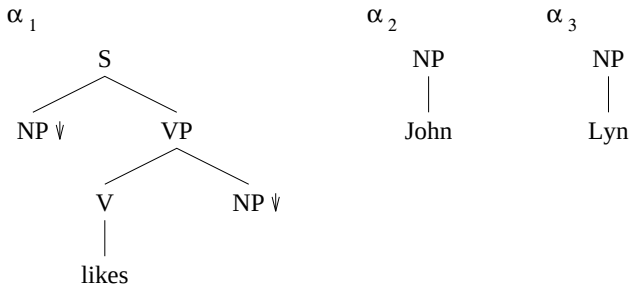

FIGURE 2 A sample TSG

We combine elementary structures in a TSG by using the operation of *substitution*, illustrated schematically in Figure 3. We can substitute tree β into tree α if there is a nonterminal symbol on the frontier of α which has the same label as the root node of β ('A' in Figure 3). We can then simply append β to α at that node.[3] A derivation in our sample TSG is shown in Figure 4. The trees representing the two arguments of the verb *like*, *John* (α_2) and *Lyn* (α_3), are substituted into the tree associated with the verb (α_1), yielding the well-formed tree α_4, from which the sentence *John likes Lyn* can be read off.

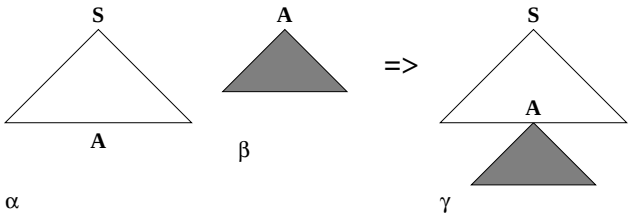

FIGURE 3 The Substitution Operation

[3] Nodes at which substitution is possible are called "substitution nodes" and are marked with down-arrows (↓).

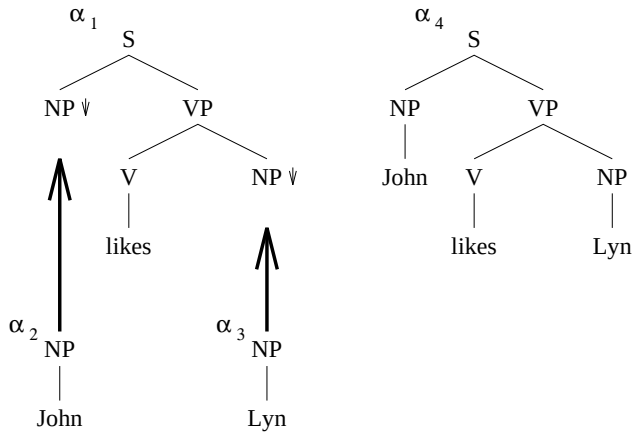

FIGURE 4 Substitution of arguments into initial tree of likes

CFGs and TSGs are weakly equivalent (i.e., they generate the same string languages). However, to a linguist, they look very different. A context-free rule contains a single "level", i.e., a phrase-structure node and its daughters; an elementary tree in a TSG may be of arbitrary height. Put differently, in TSG we have increased the **domain of locality** of the elementary structures of the grammar. This increased domain of locality allows the linguist to state linguistic relationships (such as subcategorization, semantic roles of arguments, case assignment and agreement) differently in a TSG. As an example, take agreement between subject and verb in English. The linguist working in TSG can simply state (by using some feature-based notation) that the verb and the NP in subject position in tree α_1 of Figure 2 agree with respect to number. The linguist working in CFG has a harder time: since the verb is in rule (2d), while the subject NP is in rule (2a), he or she cannot simply state the relation directly, since it is impossible to state constraints that relate nodes in different elementary structures. Instead, the linguist must propose that the NP in fact agrees with the VP in rule (2a), and that the VP agreement features are inherited by its head in rule (2d). The notion that the VP (and not only the verb) agrees with the subject may be a meaningful linguistic proposition, and in fact the TSG linguist could have adopted it as well. However, the crucial issue is that the CFG linguist, because of his choice of formalism, was *forced* to adopt it, while the TSG linguist may choose to do so or not on purely linguistic grounds.

Another example of the linguistic use of the extended domain of locality is subcategorization, which we already mentioned in Section 1.1.

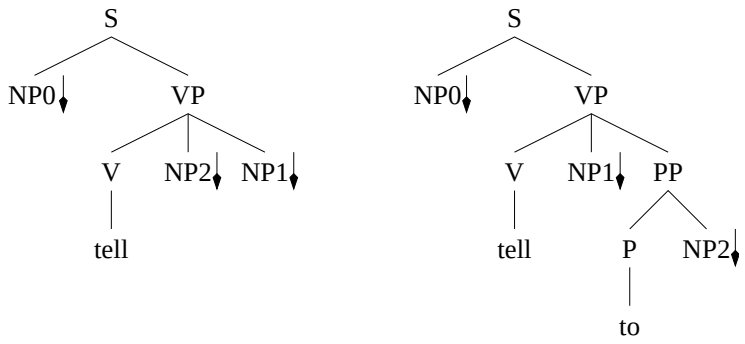

FIGURE 5 Two trees for tell

We could associate *tell* with the two trees in Figure 5, while associating *divulge* only with the one on the right (replacing *tell* with *divulge*).[4] In a CFG, we would need a separate system of features to control the appearance of arguments for the lexical heads (and perhaps more features for strongly governed prepositions such as the *to* in this example); these features are entirely superfluous in TSG.

1.2.2 Lexicalization

Now let us turn to our central concern, the role of the lexicon. We will call a grammar *lexicalized* if every elementary structure is associated with exactly one lexical item (which can consist of several words), and if every lexical item of the language is associated with a finite set of elementary structures in the grammar.

CFGs cannot be lexicalized in a linguistically meaningful manner.[5] Consider the sample grammar given above in (2). We can see that no lexical item is associated with rules (2a) and (2c); therefore, the grammar is not lexicalized. It would be possible to combine rules (2a), (2c) and (2d) into a single one:

(4) S $\longrightarrow$ NP likes NP

However, it is now impossible to correctly place the adverb *really*, since a (lexicalized) rule of the form (2b) is no longer useful (the VP

[4] In some diagrams, we distinguish among different nodes which have the same nonterminal node label by adding an integer to the label. These integers are not part of the nonterminal node label, and are used for presentational purposes only.

[5] A CFG in Greibach Normal Form is a lexicalized CFG; however, in general, the Greibach Normal Form of a linguistically meaningful CFG will not itself be linguistically meaningful.

node having been eliminated). The adverb cannot be inserted between the subject and the verb.

There is a second way of lexicalizing a CFG: instead of merging the two phrase-structure rules into a single rewrite rule, we can combine them and consider the result – a fragment of a phrase structure tree – an elementary structure. Put differently, we move from CFG to TSG. For example, tree α_1 in Figure 2 is the result of combining rules (2a), (2c) and (2d). As desired, tree α_1 is associated with exactly one lexical item, the verb *likes*. Thus, we have now obtained a TSG from a CFG. We can derive the sentence *John likes Lyn* as shown previously in Figure 4.

It turns out that a TSG is not really what we want, either: we are again faced with the problem of getting the adverb in the right place, since there is no node into which to substitute it.[6] This problem is solved by the tree composition operation of *adjunction*, first introduced by (Joshi et al. 1975). Adjunction is shown in Figure 6. Tree α (called an "initial tree") contains a nonterminal node labeled A (not on its frontier); the root node of tree β (an "auxiliary tree") is also labeled A, as is exactly one non-terminal node on its frontier (the "foot node"). All other frontier nodes are terminal nodes or substitution nodes. We take tree α and remove the subtree rooted at its node A, insert in its stead tree β, and then add at the footnode of β the subtree of α that we removed earlier. The result is tree γ. As we can see, adjunction can have the effect of inserting one tree into the center of another. Our linguistic example is continued in Figure 7. Tree β_1 containing the adverb is adjoined at the VP node into tree α_4. The result is tree α_5, which corresponds to sentence (3). Note that α_5 is composed of trees α_1, α_2, α_3 and β_1, each of which correspond to exactly one lexical item, in contrast to the grammar given above in (2).

A formalism in which the elementary structures of a grammar are phrase structure trees and in which the combining operations are adjunction and substitution is called a **Tree Adjoining Grammar** or TAG. (Schabes 1989) has shown that a tree composition system is only lexicalizable if the composition operations include adjunction.[7] Thus, the process of lexicalizing a CFG naturally leads to a TAG. Note that while we can lexicalize a TAG, a TAG need not be lexicalized; to refer specifically to a lexicalized TAG, the abbreviation LTAG is often used.

As we will see, TAGs are more powerful formally than CFGs, meaning

[6] Having two verbs *like*, one of which also subcategorizes for an adverb, does not solve the problem, since it does not generalize to multiple adverbs (in addition to being linguistically unappealing).

[7] (Rogers 1994) and (Schabes and Waters 1995) show that TAG with adjunction restricted in certain ways can also lexicalize a CFG. See Section 1.4.1.

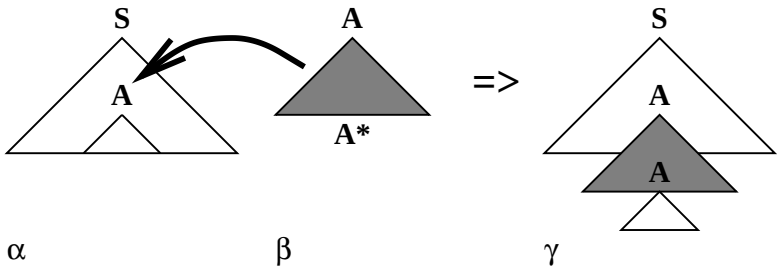

FIGURE 6 The Adjunction Operation

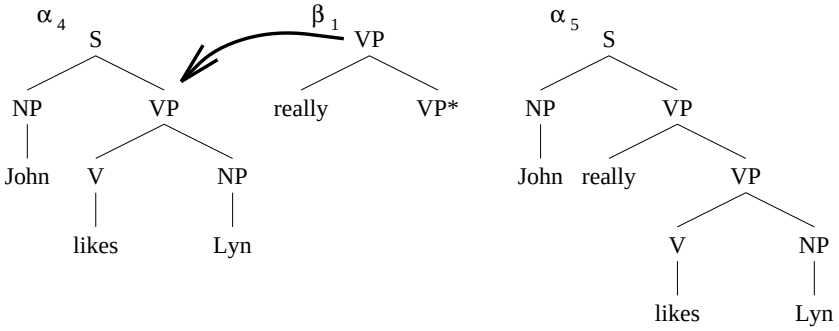

FIGURE 7 Adjunction of really into initial tree

that they can derive more complex languages than CFG. They are also more difficult to parse.

1.2.3 Derived Tree and Derivation Tree

CFG is a string-rewriting formalism (a nonterminal is replaced by a string), and therefore a CFG derives a set of strings. TAG is a tree-rewriting formalism (a nonterminal in a tree is replaced by an entire tree), and therefore it derives a phrase-structure tree, called the "derived tree". (The derived tree for our example tree α_5 in Figure 7.) For both CFG and TAG, a derivation (of a string in CFG or a tree in TAG) results in a second structure, the "derivation tree".[8] The derivation tree records how the derived string (CFG) or derived tree (TAG) was assembled from elementary rules (CFG) or elementary trees (TAG). This structure is similar for both CFG and TAG since both are context-free

[8] It is sometimes said that in a CFG, the derived tree is the same as the derivation tree. This is not strictly speaking correct. As a string-rewriting formalism, a CFG does not produce any derived tree at all, only a derived string.

rewriting systems:[9] we can perform a rewrite operation independently of the context of the symbol in the string (CFG) or tree (TAG) that we are rewriting, as long as the label matches the right-hand side of the rule (CFG) or the root node label (TAG).

We note at this point an issue in terminology that may cause confusion. In a CFG, nonterminal nodes must be rewritten in a derivation for the derivation to terminate successfully, while terminal nodes may never be rewritten. Thus, a derived string in a CFG always contains only terminal symbols. However, the derived trees in a TAG contain both terminal and nonterminal symbols. The apparent difference can easily be explained: the notions of "terminal symbol" and "nonterminal symbol" in TAG do not actually refer to tree rewriting, but are used in the same way in TAG elementary and derived trees as they appear in CFG derivation trees. In a TAG, a terminal symbol may never be rewritten, but not all nonterminal symbols must or even may be rewritten. We will return to this issue after we have introduced the notion of adjunction constraint in Section 1.2.4.

Because a TAG can be lexicalized, there is an important difference in the linguistic use of the derivation tree for CFG and TAG. Indeed, the derivation tree of a CFG derivation is a phrase-structure tree like the *derived* tree of TAG. In the derivation tree of TAG, each of the elementary trees used in the derivation is represented by a single node. If the grammar is lexicalized, we can identify this node by the name of the tree and the (base form of the) lexeme with which it is lexicalized. If a tree t_1 is substituted or adjoined into a tree t_2, then the node representing t_1 becomes a dependent of the node representing t_2 in the derivation tree. Furthermore, the arcs between nodes are annotated with the position in the "target tree" at which substitution or adjunction takes place.[10] In the TAG literature, this annotation is in the form of the tree address of the node (using a formal notation to uniquely identify nodes in trees, without reference to linguistic concepts). The derivation tree for the example derivation above is shown in Figure 8 on the left. However, in analogy to notation used in linguistic literature, we can simply assign grammatical function labels to argument positions, and introduce the convention that all other positions are attribute positions, marked as Adjunct. The

[9] In the term "context-free grammar", the "string-rewriting" is implicit since no other rewriting systems were known when CFG was first defined.

[10] In the standard definition of adjunction, only one tree may adjoin or substitute at a given node. (Schabes and Shieber 1994) propose that more than one modifier auxiliary tree (see Section 1.3.1 for an informal definition) may adjoin at a single node. This is useful since otherwise chains of modifiers are adjoined into each other, arguably not resulting in a linguistically informative derivation structure (see below).

"linguistic" derivation tree for the example derivation above is shown in Figure 8 on the right. In usual TAG notation, the node address for subject, adjunct and object are respectively 1, 2, 2.2. We can see that the derivation structure is a dependency tree which closely resembles the structures proposed in dependency grammars, such as (Mel'čuk 1988). We return to the relation between TAG and dependency grammars in Section 1.5.1.

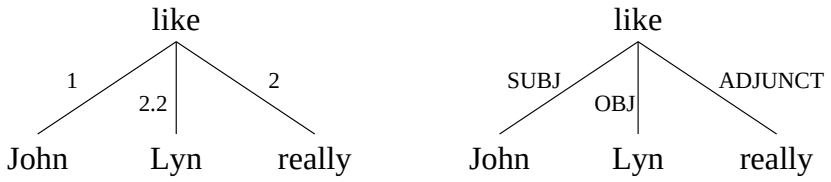

FIGURE 8 Derivation Tree for John really likes Lyn (left) and linguistic interpretation (right)

1.2.4 Adjunction Constraints and Features

Nodes in elementary trees can be annotated with *adjunction constraints*. There are three types of constraints:

- Selective adjoining constraint (SA). The SA constraint is a list of auxiliary trees which may be adjoined at the node (or a list of initial trees which may be substituted).
- Null adjoining constraint (NA). The NA constraint is a special case of the SA constraint, namely a SA constraint with an empty list: no adjunction is possible at this node.
- Obligatory adjunction constraint (OA). This constraint is a boolean which says whether or not adjunction at this node is obligatory.

A node cannot both have an NA constraint and an OA constraint. Thus, we have four options: a node has no constraint; a node has a non-obligatory SA constraint (usually written SA(β_1, ..., β_n) where β_1, ..., β_n are the trees which may be adjoined at the node); a node has an obligatory SA constraint, usually written OA(β_1, ..., β_n); or a node has an NA constraint. An example is shown in Figure 9. The initial tree α has a non-finite verb, so in order to complete the derivation of a sentence an auxiliary tree must be adjoined. Two adjunctions are possible: either the passive auxiliary *is* or the perfective auxiliary *has* (we disregard morphological variation for now).

Note that an NA node is, from the tree-rewriting perspective, a non-terminal symbol (since it cannot be rewritten), while an OA node is

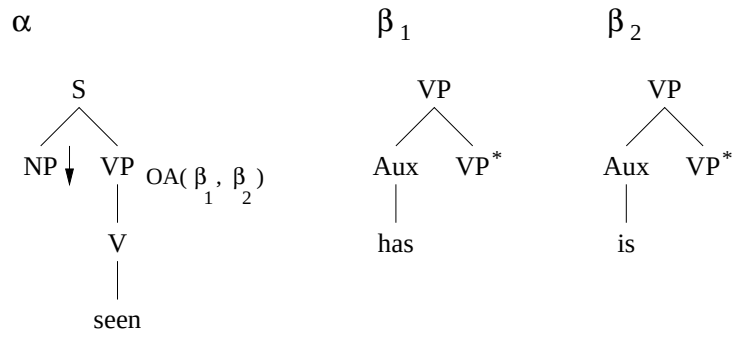

FIGURE 9 Forcing the adjunction of an auxiliary using the obligatory adjoining constraint

a nonterminal symbol for tree rewriting (it must be rewritten). Nodes marked with no constraints or with an SA constraint represent either nonterminal or terminal nodes for tree rewriting. Thus we see that adjunction constraints in TAG play the same role as the nonterminal/terminal distinction in CFG, namely that of regulating derivations.

(Vijay-Shanker 1987) proposes to annotate each node not with an adjunction constraint, but with feature structures, or more specifically, with two feature structures, called the *top feature structure* and the *bottom feature structure*. In linguistic applications, the top feature structure of a node describes (roughly speaking) the relation of the node to the tree above it, while the bottom feature structure describes the relation of the node to the subtree below it. During a derivation, when an auxiliary tree β is adjoined at a node η, the top feature structures of η and of the root of β are unified, and the bottom feature structures of η and of the footnode of β are unified. (This is shown schematically in Figure 10, where v represents unification.) If unification fails, the adjunction cannot be performed. At the end of a derivation, at all nodes all top and bottom feature structures are unified. If unification fails at at least one node, the derivation is not yet complete.

While adding feature structures to nodes increases the generative capacity of TAG greatly (TAG becomes undecidable), (Vijay-Shanker 1987) shows that if the feature structures are non-recursive (i.e., bounded in size by some constant fixed for each grammar), the resulting system has the same formal properties as TAG with adjunction constraints. In fact, the effects of adjunction constraints can be obtained with non-recursive feature structures, and *vice versa*. To see this, consider our example from earlier, now expressed using feature structures rather than

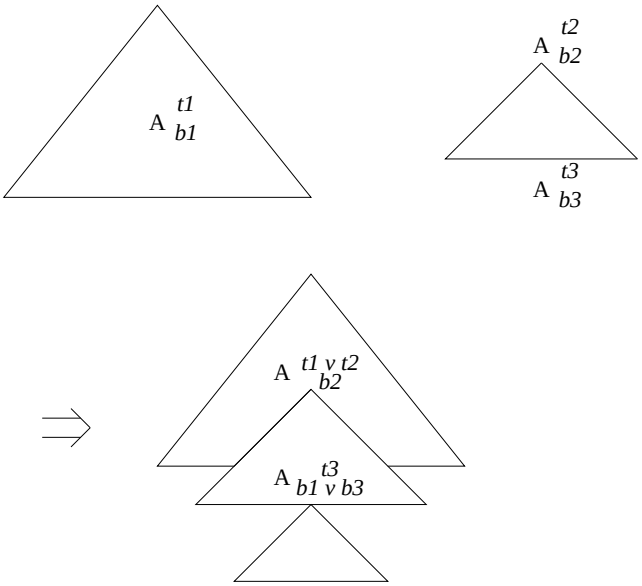

FIGURE 10 Schematic depiction of use of feature structures in adjunction

adjoining constraints (Figure 11). (The example is intended only to show how the feature structures work; we omit much linguistic detail.) The top feature structure on the VP node in tree α specifies [tense:+], reflecting the fact that above this node, the tree behaves like a tensed clause (it has a subject). The bottom feature structure on the same VP node specifies [tense:−], reflecting the fact that the tree rooted at the VP nodes does not, in fact, contain any tense element (the anchor of the tree, *seen*, is not a tensed verb form). Now, the top and bottom feature structures cannot unify, and therefore, in order to use tree α in a derivation, a tree *must* be adjoined at the VP node. We thus obtain the effect of the obligatory adjoining constraint. If we adjoin trees β_1 or β_2, the derivation can terminate: the top feature structure of the VP node of α can unify with the top feature structures of the roots of these two trees (because, in this particular grammar, the value for tense is not specified), and the bottom feature structure of the VP node can unify with the bottom feature structures of the root nodes of these two trees (again, because no value is specified), so adjunction is possible. Then (after substituting a subject at the NP node), at the end of the derivation we can unify all top and bottom feature structures. However, adjunction

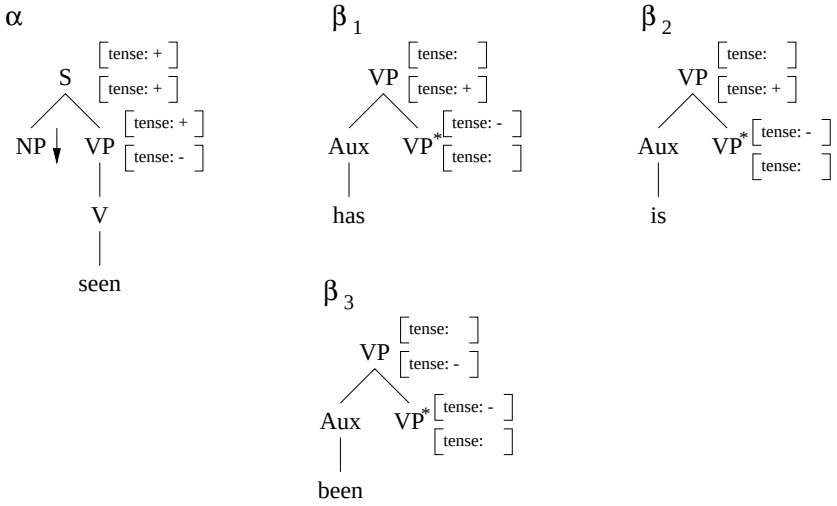

FIGURE 11 Forcing the adjunction of an auxiliary using bounded feature structures

of β_3 is not successful: while the adjunction is initially possible, at the end of the derivation we cannot unify the feature structures at the higher VP node. We thus obtain the effect of the obligatory adjoining constraint.

But note that in the case where we want to adjoin β_3, we can save the derivation by adjoining at its root node β_1, to obtain a sentence such as *Matthew has been seen*. This derivation cannot easily be obtained using adjunction constraints. Another example of the convenience of feature structures as compared to adjunction constraints can be seen from Figure 12. Here, we specify tense: [1] both in the bottom feature structure of the S node and the top feature structure of the VP node. This means that the values of the tense feature in these two feature structures must be identical (but more is not specified). Thus, we force the adjunction of the auxiliary either at the S node, or at the VP node, but disallow adjunction at both nodes. As a result, we can derive *has John seen* and *John has seen*, but not *has John has seen*, nor *John seen*. If we had only adjunction constraints, the grammar would need to include a special tree which requires adjunction at the S node as opposed to the VP node. For these reasons, non-recursive feature structures have become standard in linguistic uses of TAG.

(Vijay-Shanker 1992) observes that the feature structures, rather than the nodes themselves, are controling the derivation, and suggests

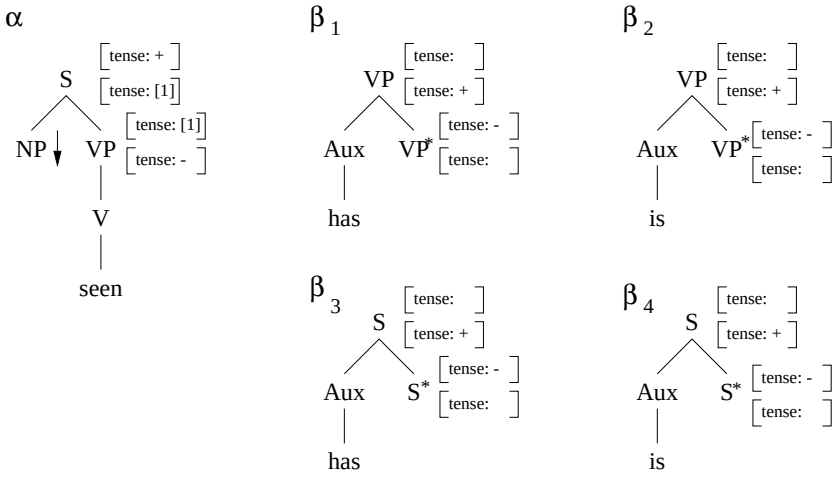

FIGURE 12 Forcing the adjunction of an auxiliary using bounded feature structures

that each node in a TAG tree should be split up into two *quasi-nodes*, each associated with exactly one feature structure, yielding *quasi-trees*. The relation between the two quasi-nodes is one of (not necessarily immediate) domination: the two quasi-nodes in a pair may coincide, or other nodes may come between them (through adjunction), but they will always be in a domination relation. D-Tree Substitution Grammar (Rambow et al. 1995) and Tree Description Grammar (Kallmeyer 1996) can be seen as a formalism based on quasi-trees rather than trees (see Section 1.4.2).

1.2.5 Formal Properties

In this section, we summarize the main formal results about TAGs. A general overview can be found in (Joshi et al. 1991).

Weak and Strong Generative Capacity

The set of languages generated by a TAG, $\mathcal{L}$(TAG), includes the set of languages generated by a context-free grammar, $\mathcal{L}$(CFG). This can easily be seen: given a context-free grammar, we create a new tree adjoining grammar by taking each context-free rewrite rule of the form $A \longrightarrow \alpha$ (where α is an arbitrary string made up of nonterminal and terminal strings) and transform it into an initial tree with root labeled A and daughter nodes labeled (from left to right) with the symbols in α. Nonterminal symbols become substitution nodes. Derivations in the thus

created TAG proceed entirely using substitution, mirroring that in the corresponding CFG exactly. However, TAG can also generate languages which a CFG cannot. To see this, consider the grammar in Figure 13. If we adjoin tree β into itself at the center node labeled A repeatedly, and then adjoin the resulting tree into tree α at the end of the derivation, we can derive any string in the language count-4 $= \{a^n b^n c^n d^n \mid n \geq 0\}$. This language, as is well known, is not in $\mathcal{L}$(CFG). (Note that the other two nodes labeled A in β are marked with the null-adjunction constraint, so no other derivations are possible. It is not known whether a TAG without null-adjunction constraints can generate count-4 .)

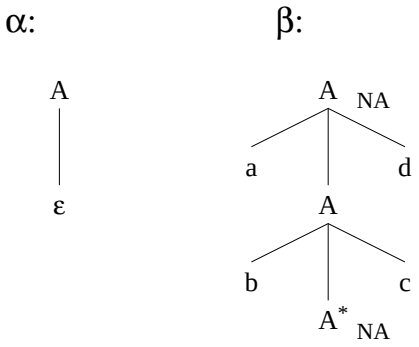

FIGURE 13 A TAG for the language count-4

The grammar in Figure 13 generates a string language that a CFG cannot generates, and thus shows that the *weak generative capacity* of TAG exceeds that of CFG. But in addition it of course also generates a tree language that a CFG cannot generate, and thus shows that the *strong generative capacity* of TAG exceeds that of CFG.[11] However, there are also tree adjoining grammars that generate string languages which a context-free grammar can generate, but that generate tree languages which no context-free grammar can generate. An example is shown in Figure 14, which generates the string language count-2 $= \{a^n b^n \mid n \geq 0\}$ which is clearly a context-free language. However, the generated tree language cannot be generated by any CFG: to generate the kind of trees generated by the grammar in Figure 14, the context-free grammar would first need to count how many a's it has generated, before generating the

[11] Terminology is a bit sloppy here: a CFG does not actually derive a tree; rather, the tree is the derivation tree for a CFG, as discussed in Section 1.2.3. We use the sloppy terminology for convenience in this section.

same number of b's. Clearly, a CFG cannot do this.

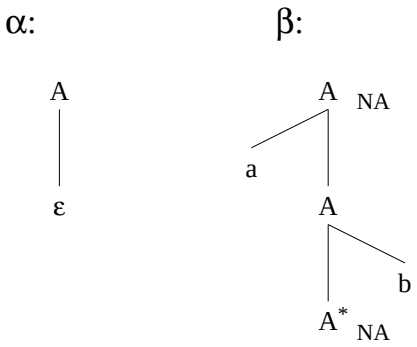

FIGURE 14 A TAG for the language count-2 which derives trees a CFG cannot derive

There are languages which are not in $\mathcal{L}$(TAG), i.e., which cannot be generated by any TAG. Among these languages is count-5 , and indeed count-n for any $n \geq 5$. This can be shown using the pumping lemma for TAG (Vijay-Shanker 1987). However, by constructing an equivalent linear bounded automaton, it is easy to show that $\mathcal{L}$(TAG) $\subset \mathcal{L}$(CSG), i.e., any language that a TAG can generate can also be generated by a context-sensitive (string-rewriting) grammar. (The inclusion is not proper because of count-5 .)

While TAG can generate tree languages which no CFG can generate, the set of derivation trees of a TAG is in fact a recognizable set (i.e., a set of trees that can be generated by a CFG). This is not surprising, as a TAG derivation is a context-free tree derivation in the same sense that a CFG derivation is a context-free string derivation. Thus, in fact, TAG and CFG have the same sets of derivation trees.

For some analyses, we are not interested in the string sets that TAG can generate, nor in the tree sets. Instead, we are interested in *how* we can derive a particular string or tree set from a given grammar. In this case, we are interested in the *derivational generative capacity* of TAG (Becker et al. 1992). Joshi et al. (in this volume) is an example of a study of TAG based on constraints on derivations rather than on the derived tree or string.

Finally, we observe that languages generated by TAG are semilinear: this means that the Parikh mapping of a tree adjoining language is a semilinear set. The Parikh mapping of a language is a set of vectors, each of which corresponds to one string in the language and records

how many times each of the terminal symbols of the grammar occur in the string. A set of vectors is semilinear if it is a finite union of linear combinations of a base set of vectors. Note that as a consequence, tree adjoining languages, like context-free languages, are letter-equivalent to regular languages (i.e., for each tree adjoining language L_1, there is a regular language L_2 such that every member of L_1 is a permutation of the terminal symbols of a member of L_2, and *vice versa*).

Formal Automata for TAG

(Vijay-Shanker 1987) defines the Embedded Pushdown Automaton (EPDA) and proves it to be formally equivalent to TAG. Like the PDA (the automaton associated with CFG), the EPDA consists of a push-down store, an input tape with a one-way read-only scanner, and a finite state automaton that controls the actions of the automaton. However, the push-down store has a more complex structure than that of a PDA: it is a stack of stacks of stack symbols, rather than a simple stack of stack symbols. In a move of the EPDA (called the WRAP move), the top element of the top stack is replaced by a finite sequence of new stack symbols, and in addition finite sequences of new stacks are inserted just below and above the top stack.

(Schabes 1990) defines a bottom-up version of the EPDA, called BEPDA. (Becker 1994) defines a different automaton which uses two stacks rather than a stack of stacks.

Parsing Complexity

The recognition problem (and the parsing problem) for TAG can be solved in polynomial time; in fact, the time complexity is in $O(n^6)$. To see this, the standard bottom-up chart parser for CFG can be extended to TAG (Vijay-Shanker 1987). Entries are stored in a four-dimensional table to account for the fact that partial derivations may include a foot-node; hence, not only are two indices needed to record the beginning and the end of the string, but also two indices are needed to record the beginning and the end of the string which is dominated by the footnode. For a discussion of parsing, see Section 1.6.2.

Equivalence to Other Formalisms

As was shown in (Vijay-Shanker and Weir 1994),[12] TAG is weakly equivalent to three other formalisms: Linear Index Grammar (LIG), Head Grammar (HG), and Combinatory-Categorial Grammar (CCG). Interestingly, all three formalisms and TAG had been defined independently of each other, but all four had been inspired directly from the needs of

[12] Some of the results had been published in previous publications by the same authors. The inclusion of $\mathcal{L}$(TAG) in $\mathcal{L}$(HG) was first shown in (Seki et al. 1991).

describing natural language syntax. We briefly describe the three formalisms and the equivalence result.

In a linear index grammar (LIG) (Gazdar 1988), a context-free grammar is augmented with a stack of special stack symbols. At each rewrite step, one or more stack symbols may be removed from the top of the stack, one or more stack symbols may be added to the top of the stack, and the entire stack may be copied to at most one of the newly introduced nonterminal symbols.

A head grammar, defined by (Pollard 1984), is a string-rewriting formalism in which, in addition to the simple juxtaposition provided by context-free grammars, strings can be wrapped around other strings in a rewrite operation. This operation of wrapping directly mirrors the effect of adjunction.

A combinatory-categorial grammar (Steedman 1990) augments the standard categorial grammar (which is weakly equivalent to CFG) with a stack of categories that can be associated with a category during a derivation and operations that allow for the manipulation of this stack. This stack is similar to the stack of an LIG.

Pitsch (in this volume) relates TAG to a version of Hypergraph Grammar.

Mild Context-Sensitivity

The notion of a *mildly context-sensitive* language was first introduced by (Joshi 1985). The goal is specifically to describe the class of formal languages needed (or well suited) for the description of natural language syntax, and as a result, the criteria are not all simple formal language criteria:

- The class of languages includes all context-free languages.
- The languages in the class are polynomially parsable.
- The languages in the class capture only certain types of dependency, such as nested or cross-serial dependencies, but perhaps not those exhibited by the mix languages (in which the strings are composed of the same number of a fixed number of different terminal symbols).
- The languages in the class have the *constant-growth property*, i.e., if all the strings in the language are arranged by size, then the difference between two consecutive lengths is always bounded by some constant (for that language). Note that the constant growth property is a consequence of the stronger requirement of semilinearity.

A formalism which generates only mildly context-sensitive languages is also called mildly context-sensitive. TAG is mildly context-sensitive in this sense.

1.3 Natural Language Syntax in TAG

TAG is a formalism and not a linguistic theory: TAG can be used to model several linguistic theories (see below), and several linguistic analyses are usually possible in TAG for a given phenomenon. However, the original properties of TAG (its extended domain of locality and its lexicalization) have led to original analyses of some linguistic phenomena, independently of a particular linguistic theory. This section is devoted to the "TAG way" of dealing with lexical categories, extraction phenomena, raising constructions and idioms. Intuitively speaking, TAGs enable one to conjoin the merits of a constituency-based and a dependency-based approach to syntax: the derived tree represents the constituent structure while the derivation tree is closer to a dependency representation. Most of the following analyses exploit this duality of TAG representations.

1.3.1 Supertags

It is well known that parts of speech tend to represent distributional classes that are needed to describe syntactic constructions and generalizations. It is also well known that they are not sufficient since different items with the same part of speech may exhibit quite different syntactic behaviors: compare the English verbs *eat* and *think*, the adverbs *probably* and *very*, the adjectives *other* and *asleep*. Subclassifications are usually added (such as transitive for verbs or predicative for adjectives) but ultimately one needs a complete description of the syntactic context called for by each item (as sketched by (Chomsky 1965) with the notion of subcategorization). It is one of the merits of the TAG formalism to associate such complete syntactic descriptions with each lexical item. In this respect, while parts of speech represent the tags associated with each word of a language, TAG elementary trees can be seen as the "supertags" associated with them (Srinivas and Joshi 1999).

While parts of speech are usually hybrid objects (coding some semantics, and morphology on top of syntax), TAG supertags represent more genuine syntactic classes. Examples of supertags for the previously mentioned English items are as in Figure 15.

The adverb *very* only premodifies an adjective, *other* is only used as a prenominal attributive adjective (cf *the other persons*, i.e. with a right N rooted auxiliary tree), the adjective *asleep* is mainly used as a predicative complement (cf *John was asleep*, i.e. with an initial A rooted initial tree). The verb *eat* is used transitively (with an NP node for

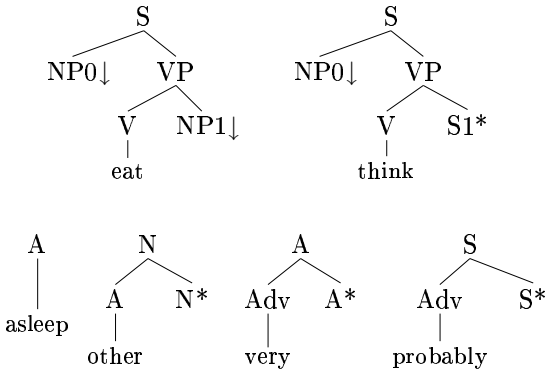

FIGURE 15 Some elementary trees (=supertags) for verbs, adverbs and adjectives

its subject and one for its object) while the verb *think* is used with a sentential complement (with an S footnode); see below for the motivation for using an auxiliary tree for verbs with a sentential complement.

Linguistic assumptions about the well-formedness of elementary trees are usually made in order to constrain the type and the topology of the elementary trees associated with each lexical item. Some such assumptions that are standardly made are the following:[13]

- An elementary tree is the maximal syntactic projection of a lexical item, with the possible addition of functional projections (the "extended projection" of (Grimshaw 1991)) and an additional level for adjuncts which determines at what nodes they can adjoin (and in what direction).
- Auxiliary trees are used for modifiers, functional categories (determiners, auxiliaries, ...), predicates with verbal complements and raising predicates.
- A predicative lexical item has (substitution or foot) nodes for each of its subcategorized arguments in its elementary trees (Predicate Argument Cooccurrence Principle or PACP). If a predicative lexical item anchors an auxiliary tree, the auxiliary tree is referred to as a *predicative auxiliary tree.*
- An adjunct has a foot node for the category it modifies (usually its semantic argument) in its elementary tree. Auxiliary trees anchored by adjuncts are referred to as *modifier auxiliary trees.*

[13] Some work challenges one or more of these traditional assumptions, for example (Lee 1993) or (Rambow and Lee 1994) for the adjoined argument hypothesis.

- An elementary tree is associated with some atomic (i.e., non-compositional) semantic meaning.

Standard assumption are also made about the number of elementary trees associated with a given predicate, and about their grouping:

- Predicates are associated with different elementary trees corresponding to their different transitivity alternations (passive, middle), the different realizations of their arguments, and the different types of clause they can anchor (relative, interrogative, ...).
- The elementary trees possibly associated with predicates with a given subcategorization frame are gathered into tree families, The number of elementary trees usually associated with a lexical item comprises a certain amount of redundancy and solutions both theoretical and practical (for grammar development) have been proposed to overcome this (see below, 1.6).

1.3.2 Extraction

The interest of using TAG in analyzing extraction phenomena was first demonstrated in the seminal paper by (Kroch and Joshi 1985). The extended domain of locality of TAG enables us to localize the filler-gap dependency (or filler-head dependency) inside an elementary tree. The relation between the extracted argument and the corresponding gap (or the predicate that subcategorizes for it) is thus kept local and no special device (movement or unbounded feature percolation) is needed. The apparent long distance dependency is captured by the adjunction of the matrix clause between the filler and the gap (or its predicate). Let us take the example of the English interrogative clause:

(5) Which book do you know that Max read?

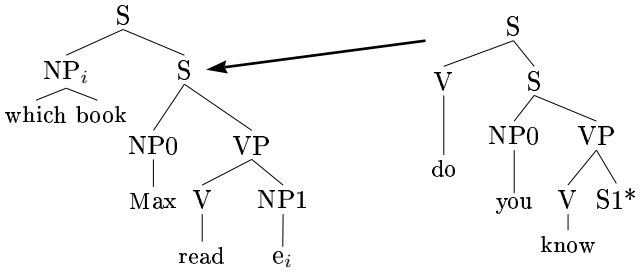

FIGURE 16 Adjunction of a matrix clause

The interrogative clause anchored by *read* is represented by an extended elementary tree and a node for the fronted *wh*-complement. The

matrix clause *do you know* is represented by an auxiliary tree adjoining at the S internal node. The derivation tree exhibits the correct dependency between *which book* and *read*, as well as between *Max* and *read*, as can be seen in Figure 17 on the bottom.

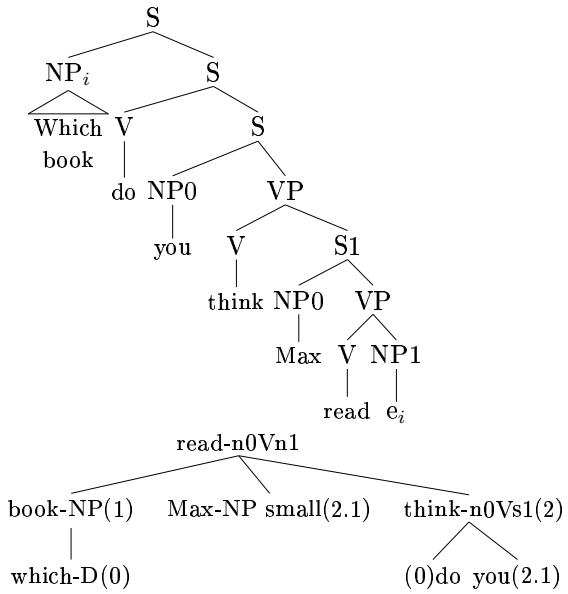

FIGURE 17 Derived tree (above) and derivation tree (below)

One may notice that, as in modern versions of HPSG or LFG, no trace is needed and the same representation without an empty NP would be as satisfactory as the previous one from the TAG point of view. An advantage of this analysis, as pointed out by (Kroch and Joshi 1985, Kroch 1987), is that no additional constraint on movement (or feature percolation) is needed to handle syntactic constraints on extraction (or island constraints). They can be represented as constraints on the topology of elementary trees. To rule out *wh*-fronting out of *wh*-islands in English for example, one rules out elementary trees with several fronted nodes for arguments (marked for the *wh*-feature). The constraints will be different for languages allowing for such phenomena. For example, in Romanian we can have multiple *wh*-fronting in one clause. Thus, we would not be able to rule out *wh*-movement from *wh*-islands in the same way that we do in English — but crucially, Romanian does allow *wh*-movement from *wh*-islands, and there is nothing we need to rule out!

1.3.3 Raising Constructions

The relevance of TAG to raising constructions was also demonstrated by (Kroch and Joshi 1985). It essentially draws upon the GB analysis (the raised argument is not subcategorized for by the raising verb) while avoiding the difficulties of a deep structure with a S argument (difficulties pointed out by (Bresnan 1982) among others). The main intuition is that raising verbs are adjoined to S-rooted elementary trees projected by the infinitival (or gerund) verb. An example is shown in Figure 18, with the raising verb *can* adjoining to the VP node in the elementary tree anchored by *swim*.

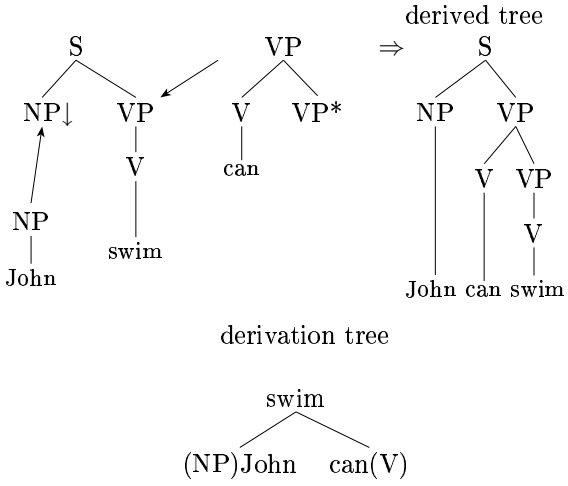

FIGURE 18 Raising construction

In the derivation tree, it is clear that there is no dependency between the raised argument (here the subject) and the raising predicate (here *can*). The dependency relation between the raised subject and the bare infinitive is thus kept local. One thus avoids the difficulties of the mapping between syntactic argument and semantic arguments (raising verbs being special cases of verbs not assigning semantic roles to all their subcategorized argument) found in LFG or HPSG (cf Pollard and Sag 1994's raising principle). Agreement is handled by features which can be shared between the subject node and the upper part of the VP node (automatically updated in case of an adjunction of a raising verb). For English auxiliaries, which can be analyzed as a special form of raising

verbs,[14] alternative adjunction to the S root node is also allowed (for *Did John arrive on time?*). For other languages, further distinctions may be made between raising verbs (see (Abeillé 1991) for French).

For cases of object raising, one can distinguish *tough* constructions, and exceptional case marking (ECM) verbs. For *tough* constructions, one can define an auxiliary tree for adjoining the *tough* adjective, which takes the infinitive as its argument (Abeillé 1991).

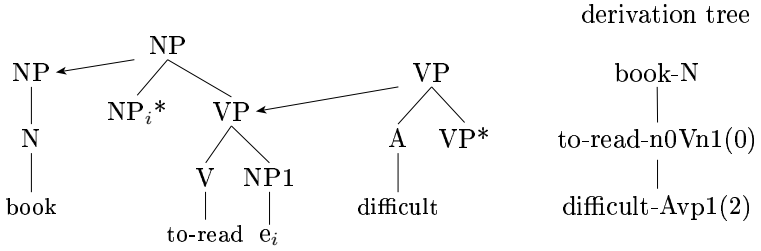

FIGURE 19 Tough adjectives

Non-*tough* adjectives (like *possible*), on the other hand, only adjoin to N. One thus avoids the idiosyncrasies associated with the traditional *tough* movement:

(6) a. It is possible / easy to fire John
b. John is * possible / easy to fire

For ECM verbs, one has to resort to the small clause analysis: the ECM takes a sentential complement, with the "raised" argument in the same elementary tree as the infinitival (as is done in the XTAG grammar (XTAG-Group 1999)).[15]

There is thus no direct link with the passive that some ECM verbs allow for:

(7) a. They expected John to arrive any minute
b. John was expected to arrive any minute

In the XTAG grammar, the passive of ECM verbs is analyzed as a reduced subject raising construction (with the same auxiliary tree as

[14] Auxiliaries are typically distinguished from raising verbs by assuming that the former do not have any arguments of their own and simply modify the semantic content of the verbal head they are adjoined into. This is not true for true raising verbs, which in addition to the verbal head they adjoin into can also have additional arguments: *John seems to all of us to have left.* Such sentences can be derived as shown in Figure 18.

[15] For an analysis of ECM verbs in analogy to raising-to-subject verbs (using an extension of TAG), see (Kroch and Rambow 1994).

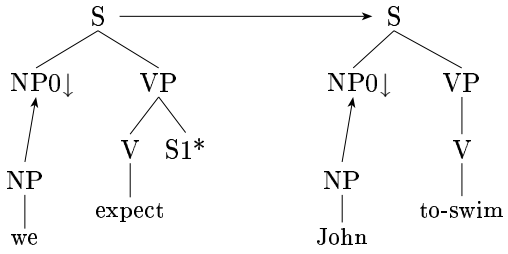

FIGURE 20 Derivation of ECM construction

seem).

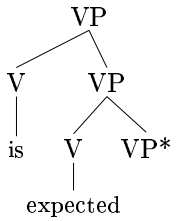

FIGURE 21 Auxiliary tree for the passive of an ECM verb

The same holds for copular constructions, which are analyzed as raising constructions, headed by the predicative complement, with the copular verb adjoining between the subject and the predicative anchor (cf (XTAG-Group 1999)).

In related work, (Harley and Kulick 1998) discuss the issue of raising verbs in VSO languages such as Welsh, which appears to require multi-component TAG (see Section 1.4.2).

1.3.4 Idioms

Non compositional or semi-compositional phenomena are always a challenge for formal models. LTAG offers an interesting way of representing idiomatic expressions which are non compositional from the semantic point of view but follow regular syntactic rules. Expressions like *to pull NP's leg* or *to take advantage of NP* need to be decomposed syntactically (to explain the associated passive or relative clause) but not semantically.[16] Abeillé and Schabes (1991, 1996), Abeillé (1995) proposes to associate with such expressions multi-anchored extended elementary trees such as those in Figure 22.

[16] Some authors have challenged the view that all idioms are non compositional. The proposed TAG representation is especially suited for those that are not.

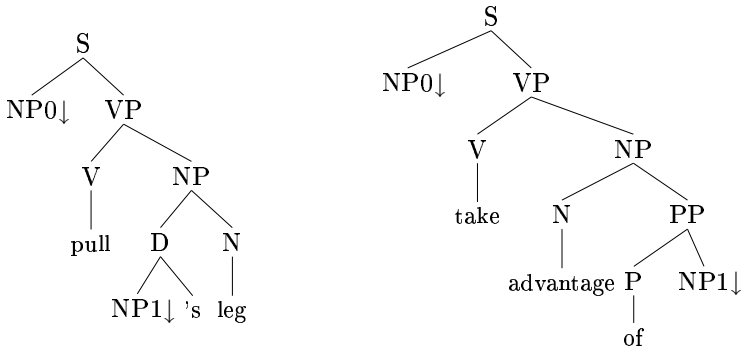

FIGURE 22 Elementary trees for idiomatic expressions

On the one hand, their regular syntactic topology make them regular candidates for most syntactic rules, on the other hand their being elementary trees makes them semantic atoms. The derivation tree shows this semantic unity, and ambiguous expressions with either a literal or an idiomatic interpretations (like *to pull one's leg*) can be disambiguated only if one looks at the associated derivation tree (Figure 23).

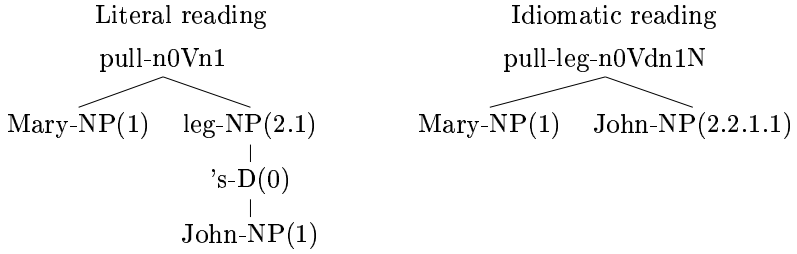

FIGURE 23 Two derivations trees for Mary pull John's leg

The literal reading correspond to the derivation starting from the transitive tree for the verb *pull* which has two arguments: *Mary* and *leg*, both substituted. The idiomatic reading correspond to a more compact derivation tree,starting with the multi-anchored idiomatic tree with only two arguments being substituted: *Mary* and *John*. It is worth noting that the associated derived trees are the same.

1.3.5 Cross-Serial Dependencies in Dutch

As in other verb-final languages such as German, embedded clauses in Dutch can occur before the (clause-final) verb in a recursively embedded

construction. However, the order of the verbs in the two languages differ: while in German, the dependencies between the verbs and their arguments are nested, they are cross-serial in Dutch.[17] Consider the following sentence:[18]

(8) ... omdat Wim Jan Marie de kinderen zag helpen
... because Wim Jan Marie the children saw to help
leren zwemmen
to teach to swim
... because Wim saw Jan help Marie teach the children to swim

The LTAG analysis in Figure 24 is based on those proposed in (Joshi 1987a, Kroch and Santorini 1991). The main verb of each clause is "raised", an analysis proposed independently of the TAG analysis in the GB literature. We then adjoin each clause into its immediately dependent clause at the S node immediately below the root node. This "pushes" both verbs away from their nominal arguments, even though they originate in the same elementary structure. The order of the verbs in the final sentence simply follows from the way the elementary structures are adjoined; no global word-order rules are necessary.

Note that, as in the case of long-distance *wh*-extraction discussed above, the correct word order falls out from the phrase structure specified in the elementary trees projected from lexical items; no system for relating trees or parts of trees beyond the formal operations of adjunction and substitution are needed.

1.3.6 Other Syntactic Issues

Many other syntactic issues have been studies in the extended TAG framework. We mention only some. (Heycock 1987) studies the Japanese causative. (Rambow and Lee 1994) and (Rambow 1994a) discuss scrambling in German and Korean. Hockey and Mateyak (in this volume) use a system of features to account for determiners in English. Finally, Mahootian and Santorini (in this volume) use TAG as a model for the grammar of bilingual speakers, and show how TAG makes correct predictions about intra-sentential code-switching.

1.3.7 Semantics

With idiomatic expressions, one sees that TAG syntactic analyses may lead to semantic-like representations: the derivation trees represent

[17] (Shieber 1985) used similar data from Swiss German to show that CFG is not powerful enough to derive these constructions. It is therefore particularly interesting that we can give an simple account in LTAG.

[18] We would like to thank Hotze Rullmann and Marc Verhagen for helping us with this example.

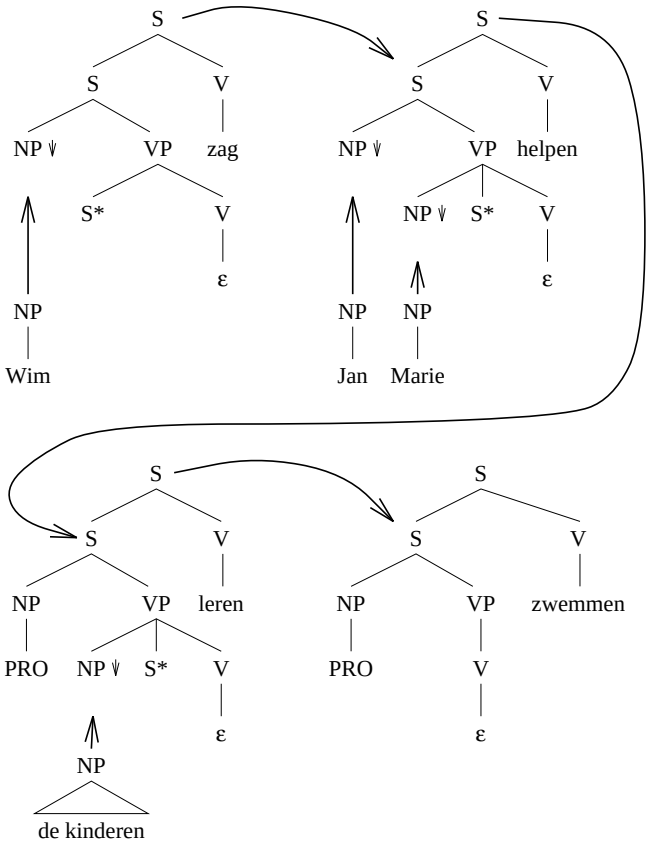

FIGURE 24 TAG Derivation for Sentence (8)

predicate-argument structures. Whether TAG derivation trees represent "deep" syntactic relations or semantic relations is a matter of debate (Rambow and Joshi 1994a, Candito and Kahane 1998). Also see Section 1.5.1 for a related discussion of TAG and dependency grammars.

The interest of LTAGs for underspecified semantic representations has been stressed by (Joshi and Vijay-Shanker 1999, Kallmeyer and Joshi 1999) and for lexical semantics by (Palmer and Rosenzweig, 1996, Palmer et al., this volume).

TAG has also been used for more detailed semantic analysis of quantifier scope (Shieber and Schabes 1990, Rambow 1994b, Rambow and Satta 1996) and of connectors and their discourse function (Jayez and Rossari, in this volume, Danlos, in this volume, Webber and Joshi, 1998,

Gardent and Webber, 1998). These studies often use an extended version of the formalism (notably synchronous TAGs, see Section 1.4.4).

1.4 Variants of TAGs

We have been presenting so far the "classical" version of tree adjoining grammars. Several variants have been proposed: some have a restrictive generative power (and are most useful to improve the efficiency of implementations), some have an extended generative power and are used to increase the linguistic expressivity of the basic TAG formalism.

1.4.1 Constraining the Power of TAGs

Chomsky and Schutzenberger (1963) argue that regular languages may be sufficient to model speaker's performance. More recently, some authors have proposed to use restrictive formalisms (equivalent to regular languages or to a subclass of context-free languages) for natural language processing (Gross 1989, Roche and Schabes 1997). Restrictive variants have been defined for TAG as well. Schabes and Waters (1995) propose a variant equivalent to Context-Free Grammar which they have called Tree Insertion Grammar or TIG. TIGs use both substitution and adjunction, with some constraints on the latter. Schabes and Waters distinguish between three formal types of auxiliary trees: left auxiliary trees (with the foot node as the leftmost frontier node); right auxiliary trees (with the foot node as the rightmost frontier node); and wrapping auxiliary trees (with the footnode surrounding by other frontier nodes). While the three types are allowed in standard TAGs, only the first two are in TIG. So the elementary trees in a TIG are either initial trees, left auxiliary trees or right auxiliary trees. Further constraints in TIG are the following, so it is not possible to derive a wrapping auxiliary tree: a left auxiliary tree cannot adjoin to the spine of a right auxiliary tree, and a right auxiliary tree cannot adjoin to the spine of a right auxiliary tree. Although this model has not been used for parsing, nor to build any sizable grammar, it may be motivated from the linguistic point of view. TAG grammars for natural languages make little use of wrapping auxiliary trees. An example of such an elementary tree would be that of a verb with a sentential complement and a second prepositional complement: the foot node for the S complement (in case of an extraction) would be surrounded by the verbal head and the second complement, as in:

(9) What did Peter say that Mary loves to his parents?

But such sentences are not very natural, and preposing the second complement is usually prefered, as pointed out by (Kuno 1973) who

proposes a no-internal-gap constraint:

(10) What did Peter say to his parents that Mary loves?

The restriction on not combining auxiliary trees of different types is more artificial, given the standard definition of adjunction (where multiple adjuncts of the same head are supposed to adjoin to one another's root). It is only viable given the revised definition of derivation of Schabes and Shieber 1994, which Schabes and Waters retain for TIG.

(Rogers 1994) proposes an alternative way of lexicalizing CFG, which he calls Regular-Form TAG. A TAG is in regular form if the following condition holds: whenever an auxiliary tree is derivable that has on its spine a node η (other than root and foot node) that has the same label as the root and foot node, then the grammar can also derive the auxiliary tree that consists of the original tree but with the part below η removed (i.e., η is now the footnode). This condition is a closure condition on the set of elementary trees of the grammar and is thus more difficult to verify than the conditions of TIG. However, Rogers shows that Regular Form is decidable. Regular form serves as a normal form for TAGs which derive local sets (just as regular grammars serve as a normal form for CFGs which derive regular languages). Regular-Form TAG is potentially appealing since it is less restrictive than TIG, and in particular the linguistic cases that are troublesome for TIG do not, in themselves, pose a problem for Regular-Form TAG.

Frank (in this volume) offers a regular version of TAGs, using only substitution and the Kleene star operation, which, he argues, is suitable for modeling child grammar at a stage where no proper subordination is used (only coordination or juxtaposition). Harbusch (in this volume) defines related schema-tag type formalisms based on TAG.

1.4.2 Extending the Power of TAG

While for many purposes the full power of TAG is not needed, there are constructions in different languages for which TAG cannot provide a satisfactory analysis, and extensions to TAG are needed to capture them. An example of such a construction is picture-NP extraction:

(11) This building, John bought a picture of.

Here, *this building* is the (prepositional) complement of the noun *picture*, and therefore the NP node into which it is substituted should originate in the same elementary tree as the head noun *picture*.[19] How-

[19] Here, we are analyzing *picture* as a noun that takes a complement noun introduced with a preposition, similar to the cases shown in Section 1.3.4. Instead, we could also analyze the construction by saying that *of this building* is an adjunct prepositional phrase to the noun *picture*. This is a linguistic choice, not a formal one,

ever, then we cannot derive the sentence, since in that case we would have to adjoin the tree for *bought* into the tree for *a picture of*. This is impossible, however, since the footnode on the tree for *bought* would have to be labeled NP while the root would need to be labeled S, as shown in Figure 25. But TAG requires root and foot nodes to have the same label, so the tree shown in Figure 25 is not a valid auxiliary tree.

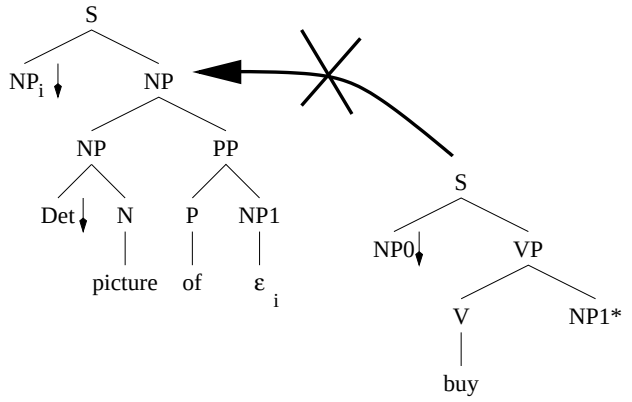

FIGURE 25 Impossible extraction from Picture-NP in TAG: illegal auxiliary tree (right)

In order to account for cases such as these, (Weir 1988) proposes several extensions to TAG called "multi-component TAG" or MC-TAG. These extensions have in common that the elementary structures are no longer trees, but sets of trees. In *tree-local multicomponent TAG*,[20] all members of an elementary set must adjoin simultaneously into a single elementary tree. In *set-local multicomponent TAG*,[21] all members of a derived set of trees must adjoin simultaneously into trees from a single elementary set.[22] As an example, consider the solution to the derivation of sentence (11), shown in Figure 26. Here, we have split up the tree for *picture of* into two trees and grouped them in one set. We adjoin the extracted NP at the root of the tree for *bought*, and substitute the

and we could make the same formal point using either analysis.

[20] The term "tree-local" was proposed by Stuart Shieber and corresponds to "Type 1" in (Weir 1988, p.31ff).

[21] Weir's "Type 2".

[22] For practical purposes, this definition must be extended to allow for substitution as well. Furthermore, we need to be able to define dominance constraints between the components of a tree set — see below.

other part containing the noun and the preposition. Note that we have adjoined and substituted the two parts of the tree set for *picture of* into the same tree: this is a tree-local multicomponent TAG derivation.

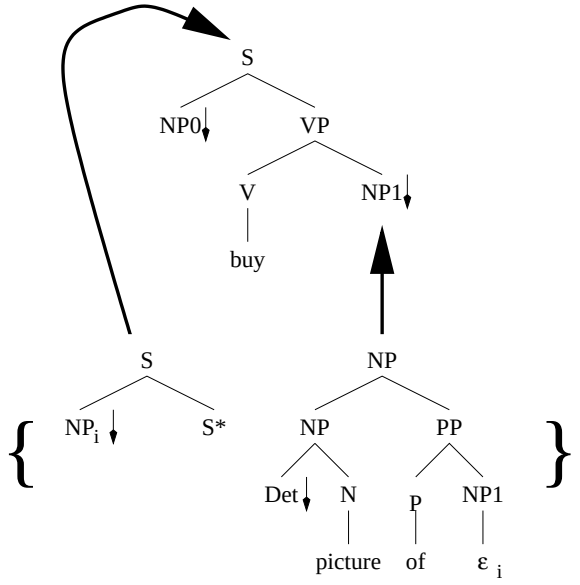

FIGURE 26 Extraction from Picture-NP in tree-local multicomponent TAG

The derivation remains tree-local for more complex cases of extraction:

(12) a. This building, John bought a copy of picture of
b. This building, John bought a copy of a facsimile of a picture of
c. ...

These sentences can be obtained by adjoining a structure such as that shown in Figure 27. This tree is adjoined to the root node of the initial tree for *picture of* (the second tree in the set in Figure 26), prior to the adjunction and substitution of the two trees from that tree set into the initial tree for *bought*. The derivation remains tree-local, since all components of the set are combined with the same elementary tree. Clearly, in English, the adjunction of trees such as that in Figure 27 can be iterated, yielding the sentences in (12).

Indeed, a vast majority of linguistic constructions that cannot be derived in TAG can be derived in tree-local MC-TAG. For a phenomenon

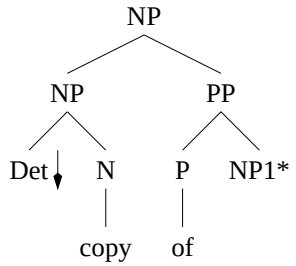

FIGURE 27 Auxiliary tree for copy of

that may require set-local MC-TAG, see the discussion by Bleam (in this volume) of clitic climbing in Romance.

Both tree-local MC-TAG and set-local MC-TAG preserve a notion of **locality in derivation**. By this term, we mean the fact that elementary structures combine in certain restricted ways which ensure that nodes from one elementary structure are related simultaneously to nodes from a single second elementary structure. (Clearly, in simple TAG as in CFG, there is locality in derivation simply because the elementary structures consists of coherent pieces of structure.) Because of the locality of derivation, derivations in tree-local and in set-local MC-TAG can always be represented as a tree, and the set of derivation trees can be generated by a CFG. (Vijay-Shanker et al. 1987) generalize this notion to the class of Linear Context-Free Rewriting Systems (LCFRS), which are equivalent to the independently defined multiple context-free grammar (MCFG) (Seki et al. 1991). These formalisms generate only languages which are semilinear, context-sensitive, and polynomially parsable, as do TAGs. In fact, it can be shown that tree-local MC-TAG is weakly equivalent to TAG (i.e., it generates the same string languages as TAG), while set-local MC-TAG is weakly equivalent to LCFRS/MCFG.

We can also define multicomponent versions of TAG in which locality of derivation is abandoned. Non-local MC-TAG (Weir's Type 3) is such a non-local MC-TAG: here, all members of one set must be adjoined simultaneously, but there is no restriction on where the adjunction takes place. Non-local MC-TAG generates non-semi linear languages, and they generate languages that are NP-complete.[23] If the requirement that all members of a set be adjoined *simultaneously* is abandoned along with locality in derivation, one obtains V-TAG (Rambow 1994a). V-TAG,

[23] These results follow from results in (Dassow and Păun 1989, p.26) and (Dahlhaus and Warmuth 1986). See (Rambow 1994a) for a complete discussion.

when lexicalized, is polynomially parsable and generates only semi-linear languages. (Rambow and Lee 1994) use V-TAG to show how to derive sentences in the free constituent-order languages German and Korean.

V-TAG also formalizes the notion that the adjunction of different elements of a set of trees can be restricted by dominance constraints. Such dominance constraints can be used to impose c-command restrictions between a displaced constituent and its trace (i.e., its usual position) or its lexical governor. For example, in Figure 26, we would want a dominance constraint between the two trees in the set, specifically, between the foot node of the auxiliary tree rooted in S and the root node of the initial tree rooted in NP.

D-Tree Substitution Grammar (DSG)[24] (Rambow et al. 1995) and Tree Description Grammar (TDG) (Kallmeyer 1996) use a notion of underspecified dominance (related to the dominance constraint of V-TAG) in order to define a formalism whose elementary structures are not trees, but tree descriptions, or underspecified trees. They are combined using a generalized form of substitution (and, in some versions, sister-adjunction for linguistic adjuncts).

1.4.3 Relaxing Precedence

In many different frameworks, it is common to allow for the linear order of sister constituents to be specified (and underspecified) by separate statements of linear precedence (LP statements), rather than specify both domination and linear precedence at the same time as in the case in TAG. A variant of TAG that allows for a separate statement of LP rules, called LD/LP-TAG (Linear Dominance/Linear Precedence TAG), was first proposed by (Joshi 1987b). (Becker et al. 1991) introduce a slight variant, called FO-TAG (Free-Order TAG), and show how FO-TAG can be used to handle German word order variation. A FO-TAG grammar consists of a set of elementary structures. Each elementary structure is a pair consisting of a linear dominance (LD) structure (i.e., an unordered tree) and corresponding linear precedence (LP) rules. The LD structure (which will, imprecisely, be referred to as a "tree" here) is either an initial or an auxiliary tree. The LP rules may relate any two nodes of the tree unless one linearly dominates the other. As a result, we can have "tangled" trees such as the one shown in Figure 28. However, these precedence rules can only be stated with respect to the nodes of a single elementary structure; it is not possible to relate nodes in different structures. In determining a possible surface order for a FO-TAG tree, we must follow the *Consistency Condition*, which states that if two nodes are

[24] DSG is also called D-Tree Grammar (DTG) in earlier papers.

ordered by an LP constraint, then all pairs of nodes linearly dominated by the two nodes inherit the LP constraint. A sample elementary tree is shown in Figure 28 for the German verb *versprechen* 'to promise'. (The bar separates the specification of the ID structure on the left from the LP rules on the right.)

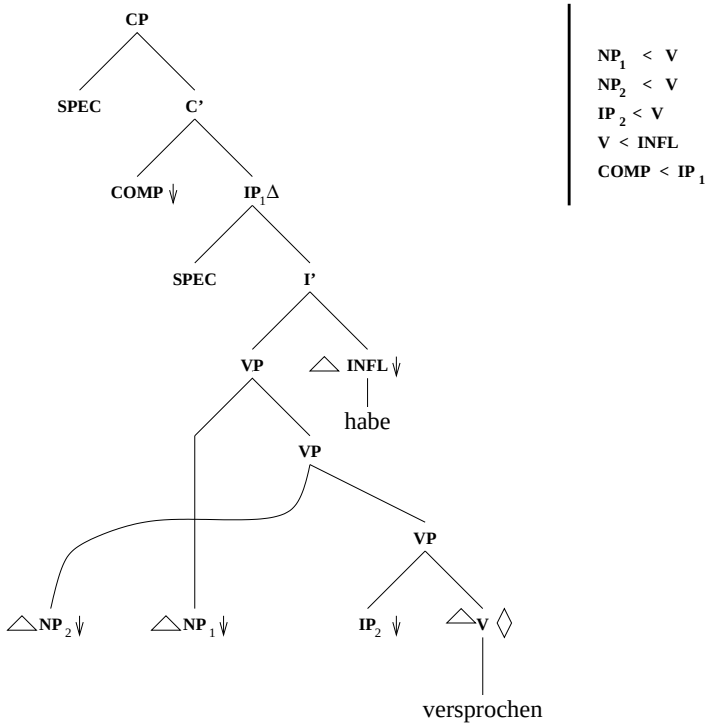

FIGURE 28 Sample FO-TAG elementary tree with integrity constraint and tangled arcs

Note that if a node η_1 at which adjunction or substitution takes place is unordered with respect to some other node η_2 in its own elementary tree, then any two nodes in the tree adjoined or substituted at η_1 are *individually* unordered with respect to η_2, meaning that one node in the adjoined or substituted tree could precede η_2 and the other could follow η_2. This effect, "indeterminacy", is overridden by the "integrity constraint", written as Δ. If we have ΔX for some node X, then any node not in the subtree rooted at X must either precede or follow every

node in the subtree rooted in X.

Because of the integrity constraints on the nodes labeled NP_3, the strings dominated by this nodes will always remain "intact" and no other string can be interposed between the determiner *dem* and the noun *Kunden*.

Adjunction and substitution are defined as in the case of regular TAG, except that the linear precedence rules of the derived tree must be stated separately. The set of LP rules of the derived tree is the union of the sets of LP rules of the two original trees. (In the case of adjunction, copies of those rules affecting the node at which adjunction takes place must be added to reflect the "splitting" of that node.) Furthermore, because of the Consistency Condition, any LP rule that relates the node at which adjunction or substitution takes places is inherited by all nodes in the adjoined or substituted tree.

(Poller 1994) proposes a simplified version of LD/LP-TAG called LD/TLP-TAG, in which only sister nodes may be ordered through LP rules. Thus, LD/TLP-TAG does not allow tangled arcs, as do LD/LP-TAG and FO-TAG. LD/TLP-TAG necessarily obeys the Consistency Condition.

While LD/TLP-TAG is formally equivalent to TAG, little is known about the formal properties of LD/LP-TAG and FO-TAG. (Poller 1994) also defines a parser for LD/TLP-TAG, while (Minnen 1994) proposes a parser for a version of LD/TLP-TAG which allows tangled arcs, but only within elementary trees. Both parsers have the same time-complexity as the Earley-style parser of (Schabes 1990) for TAG, namely $O(n^6)$ in the worst case.

1.4.4 Synchronous TAG

In synchronous TAG, two independent TAG grammars are "synchronized" by pairing trees from the two grammars. The pairing relation need not be bijective. In each pair of trees, nodes from one tree are furthermore related to nodes from another tree (again, not necessarily in a bijection). During a derivation, two trees are derived in parallel, one using only trees from the first grammar, the other using only trees from the second grammar. The derivation starts with two paired initial trees. When adjunction or substitution happens at a node in one tree, and this node is related to a node in the paired tree, then adjunction or substitution must happen in the other tree at a related node as well, and furthermore, the two trees being adjoined or substituted must be paired.[25] This basic principle is shown in Figure 29.

[25]Similar systems have been defined for other formalisms as well, for example *finite state transducers* (automata for paired regular languages which are widely used

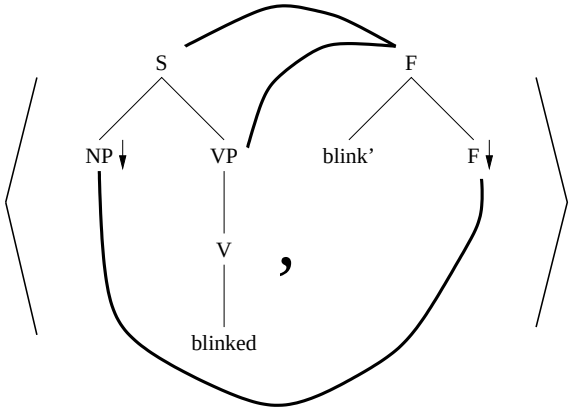

FIGURE 29 Synchronization of a syntactic tree (left) with a tree representing its semantics (right); both the VP node and the S node are linked to the same node in the semantics

An important question is what happens to "unused" links. The question arises in cases in which several links impinge on one node, as is the case in Figure 29. Only one of these links is used up in an adjunction or substitution. In the first definition for synchronous TAG, (Shieber and Schabes 1990) allow these unused links to be transferred to the tree that is being adjoined or substituted. They use this mechanism to derive different orderings of raised quantifiers (see Section 1.3.7). However, as (Shieber 1994) shows (using adjunction constraints), this synchronization increases the generative capacity of each of the synchronized grammars (i.e., the synchronization does not have the *weak language preservation property*). (Shieber 1994) proposes a new definition (*isomorphic synchronous TAG*) in which the unused links are removed after a rewrite step. (Shieber defines this system by equivalently requiring the derivation trees of the two derivations to be isomorphic.) However, as Shieber points out, the new definition no longer allows for the representation of certain phenomena, including the ordering of raised quantifiers, and of several non-isomorphic phenomena in translation.

Several authors have since attempted to overcome the descriptive limitations of isomorphic synchronous TAG, while avoiding the explosion in generative capacity of the original definition. Harbusch and Poller (in

in computational linguistics) and *pushdown transducers* and *syntax directed translation schemata* (SDTS) (Aho and Ullman 1969) which translate between context-free languages.

this volume) propose *Dynamic Link Synchronous Tree–Adjoining Grammar* in which the links required for rewriting can be created dynamically. (Rambow and Satta 1996) synchronize two multicomponent TAG systems (see Section 1.4.2). And (Dras 1999) suggests a synchronous derivation of the derivation tree using a metagrammar. Synchronous TAGs have been used extensively, for modeling semantics (see Section 1.3.7 for references), for generation (Section 1.6.3) and most prominently for machine translation (Section 1.6.4).

1.4.5 Stochastic LTAGs

In a probabilistic context-free grammar (PCFG), probabilities are assigned to the production rules in a grammar in such a way that the probabilities of the rules that expand a given nonterminal symbol always sum to one. Using such a grammar, a probability can be assigned to parses of a sentence by multiplying the probability of each production used in that derivation. It is well known that a PCFG defined in this manner is not a good candidate for modeling natural languages, since it is insensitive to lexical context (see (Resnik 1992) and (Charniak 1993) for a discussion). At the same time, it is clear that models based purely on the linear string of words and not on structural hierarchy (n-grams) will never be able to fully capture natural language, either. As result, recent attempts at creating stochastic models based on CFGs incorporate head information in order to overcome the disadvantages of PCFGs (Magerman 1995, Collins 1997), and these attempts have proven quite successful.

Because of its lexicalization, LTAG provides a different way of maintaining both lexical context and structural hierarchy. (Resnik 1992) proposes to carry over the PCFG model to LTAG in a straightforward way. For each elementary tree in the grammar, and for each node in that tree, we consider what operations we can perform at that node. In the case of substitution nodes, we identify all trees that can be substituted; in the case of adjunction nodes (without null-adjunction constraints), we identify all trees that can be adjoined at that node. (If the node does not have an obligatory-adjoining constraint, we also consider the null option of not adjoining anything at that node.) For a given node, we assign each of these possible operations at that node probabilities, so that the probabilities of all possible operations (including the null operation if applicable) sum to one. A derivation can be represented as a sequence of such operations, so we can, as in the case of PCFG, determine the probability of a derivation by multiplying the probabilities of the operations performed during the derivation. (Schabes 1992), using the same intuition, defines stochastic LTAGs (using the equiv-

alent linear index grammar), presents an inside-outside algorithm for reestimating a stochastic LTAG, and presents some initial results for learning grammars from corpora. (Hwa 1998) reports on some initial experiments using probabilistic tree insertion grammar (PTIG; see Section 1.4.1 for a discussion of TIG), which show that it far outperforms PCFG in achieving similar results as n-grams, while at the same time providing rich structural accounts. (Sarkar 1998) shows conditions under which a probabilistic TAG is consistent (i.e., the probabilities of all possible derivations for a given grammar sum to one), and (Nederhof et al. 1998) discuss the computation of prefix probabilities for probabilistic TAG. (Neumann 1998) discusses the automatic extraction of stochastic LTAGs from corpora.

1.5 Relation to Other Frameworks

1.5.1 Dependency Grammar

The relation between TAG and syntactic dependency was already mentioned in Sections 1.2.3 and 1.3.7.[26] The close relation comes from the fact that TAG can be lexicalized: when a TAG is lexicalized, each operation in that TAG (adjunction or substitution) can be seen as establishing a direct relationship between two lexical items (those which anchor the trees being combined). As a direct consequence of this fact, the derivation tree is a dependency tree, since we can identify its nodes with lexemes. But unlike dependency grammars, a TAG analysis also (at the same time) provides a phrase-structure tree.

(Rambow and Joshi 1994a) explore the relationship between TAG and dependency-based approaches to syntax, and observe that the TAG derivation tree is in many ways closer to a "deep" syntactic representation in which not all the words of the surface string are represented as nodes, for example, the Deep-Syntactic Representation (DSyntR) of Meaning-Text Theory (MTT) (Mel'čuk 1988). This is because a single elementary tree in a TAG may contain several words, i.e., several words form a single lexeme. This is the case for strongly governed prepositions (see Section 1.2.1) and idioms (Section 1.3.4), and in both the TAG derivation tree and the DSyntR of MTT strongly governed prepositions are not represented, and idioms are represented as a single node.[27]

The question arises why we should use a phrase-structure representation at all and not simply abandon TAG for dependency grammar and

[26] Dependency syntax is not a unified approach, and much rather different research falls under this term, for example: (Mel'čuk 1988, Hudson 1990, McCord 1990).

[27] There is a difference in auxiliaries: these appear in the derivation tree but not in the DSyntR of MTT.

the ensuing representational simplification. The reason, (Rambow and Joshi 1994a) suggest, is word order: while dependency grammars can give an elegant account of projective word order (where the dependency tree can project to the linear order of its words in the sentence without crossing lines of projection), non-projective word order poses a problem and requires special mechanisms, such as global word order rules (Mel'čuk 1988) or multi-headed structures (Hudson 1990) which complicate the formalism. In TAG, on the other hand, certain non-projective structures (such as *wh*-extraction, raising, and Dutch cross-serial dependencies) can be analyzed in exactly the same way as projective constructions, requiring no further mechanism (see the discussions in Section 1.3).

(Rambow et al. 1995) point out that in certain cases, the TAG derivation structure does not correspond to the dependency structure (as typically analyzed). Instead, they use DTG, a variant of TAG that allows more interleaving of structure (see Section 1.4.2), which allows them to obtain the correct dependency structures. (Joshi and Vijay-Shanker 1999) address the same problem within TAG by reading off the dependency structure from the spine of the derived tree, not the derivation tree; (Schuler 1999) provides algorithms for parsing into this dependency structure and transfer based on it using a revised notion of synchronous TAG. (Nasr 1995) uses a tree-rewriting formalism inspired by TAG to directly model derivations of dependency trees.

1.5.2 Generative Grammar

Generative grammar (Chomsky 1957, Chomsky 1965, Chomsky 1981, Chomsky 1986, Chomsky 1995) has introduced two important concepts into linguistics. One is the formal machinery employed, namely the generation of an initial structure (for example, the D-structure of (Chomsky 1981)) which is subsequently transformed into other structures using rules or principles of transformation (for example, the move-α of (Chomsky 1981)). The second major concept is the methodology: Generative Grammar assumes that grammar is organized into two types of entities, principles and parameters. The principles are language-independent universals, while the values of parameters differ between languages. The language learner succeeds in learning her mother tongue despite the poverty of stimulus because all she needs to do is set the value of the parameters. While the transformational approach has not been universally popular, the principles-and-parameter methodology has been more widely adopted, for example in much work in LFG and HPSG.

For a TAG-based approach, the relevance of Generative Grammar can also be discussed separately according to the two concepts mentioned above. We first discuss the importance of transformations to TAG.

(Kroch 1987) and (1989) first explored the possibility of using transformations to derive the set of initial trees. Specifically, as in Generative Grammar, a lexical item projects structure in a uniform manner and then transformations (move-α) apply, deriving alternate projections for the lexical item, or, to use standard TAG terms, the other members of the tree family. The advantage of this approach to TAG is twofold: first, results can be transferred from Generative Grammar to TAG-based linguistic theories. Second, there is a consistent and principled account of what the elementary structures in the TAG grammar for a particular language are. From the point of view of Generative Grammar, there is also a major advantage: the scope of the application of transformations has been reduced to the elementary structures of TAG, i.e., the projections of single lexical items! This means that much of the work that has been done in Generative Grammar on long-distance effects (e.g., successive-cyclic movement) can be abandoned, since its effects are obtained through the use of the formal operation of adjunction. This line of research has been explored in great detail in (Frank 1992, Frank 2000); in this volume, the contributions by Bleam and Frank most clearly fall into this approach. (Frank and Kroch 1995) provide a detailed discussion of the relation between the Generalized Transformations of (Chomsky 1957), the operations of GT and adjunction of (Chomsky 1995), and TAG.

The relevance of a principles-and-parameters methodology is, as observed, independent of whether one adopts a transformational approach in the style of Generative Grammar. The use of TAG as a language for formulating a theory of syntax is in some sense already a very strong theoretical position in favor of language universals, since the claim is that the constrained grammatical formalism represents (or captures) universal truths about syntax: we can use TAG to model natural language syntax because natural language syntax universally obeys certain principles. The difference between TAG on the one hand and LFG, HPSG, and Generative Grammar on the other hand is that a portion of the universal is captured by the mathematical formalism and is not subject to linguistic theorizing.[28] In TAG-related non-transformational work, (Rambow 1994a) explores a completely non-transformational approach to deriving lexical derivations in a principles-and-parameters methodology using an extension of TAG. Furthermore, all work in grammar development (see Section 1.6.1) can be interpreted as establishing a principles-and-parameters framework as well.

[28] Linguists may object to or welcome this limitation on the scope of their domain, depending on temperament.

1.5.3 Head-Driven Phrase Structure Grammar (HPSG)

Head-driven Phrase Structure Grammar is a surface-based, feature-based, lexicon-driven linguistic theory which has been proposed by (Pollard and Sag 1987, Pollard and Sag 1994) as an alternative to GPSG, in order to incorporate several merits of other linguistic theories such as LFG or GB.

As a linguistic theory, HPSG takes into account various aspects of natural languages, including phonology, morphology, syntax, semantics and certain aspects of discourse phenomena (Bouma et al. 1999).

Formally speaking, it is based, for its syntactic component, on partial descriptions of phrases called Immediate-dominance (or ID) schemata, which are equivalent to underspecified context-free rewriting rules, or elementary trees of depth 1. The nodes are annotated with feature structures. Since the feature structures can be cyclic and unbounded, the generative capacity of HPSG grammars is that of a Turing machine (as shown by (Carpenter 1991) for the case of the subcategorization feature modified by lexical rules). As far as syntax is concerned, although HPSG and TAG linguists may share a lot of intuitions,[29] HPSG itself as a formal model does not have the key properties of TAG:

- although lexical descriptions play a key role, HPSG grammars are not necessarily lexicalized (some phrase structures can have phrasal heads);
- HPSG grammars are not directly equivalent to a mathematically constrained class of formal grammars;
- HPSG grammars do not use an extended domain of locality (phrase structure descriptions usually only comprise immediate constituents), though recent work on clause types proposes phrasal descriptions referring to non-immediate constituents (Sag 1997).

Natural language processing systems using TAG can be simpler and faster than those for HPSG, so for practical purposes, conversion of an HPSG grammar into a TAG grammar can be tempting.[30] The Verbmobil spoken translation project (Kay et al. 1994) involves sizable HPSG grammars for English, German and Japanese. The analysis module is based on HPSG, but the generation system is based on TAG. Thus, for generation, the HPSG grammars are converted into TAG, and (Kasper

[29] The TAG grammar for French (FTAG, see (Abeillé and Candito, in this volume)) implements certain analyses proposed in HPSG.

[30] Since HPSG is formally more powerful than TAG, this conversion is not possible in the general case. Note also that (Makino et al. 1998) propose an alternative conversion (from TAG to HPSG) in order to test their HPSG parser with the sizable XTAG grammar for English.

et al. 1995) and (Becker 1998) propose algorithms for doing so automatically. The idea is to precompile several HPSG ID schemata in order to obtain lexicalized TAG elementary trees. In a bottom-up strategy, starting with lexical descriptions, one defines certain HPSG features as termination features (feature lists which, when empty, correspond to a possible root category for a TAG elementary tree). Furthermore, one needs criteria for choosing the description of a daughter sign as a substitution or foot node (a foot node is chosen when a termination feature of the daughter and the root categories unify). As noted by Becker (1998), this process should not be seen as grammar conversion but rather as a preprocessing, since the output grammar still is, from the linguistic point of view, an HPSG grammar.

1.5.4 Categorial Grammar

Categorial Grammars (CGs) are a system for syntactic calculus originally developed by logicians such as Adjukiewicz, Bar-Hillel or Lambek. Several variants have been proposed, some equivalent to context-free grammars, some more powerful, and some have lead to detailed linguistic analyses (for example (Moortgat 1988, Morrill 1994, Steedman 1996, Steedman 1998)). In all variants, the core of the grammar is a set of complex categories associated with the lexical entries, which are combined using a set of formal rules which typically include functional application and functional composition.

The relationship to TAG is interesting in that both models are mathematical formalisms, and both can be strictly lexicalized. There is a natural parallelism between the complex categories of CG and TAG elementary trees since both are syntactic structured objects anchored in the lexicon. CG's basic logical operations also bear some similarities to TAG adjunction (for composition) and TAG substitution (for application).

Several papers have given a closer look at the relationship between LTAG and some versions of CG (see for example (Abruschi et al. 1999)). From a formal point of view, one can see CG analyses as deduction trees and this logical proof approach to parsing can be adapted to TAG. (Joshi and Kulick 1997) precompile some parts of CG deduction trees using TAG's extended elementary trees for this purpose. From the CG point of view, one can make more precise the generative capacity of CG variants (and extend to them existing parsing algorithms, as was done by (Vijay-Shanker and Weir 1994) for Combinatorial Categorial Grammars). Doran and Srinivas (in this volume) use the close connection between CCG and TAG to compile the XTAG grammar into CCG format, thus obtain a wide-coverage categorial grammar for English. From the TAG point of view, one can adapt some successful CG analyses such as

those proposed for non-constituent coordination (as was done by (Joshi and Schabes 1992)) and this is what Steedman (in this volume) proposes for control and binding phenomena.

1.5.5 Lexical-Functional Grammar (LFG)

LFG (Kaplan and Bresnan 1982) is based on the now fairly standard assumption that there are several interrelated but independent levels of representation. LFG's c-structure is a representation of phrase structure, and is derived by the underlying context-free grammar (CFG). f-structure is a representation of the functional structure of a sentence, using categories such as Subject, Object, and so on. C- and f-structure are related by functional constraints associated with CFG rules, called *functional schemata*. As has been discussed in the LFG and related literature (Maxwell and Kaplan 1993), parsing grammars that are associated with functional constraints is computationally costly, the time complexity being exponential in the length of the input string in the worst case.

(Kameyama 1986), (Burheim 1996) and (Rambow 1996) discuss ways of combining LFG with TAG. The underlying intuition is the same: the CFG-based characterization of c-structure is replaced with a TAG. In LFG terms, in a TAG we "pre-assemble" into a single tree all those c-structure rules whose left-hand side nonterminal will be associated with the same f-structure predicate through the use of the $\uparrow{=}\downarrow$ equation (which indicates syntactic projection in LFG). In a TAG derivation, the action of substituting or adjoining a tree to another corresponds directly to making the lexical item of the first tree an argument or an adjunct of the lexical item of the second tree. Thus, roughly speaking, the requirements of coherence (only one constituent can fill a certain grammatical role) and completeness (each obligatory grammatical role must be filled), which in LFG must be stipulated, are corollaries of the definition of TAG (since substitution nodes require substitution of exactly one tree). (Kameyama 1986) and (Burheim 1996) discuss ways of deriving an f-structure which is a feature structure, while (Rambow 1996) proposes that the derivation structure can simply serve as f-structure (re-entrancy in the f-structure being in practice restricted to bounded phenomena such as control). Reformulating LFG in this manner provides for polynomial parsing algorithms for LFG.

However, many grammars in the LFG framework cannot be expressed in such a TAG-like manner. There are two reasons for this. First, the same functional schema can be used at different nodes in a derivation tree which themselves are associated with the same f-structure. This will have the effect of inducing two dependent derivations from two different nodes, and has been used in the LFG literature to handle the

crossing dependencies of Dutch (Bresnan et al. 1983). While TAG can handle these Dutch constructions (see Section 1.3.5), it cannot derive the phrase structure proposed by (Bresnan et al. 1983), which has two "spines", one carrying the nouns, and the other carrying, in the same order, the verbs. These types of cases can, however, be derived by set-local multicomponent TAG and the formalisms equivalent to it, MCFG and LCFRS (see Section 1.4.2). (Seki et al. 1993) show that a particular type of restricted LFG is weakly equivalent to MCFG/LCFRS. This is significant, since MCFG/LCFRS, like TAG, has restricted generative capacity and is polynomially parsable.

The second reason why many LFG grammars cannot simply be converted to a TAG grammar is the use of *functional uncertainty* (Kaplan and Zaenen 1989) in functional schemata. In functional uncertainty, a functional equation uses regular expressions, which are interpreted as a shorthand for an infinite set of "ordinary" functional equations. This device is used in LFG grammars to handle long-distance extraction of *wh*-words in English, and for a variety of word order phenomena in West Germanic (Zaenen and Kaplan 1995). Using functional equations such as $\downarrow{=}\uparrow$ xcomp* object, one can specify that, for instance, the filler at a specifier position of CP (i.e., the left daughter of an S node) is in fact the object of an arbitrarily deeply embedded clause (designated by xcomp). Functional uncertainty is not considered by (Seki et al. 1993) and cannot in general be modeled by set-local multicomponent TAG: (Vijay-Shanker and Joshi 1989) show that functional uncertainty is a corollary in TAG, but this is of course only true for those cases in which TAG can provide an analysis of long-distance extraction. It is easy to construct formal-language examples that LFG with functional uncertainty can derive, but TAG cannot. Furthermore, as (Becker et al. 1991) and (Rambow et al. 1995) argue, there are linguistic examples in which TAG cannot provide a linguistically motivated analysis; functional uncertainty can derive these cases. (Rambow 1996) proposes a version of DSG (Section 1.4.2) in which dominance links can be annotated with regular expressions, which retains the polynomial parsability of lexicalized DSG, and which can be used to model LFG with functional uncertainty.

1.6 Applications

LTAGs have been used for several applications in natural language processing. We briefly overview here their application for grammar development, parsing, generation and machine translation.

1.6.1 Grammar development

LTAGs have been used to develop sizable grammars for several natural languages. These grammars have been developed from scratch, starting from some more abstract level of representation, or starting from an existing grammar in another formalism. The LTAG grammars can be used directly for parsing, translation or generation applications (see below), or simply as input for other grammar developments projects, such as a sizable CCG grammar (see Doran and Srinivas, in this volume).

The XTAG System

The XTAG system (see (Doran et al., this volume) and (XTAG-Group 1999)) is the most complete TAG workbench currently available (others are being built, for example (Lopez 1999)). It includes a graphical interface (Paroubek et al. 1992), a parser (Schabes 1994) and a lexicon compiler. It is a resource-independent platform which can be used for a variety of languages. In its English version, it also includes a part-of-speech tagger and a supertagger. It divides the internal representation of a lexicalized TAG into three files, which can be developed and maintained independently:

- a morphological lexicon which associates the full inflected forms of a language with their lemma, part of speech and some features;
- a syntactic lexicon which associates each lemma with its schematic tree or tree family and some features;
- schematic trees usually organized into tree families which gather trees instantiating regular syntactic alternations for a given subcategorization frame.

The three resources are compiled to produce the lexicalized TAG used for parsing (or supertagging). An example is shown in Figure 30.

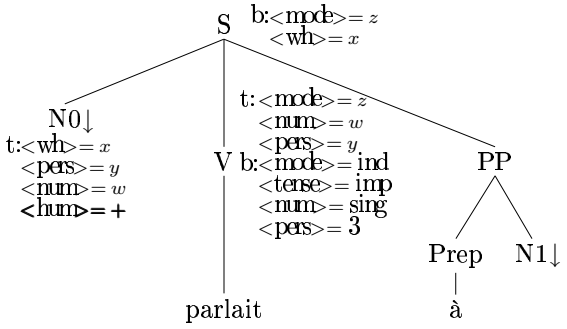

FIGURE 30 An example of elementary tree

Figure 31 shows the three pieces of information that represent it internally (from FTAG, see (Abeillé and Candito, in this volume)). The inflected form *parlait* ('talked') points to the lemma PARLER in the morphological lexicon, that selects the n0Van1 family in the syntactic lexicon, with a dative complement and the preposition *à* (*to*) as a co-anchor.

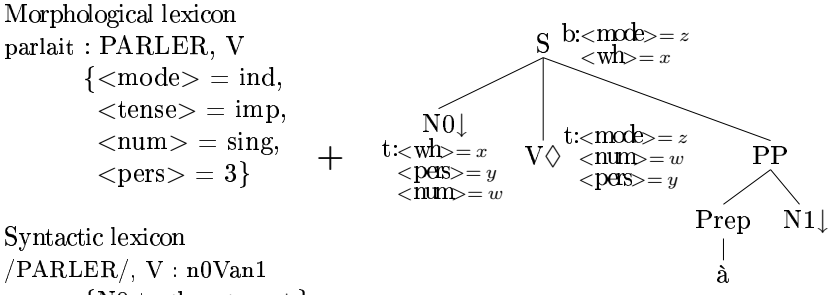

FIGURE 31 Three sources of information for the tree of Figure 30

Machine-Aided Grammar Development

Even with this internal division of labor, elementary tree sketches are complex objects which encode different types of syntactic information usually distinguished in other formalism: word order, subcategorization, transitivity alternation, syntactic realization of each argument, type of clause, and so on. This leads to a multiplication of elementary trees (a sizable LTAG easily comprises 1000 elementary trees) and to an important redundancy between them. This situation has lead some authors to propose a more abstract level of representation for elementary trees: (Vijay-Shanker and Schabes 1992, Candito 1996, Xia et al. 1998) and (Becker, in this volume). All these proposals borrow some representational mechanism from other frameworks: the GPSG metarules, the LFG lexical rules, the HPSG type hierarchy and inheritance. From the theoretical point of view, these proposals help express syntactic generalizations hidden in the elementary trees (for example across tree families). From the practical point of view, they enable the grammar writer to specify only parts of the different elementary trees which are automatically unified and compiled. The general idea is to adapt for LTAGs the classification mechanisms used for the lexicon in other frameworks (such as HPSG). The difficulty is that the elementary trees in a TAG

are more complex objects than just categories or features structures. Abstracting away the feature equations associated with each elementary tree is not the most difficult part. Having them also share some of their topology calls for a tree representation language such as (Rogers and Vijay-Shanker 1994). (Vijay-Shanker and Schabes 1992) propose to decompose the elementary trees using precedence, immediate dominance and unspecified dominance relations, and to organize them in a type hierarchy where some of their topological properties can be shared. They use lexical rules for transitivity alternations such as passive.

Candito (1996, 1999)(Candito 1996, Candito 1999) proposes a universal three-dimensional classification for lexicalized grammars, which she calls a metagrammar (MG). MG comprises one dimension for basic subcategorization information, one for function redistribution (or transitivity alternations) and one for the syntactic realization of heads and arguments (extraction, cliticization...), and each elementary tree inherits from the 3 dimensions (see Abeillé and Candito (in this volume) for an example of MG for French). Given her functional approach to subcategorization, she can use strict inheritance mechanisms (without defaults) and dispense with lexical rules. She also offers a tool for automatically generating a complete (or partial) LTAG starting with MG. (Xia et al. 1998) propose a similar, but independently developed, approach.

Becker (in this volume) proposes metarules operating on portions of elementary trees in order to automatically augment or modify an existing LTAG. Since the metarules operate on finitely bounded trees, they do not increase the generative capacity of the formalism (contrary to GPSG metarules). Evans et al. (in this volume) propose to use DATR as a metalanguage to code and maintain LTAGs. They use default mechanisms and lexical rules.

Current TAG grammars are available for English, French, Italian, Korean. Others are being developed for Chinese, Portuguese and German.

Grammar Migration

Grammar migration — exporting the syntactic resources from one framework or platform to another — has been a subject of study for quite a while. Two experiments have been made to produce a sizable TAG starting from a sizable HPSG grammar, namely (Kasper et al. 1995) for English, and (Becker et al. 1998) for German. See Section 1.5.3 for a fuller presentation.

It is worth noting that TAG grammars themselves are syntactic databases which can be reused for other formalisms. Doran and Srinivas (in this volume) present the automatic development of a Combinatory

Categorial grammar (CCG) starting with the English XTAG.

1.6.2 Parsing

In this section, we first survey the theoretical literature on TAG recoognition and parsing algorithms, and then turn to implementations and practical issues.

Theoretical Work

Several recognition and parsing algorithms have been defined for TAG. The basic bottom-up chart parser (Vijay-Shanker 1987) proceeds bottom-up in recognizing the elementary trees used in a derivation, and also proceeds bottom-up in assembling these elementary trees into a derivation. Items representing recognized subtrees are stored in a four-dimensional table to account for the fact that partial derivations may include a footnode; hence, not only are two indices needed to record the beginning and the end of the derived substring, but also two indices are needed to record the beginning and the end of the string which is dominated by the footnode. The most complex operation is that of adjunction of an auxilary tree into an auxiliary tree, in which case items with four indices must be matched with items with two independent indices (representing the other auxiliary tree's footnode), giving us a $O(n^6)$ time complexity (worst and best case).

Three different Earley-style algorithms (that combine bottom-up parsing with top-down prediction on derived trees) have been proposed. (Schabes and Joshi 1988) present an algorithm whose worst-case time complexity is in $O(n^9)$. However, average case run time is better, and furthermore the algorithm has the *valid prefix property*: processing the input string left-to-right, the algorithm rejects an input string as soon as the prefix it has read is not the prefix of any string in the language being recognized (or parsed). The algorithm of (Schabes 1994) reduces time complexity to $O(n^6)$ at the cost of giving up the valid prefix property. (Nederhof 1999) presents an algorithm that has the valid prefix property and has worst-case time complexity in $O(n^6)$.

A head-driven algorithm was first proposed by (Lavelli and Satta 1991). The algorithm extends parses along the path from the anchor of an elementary tree to its root by performing adjunctions. (van Noord 1994) introduces the notion of "headed TAG" in order to define a head-corner parser for TAG. In a headed TAG, each anchor of an initial tree is a head corner for its root, while in an auxiliary tree the foot node must be a head corner. All these algorithms achieve a worst-case $O(n^6)$ time complexity.

A different type of algorithm was first proposed by (Harbusch 1990)

and subsequently studied by (Poller 1994) and (Poller and Becker 1998). Here, a context-free grammar called the *kernel grammar* is created, which can derive all of the elementary trees in the TAG grammar. The input is parsed with the kernel gramar, and in a second step those derivations are eliminated from the context-free parse forest which are not compatible with the TAG grammar. The time complexity of this algorithm is also in $O(n^6)$.

Several other parsing algorithms for TAG have been proposed exploiting the similarity between TAG parsing and other computations. (Vijay-Shanker and Weir 1993) propose a bottom-up algorithm on the derived tree which is inspired by the equivalence of TAG to linear index grammar (LIG, see Section 1.2.5). The $O(n^6)$ algorithm extends the algorithm for context-free grammar by using pointers to track the development of the stack of index symbols. (Boullier 1999) exploits the equivalence of TAG to a restricted version of range concatenation grammars to present a $O(n^6)$ parsing algorithm for TAG.

It is striking that no parsing algorithm has been found for general TAG with worst-case time complexity better than $O(n^6)$. It turns out that this is probably not a coincidence. (Satta 1994) shows that a TAG parsing algorithm that is faster than $O(n^6)$ implies that we have a Boolean matrix multiplication algorithm that is faster than $O(n^3)$. Such algorithms are hard to find, and those that have been presented have high constants making them unappealing for practical computation. Therefore, finding a better TAG parser is also a hard probelm, and it is unlikely that a simple and practical algorithm will be found which significantly improves on the $O(n^6)$ time complexity (which no current algorithm beats).

As noted in Section 1.4.1, several restricted versions of TAG have been proposed specifically in order to reduce the parsing complexity. (Schabes and Waters 1995) and (Rogers 1994) propose restricted versions of TAG without full adjunction that generate only context-free languages and that can be parsed in $O(n^3)$ time, while (Satta and Schuler 1998) allow full adjunction, but only at one node of the spine of an auxiliary tree, thus reducing the time complexity of parsing to $O(n^5)$.

For deterministic parsing, (Schabes and Vijay-Shanker 1990) explore the use use the bottom-up embedded pushdown automaton in an LR parsing algorithm for TAG, but (Kinyon 1997) discusses problems of corectness in this approach. She proposes a way to incorporate lexicalization information offline into an LR table, in order to reduce conflicts online via filtering. (Nederhof 1998) presents an implemented LR parser based using the pushdown automaton for linear index grammars, and finds the LR tables to be "prohibitively large" for a grammar the size of

the XTAG grammar. (Prolo 2000) proposes a modified algorithm that greatly reduces the number of states.

Implementations

We now turn now to implemented parsers used in implemented systems and/or with wide coverage. The XTAG system (see Section 1.6.1 for a fuller description) includes an implementation of the Earley-style parser of (Schabes 1994). (See (Doran et al. 1994) for a discussion of the results of this parser on corpora.) Like all parsers discussed in the theoretical section above (except the deterministic parsers), this parser returns all possible parses, unranked. (Doran et al. 1994) report that XTAG finds an average of 7.46 parses for a corpus of 18,730 sentences from the *Wall Street Journal*. Since typically in applications, the goal is not to obatin all possible syntactic analyses for the input but rather a single one (which can then be used as the basis for further processing), a significant body of work has been performed to address this issue. (Srinivas et al. 1995) propose a set of domain-independent but language- and grammar-specific heuristics to rerank parses in a postprocessing step. (Kinyon 1999) extends this approach by using insights gained from psycholinguistics. Specifically, she adapts the well-known principles of *minimal attachment* and *right associataion* to derivation trees (rather than derived trees), and also incorporates a preference for initial trees (i.e., arguments) over adjunct trees.

A different approach in chosen in *supertagging*:[31] instead of choosing among several candidate parses, supertagging eliminates possible parses from the outset. (Srinivas and Joshi 1999) propose to associate lexical items in the input string with the names of elementary trees in a TAG grammar, which function as syntactially rich descriptions (hence the term "supertag"). By using stochastic techniques to assign supertags to lexemes, the actual parsing becomes nearly trvial since the supertags already contain extensive syntactic information. If supertagging (assigning tags to lexemes) were pefect, any TAG parser could be used to derive the parse, but since suppertagging (using trigrams with smoothing on 1,000,000 word training corpus) achieves only about 92% accuracy, (Srinivas and Joshi 1999) propose instead to use a heuristic linear-time "lightweight dependency analyzer" to derive the parse tree. (Chen et al. 1999) investigate ways of improving supertagging accuracy.

Several other practical aspects of using TAG parsers in applications have been discussed in the literature. (Lopez et al. 1999) discuss issues

[31] Note that the "tag" in "supertagging" is not the abbreviation with the meaning "tree adjoining grammar", but rather the noun with the meaning "attached information".

related to extracting an LTAG grammar for a sublanguage. (Lopez and Roussel 1998) discuss issues related to parsing of spontaneous spoken language with TAG, such as self repair and repitition. Issac and Fouqueré (in this volume) use a bottom-up TAG parser in an application for learning lexical items in a foreign language.

1.6.3 Generation

(Joshi 1987b) claims that TAG has properties that make it particularly suited as a syntactic representation for generation. Specifically, its extended domain of locality is useful in generation for localizing syntactic properties (including word order as well as agreement and other morphological processes), and lexicalization is useful for providing an interface from semantics (the derivation tree represent the sentence's predicate-argument structure). Indeed, generation was the first application to use TAG as a grammatical framework: (McDonald and Meteer 1990) (a project that started in the early 80s) use TAG in a generation project aimed at modeling human sentence production (also see (Meteer 1992) for an overview and (McDonald and Pustejovsky 1985) for a discussion specifically of the role of TAG).

A series of projects at the University of the Saarland and at the DFKI have used TAG as the underlying formalism in sentence generation for a sequence of applications (Harbusch et al. 1991, Wahlster et al. 1993, Kilger 1994). One of the emphases in this work has been on incremental generation, requiring modifications to TAG to allow for the permutation of constituents and the representation of underspecification. The formalism discussed by Harbusch (in this volume) is motivated by (among others) considerations of incremental generation. More recently, (Becker et al. 1998) describe a (non-incremental) system which transforms a semantic representation (used as the interlingua) to a surface form (also see (Becker 1998)). The syntactic generator proper produces a surface string from the dependency representation, by choosing trees for each node in the dependency using a best-first-search strategy. The grammar used in the syntactic generator is a TAG grammar compiled from an HPSG grammar in a manner similar to that described in (Kasper et al. 1995).

In other work, (Yang et al. 1991) present a system in which a systemic-functional network is combined with a lexicalized TAG grammar: the systemic-functional network is used to choose realization features based in functional criteria; these features are in turn used to choose elementary trees. (Shieber and Schabes 1991) use synchronous TAG (see Section 1.4.4) to map from a semantic representation to the syntactic representation provided by TAG; as is the case with other

synchronous TAG applications, their approach provides for reversibility (the same grammar can be used for parsing as well). (Stone and Doran 1997, Stone and Webber 1998) extend lexicalized TAG with features designed to describe semantic and pragmatic interpretation to guide syntactic and lexical choice during sentence planning. (Nicolov and Mellish 2000) use the TAG-related formalism DSG (see Section 1.4.2) in a system that generates English sentences from conceptual graphs. Finally, Danlos (in this volume) presents G-TAG, an extension of TAG which allows her to extend lexicalization beyond the traditional domain of the sentence by encoding the syntax of clausal connectives and of multi-sentential text, and by representing the conceptual-semantic interface in the formalism as well.

1.6.4 Machine Translation

(Abeillé et al. 1990) have proposed to use LTAG for machine translation. The proposed application is based on synchronous TAG (Section 1.4.4) in the following way:

- bilingual dictionaries are seen as correspondences between lexicalized elementary trees (or their tree families);
- the parsing and generation phases are based on the monolingual TAG grammars (and systems) developed independently of each other;
- the transfer component is replaced by the synchronization of the monolingual TAG derivations,
- some transfer principles may be added to match feature equations or different members of tree families (for example when passive exists in the source language but not in the target language).

The advantages of such an approach include:

- it is modular (a module for a given language can be reused with another paired language);
- it is reversible;
- one can reuse TAG grammars and lexicons developed in a monolingual perspective, as well as TAG parsing and generation modules;
- it is based on derivation and not derived structures, thus minimizing the structural mismatches between the paired languages;
- it is lexicalized, so one can easily incorporate all the lexical idiosyncrasies often found when translating natural languages.

For paired sentences from different languages, derivation trees are usually closer than derived (or phrase structure) trees because the derivation trees express pure lexical argument structure rather than syntactic

detail (see Section 1.2.3). For example, even when a simple verb has to be translated into a phrasal expression (light verb construction) in the target language, or when a direct objet must be translated as a prepositional object, the derivation trees are often identical (or very similar). Relevant examples include the English/French pairs (E) *commute* = (F) *faire la navette*, (E) *allude to* = (F) *faire allusion à*, or (E) *resemble* = (F) *ressembler à*

In addition, structural divergences in different languages (Dorr 1994) can often be handled by simply linking up nodes in trees in a non-canonical manner. Complex restructuring rules that are often necessary in other MT systems can thus be avoided, since in the TAG approach, there is no requirement that elementary or derived trees be identical. For example, in the following the subject of English *miss* becomes the prepositional object of French *manquer à*, and the direct object in English becomes the subject in French.

(13) (F) Jean manque à Marie = (E) Marie misses Jean

The solution in synchronous TAG is shown in Figure 32. Note that this pair of trees can be used for translation in either direction.

(Shieber 1994) proposes to view the synchronous derivations as isomorphic (same number of nodes and same dominance relations), but Harbusch and Poller (in this volume) show counter-examples and propose an alternative implementation (also see Section 1.4.4 for a fuller discussion of synchronous TAG).

Another advantage of using LTAGs is that lexicalized elementary trees provide an extended domain of locality for lexical disambiguation. Frequently, one word in the source language corresponds to several words in the target language, but often the correct word can be chosen as a function of a complement of the word. For example, English *wear* may be translated into Japanese as *kaburu* or *haku*, but the choice can be made dependeing on the direct object (the first is used for items worn on the head). Palmer et al. (in this volume) show how to exploit the extended domain of locality of TAG by associating a rich array of semantic features with verbs and their arguments in the context of an English to Chinese MT system. These semantic features guide the lexical choice.

1.6.5 Psycholinguistic Modeling

The psycholinguistic relevance of tree adjoining grammars has been studied by (Joshi 1990, Kim et al. to appear, Kinyon 1999). Joshi (1990) shows that a performance model based ob the embedded push-down automaton associated with TAG predicts that Dutch crossed-serial dependencies are easier to parse that German nested dependencies, which

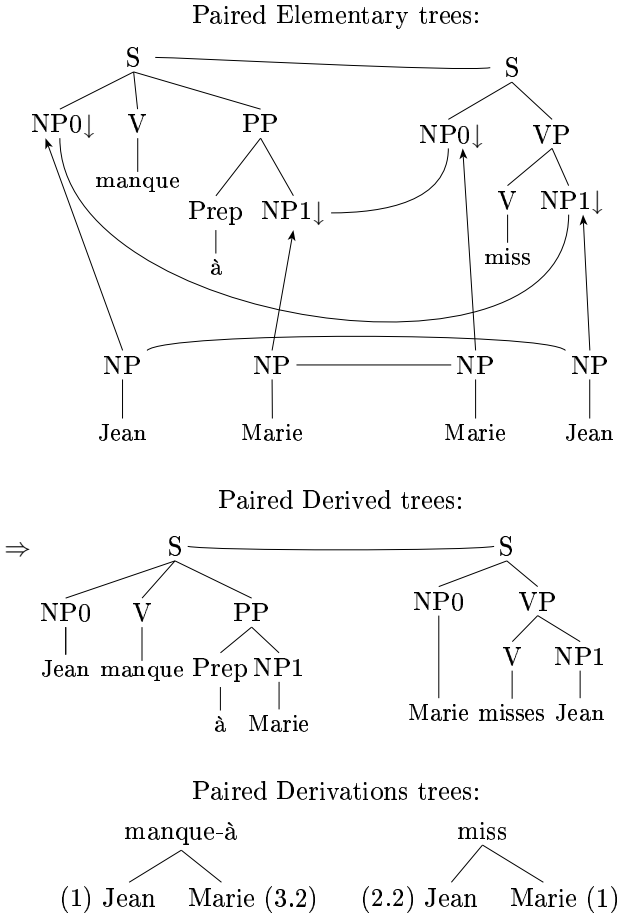

FIGURE 32 Bilingual synchronous derivations

is what psycholinguistic studies have shown (Bach et al. 1986). (Rambow and Joshi 1994b) extend the model to make predictions about acceptability of German sentences with various word orders. Kim et al. emphasize the importance of lexical triggers for parsing and propose a TAG-based connectionist model. Kinyon (1999) shows the importance of the notion of derivation tree in LTAG, which makes the correct predictions for parsing preferences and garden path effects (better than the derived phrase structure tree). For example, the preference for substitution over adjunction explains the well known preference for argument over adjunct interpretation, while the preference for the derivation tree

with fewer nodes (fewer operations) mirrors the well known prefence for idiomatic over literal interpretations.

Joshi et al. (in this volume) show that LTAG modeling can predict the agrammaticality of German long distance scrambling up to two levels of embedding, a result usually attributed to performance effects. Frank (in this volume) models the diferent stage of syntax acquisition for children as coming not from different grammars but from different operations available at each stage: first iteration (or reduplication), then substitution, and finally adjunction. It mirrors Kinyon's proposal for parsing preferences, also based on the fact that substitution is less costly than adjunction.

References

Abeillé, Anne. 1991. Une grammaire lexicalise d'arbres adjoints pour le francais. Doctoral dissertation, Université Paris 7.

Abeillé, Anne. 1995. The Flexibility of French Idioms: A Representation with Lexicalized Tree Adjoining Grammar. In Idioms: Structural and Psychological Perspectives, ed. Martin Everaert, Erik-Jan van der Linden, André Schenk, and Rob Schreuder. Hillsdale, NJ: Lawrence Erlbaum Associates.

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

Abeillé, Anne, and Yves Schabes. 1996. Non Compositional Discontinuous Constituents in TAG. In Discontinuous Constituency, ed. A. van Horck and W. Sijtsma. Berlin: Mouton de Gruyter.

Abeillé, Anne, Yves Schabes, and Aravind Joshi. 1990. Using Lexicalized TAGs for Machine Translation. In Proceedings of the 13th International Conference on Computational Linguistics (COLING'90). Helsinki. COLING-90.

Abruschi, V. Michele, Christophe Fouqueré, and Jacqueline Vauzeilles. 1999. Tree Adjoining Grammars in a fragment of the Lambek calculus. Computational Linguistics 25(2):209–236.

Aho, Alfred V., and Jeffrey D. Ullman. 1969. Syntax Directed Translations and the Pushdown Assembler. J. Comput. Syst. Sci. 3(1):37–56.

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

Becker, Tilman. 1994. A New Automaton Model for TAGs: 2-SA. Computational Intelligence 10(4):422–430.

Becker, Tilman. 1998. Fully Lexicalized Head-Driven Syntactic Generation. In Proceedings of the 8th International Workshop on Natural Language Generation. Niagara-on-the-Lake, Ontario.

Becker, Tilman, Wolfgang Finkler, Anne Kilger, and Peter Poller. 1998. An Efficient Kernel for Multilingual Generation in Speech-to-Speech Dialog

Translation. In 36th Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics (COLING-ACL'98), 110–116. Montréal, Canada.

Becker, Tilman, Aravind Joshi, and Owen Rambow. 1991. Long Distance Scrambling and Tree Adjoining Grammars. In Fifth Conference of the European Chapter of the Association for Computational Linguistics (EACL'91), 21–26. ACL.

Becker, Tilman, Owen Rambow, and Michael Niv. 1992. The Derivational Generative Power, or, Scrambling is Beyond LCFRS. Technical Report IRCS-92-38. Institute for Research in Cognitive Science, University of Pennsylvania. A version of this paper was presented at MOL3, Austin, Texas, November 1992.

Boullier, Pierre. 1999. On TAG Parsing. In Proceedings TALN'99. Cargèse, France. to appear in TAL 2000.

Bouma, Gosse, Erhard Hinrichs, Gert-Jan Kruijff, and Richard Oehrle (ed.). 1999. Constraints and Resources in Natural Language Syntax and Semantics. CSLI Publications.

Bresnan, Joan W. 1982. The Mental Representation of Grammatical Relations. Cambridge, Massachusetts: MIT Press.

Bresnan, Joan W., Ronald M. Kaplan, S. Peters, and Annie Zaenen. 1983. Cross-serial dependencies in Dutch. Linguistic Inquiry 13:613–635.

Burheim, Tore. 1996. Aspects of Merging Lexical-Functional Grammar with Lexicalized Tree-Adjoining Grammar. Unpublished abstract, University of Bergen.

Candito, Marie-Hélène. 1996. A principle-based hierachical representation of LTAG. In Proceedings of the 16th International Conference on Computational Linguistics (COLING'96). Copenhagen.

Candito, Marie-Hélène. 1999. Reprsentation modulaire et paramtrable de grammaires lectroniques lexicalises. Application au franais et l'italien. Doctoral dissertation, University Paris 7.

Candito, Marie-Hélène, and Sylvain Kahane. 1998. Can the TAG derivation tree represent a semantic graph? An answer in the light of Meaning-Text Theory. In Proceedings of the Fourth International Workshop on Tree Adjoining Grammars and Related Frameworks (TAG+4), 21–24. IRCS Report 98–12. Institute for Research in Cognitive Science, University of Pennsylvania.

Carpenter, Robert. 1991. The generative power of categorial grammars and HPSG with lexical rules. Computational Linguistics 17(3):301–314.

Charniak, Eugene. 1993. Statistical Language Learning. MIT Press.

Chen, John, B. Srinivas, and K. Vijay-Shanker. 1999. New Models for Improving Supertag Disambiguation. In Ninth Conference of the European Chapter of the Association for Computational Linguistics (EACL'99). Bergen, Norway.

Chomsky, Noam. 1957. Syntactic Structures. The Hague: Mouton.

Chomsky, Noam. 1965. Aspects of the Theory of Syntax. Cambridge, Mass.: MIT Press.

Chomsky, Noam. 1981. Lectures in Government and Binding. Studies in generative grammar 9. Dordrecht: Foris.

Chomsky, Noam. 1986. Barriers. Cambridge, Mass.: MIT Press.

Chomsky, Noam. 1995. The Minimalist Program. Cambridge: MIT Press.

Chomsky, Noam, and Marcel Paul Schützenberger. 1963. The algebraic theory of context-free languages. In Computer Programming and Formal Systems, ed. P. Braffort and D. Hirschberg. 118–161. Amsterdam, The Netherlands: North-Holland.

Collins, Michael. 1997. Three Generative, Lexicalised Models for Statistical Parsing. In Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics. Madrid, Spain, July.

Dahlhaus, Elias, and Manfred K. Warmuth. 1986. Membership for growing Context-Sensitive Grammars is polynomial. J. Comput. Syst. Sci. 33:456–472.

Dassow, Jürgen, and Gheorghe Păun. 1989. Regulated Rewriting in Formal Language Theory. Berlin, Heidelberg, New York: Springer Verlag.

Doran, Christy, Dania Egedi, Beth Ann Hockey, B. Srinivas, and Martin Zaidel. 1994. XTAG System - A Wide Coverage Grammar for English. In Proceedings of the 17^{th} International Conference on Computational Linguistics (COLING '94). Kyoto, Japan, August.

Dorr, Bonnie J. 1994. Machine Translation Divergences: A Formal Description and Proposed Solution. Computational Linguistics 20(4):597–635.

Dras, Mark. 1999. A Metalevel Grammar: Redefining Synchronous TAGs for Translation and Paraphrase. In 37th Meeting of the Association for Computational Linguistics (ACL'99). College Park, Maryland.

Frank, Robert. 1992. Syntactic Locality and Tree Adjoining Grammar: Grammatical, Acquisition and Processing Perspectives. Doctoral dissertation, Department of Computer and Information Science, University of Pennsylvania.

Frank, Robert. 2000. Phrase Structure Composition and Syntactic Dependencies. Cambridge, Mass.: MIT Press.

Frank, Robert, and Anthony Kroch. 1995. Generalized Transformations and the Theory of Grammar. Studia Linguistica 49(2):103–151. Available at http://www.cog.jhu.edu/faculty/rfrank/glow93.ps.

Gardent, Claire, and Bonnie Webber. 1998. Varieties of Ambiguity in Incremental Discourse Processing. In Proceedings of AMLap-98 (Architectures and Mechanisms for Language Processing). Freiburg, Germany.

Gazdar, Gerald. 1988. Applicability of indexed grammars to natural languages. In Natural Language Parsing and Linguistic Theories, ed. U. Reyle and C. Rohrer. Dordrecht: D. Reidel.

Gazdar, Gerald, Ewan Klein, Geoffrey Pullum, and Ivan Sag. 1985. Generalized Phrase Structure Grammar. Cambridge, Mass.: Harvard University Press.

Grimshaw, Jane. 1991. Extended Projection. In Lexical Speciøcation and Lexical Insertion, ed. Peter Coopmans, Martin Everaert, and Jane Grimshaw. Hillsdale, NJ: Lawrence Erlbaum Associates.

Gross, Maurice. 1989. The Use of Finite Automata in the Lexical Representation of Natural Language. In Proceedings of the LITP Spring School on Theoretical Computer Science: Electronic Dictionnairies and Automata in Computational Linguistics, ed. M. Gross and D. Perrin, LNCS, Vol. 377, 34–50. Berlin. Springer.

Harbusch, Karin. 1990. An Efficient Parsing Algorithm for Tree Adjoining Grammars. In Proceedings of the 28th Annual Meeting of the Association for Computational Linguistics (ACL˜90), 284–291. Pittsburgh, Pennsylvania, USA.

Harbusch, Karin, Wolfgang Finkler, and Anne Schauder. 1991. Incremental Syntax Generation with Tree Adjoining Grammars. In Proceedings 4.Int. GI-Kongress Wissensbasierte Systeme. München. GWAI.

Harley, Heidi, and Seth Kulick. 1998. TAG and Raising in VSO Languages. In Proceedings of the Fourth International Workshop on Tree Adjoining Grammars and Related Frameworks (TAG+4), 62–65. IRCS Report, No. 98–12. Institute for Research in Cognitive Science, University of Pennsylvania.

Heycock, Caroline. 1987. The structure of the Japanese causative. Technical report MS-CIS-87-55. Department of Computer and Information Science, University of Pennsylvania.

Hudson, Richard. 1990. English Word Grammar. Oxford: Basil Blackwell.

Hwa, Rebecca. 1998. An Empirical Evaluation of Probabilistic Lexicalized Tree Insertion Grammars. In 36th Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics (COLING-ACL'98), 557–563. Montréal, Canada.

Joshi, Aravind K. 1985. How Much Context-Sensitivity is Necessary for Characterizing Structural Descriptions — Tree Adjoining Grammars. In Natural Language Processing ¯ Theoretical, Computational and Psychological Perspective, ed. D. Dowty, L. Karttunen, and A. Zwicky. 206–250. New York, NY: Cambridge University Press. Originally presented in 1983.

Joshi, Aravind K. 1987a. An Introduction to Tree Adjoining Grammars. In Mathematics of Language, ed. A. Manaster-Ramer. 87–115. Amsterdam: John Benjamins.

Joshi, Aravind K. 1987b. Word-Order Variation in Natural Language Generation. Technical report. Department of Computer and Information Science, University of Pennsylvania.

Joshi, Aravind K. 1990. Processing Crossed and Nested Dependencies: an Automaton Perspective on the Psycholinguistic Results. Language and Cognitive Processes 5(1):1–27.

Joshi, Aravind K. 1994. Preface to Special Issue on Tree-Adjoining Grammars. Computational Intelligence 10(4):vii–xv.

Joshi, Aravind K., and Seth Kulick. 1997. Partial Proof Trees as Building Blocks for a Categorial Grammar. Linguistics and Philosophy 20(6):637–667.

Joshi, Aravind K., Leon Levy, and M. Takahashi. 1975. Tree Adjunct Grammars. J. Comput. Syst. Sci. 10:136–163.

Joshi, Aravind K., and Yves Schabes. 1992. Fixed and flexible phrase structure: coordination in Tree Adjoining Grammars. Language Research 26(4).

Joshi, Aravind K., and K. Vijay-Shanker. 1999. Compositional Semantics with LTAG: How much Underspecification is Necessary? In Proceedings IWCS. Tilburg.

Joshi, Aravind K., K. Vijay-Shanker, and David J. Weir. 1991. The Convergence of Mildly Context-Sensitive Grammatical Formalisms. In Foundational Issues in Natural Language Processing, ed. P. Sells, S. Shieber, and T. Wasow. 31–81. Cambridge, Mass.: MIT Press.

Kallmeyer, Laura. 1996. Tree Description Grammars. In Natural Language Processing and Speech Technology. Results of the 3rd KONVENS Conference, ed. D. Gibbon, 332–341. Berlin. Mouton de Gruyter.

Kallmeyer, Laura, and Aravind K. Joshi. 1999. Factoring Predicate Argument and Scope Semantics: Underspecified Semantics with LTAG. In Proceedings of the 12^{th} Amsterdam Colloquium, ed. P. Dekker.

Kameyama, Megumi. 1986. Characterising Lexical Functional Grammar (LFG) in terms of Tree Adjoining Grammar (TAG). Unpublished Manuscript. Dept. of Computer and Information Science, University of Pennsylvania.

Kaplan, Ronald M., and Joan W. Bresnan. 1982. Lexical-Functional Grammar: A Formal System for Grammatical Representation. In The Mental Representation of Grammatical Relations, ed. J. W. Bresnan. Cambridge, Mass.: MIT Press.

Kaplan, Ronald M., and Annie Zaenen. 1989. Long Distance Dependencies, Constituent Structure, and Functional Uncertainty. In Alternative Conceptions of Phrase Structure, ed. M. Baltin and A. Kroch. Chicago. IL: University of Chicago Press.

Kasper, Robert, Bernd Kiefer, Klaus Netter, and K. Vijay-Shanker. 1995. Compilation of HPSG and TAG. In 33rd Meeting of the Association for Computational Linguistics (ACL'95), 92–99.

Kay, Martin, Mark Gawron, and Peter Norvig. 1994. Verbmobil: A Translation System for Face-to-Face Dialog. Stanford: CSLI Publications.

Kilger, Anne. 1994. Using UTAG for Incremental and Parallel Generation. Computational Intelligence 10(4):591–603.

Kim, Albert, B. Srinivas, and John Trueswell. to appear. The convergence of lexicalist perspectives in psycholinguistics and computational linguistics. In Sentence Processing and the Lexicon: Formal, Computational and Experimental Perspectives, ed. Paola Merlo and Suzanne Stevenson. John Benjamins.

Kinyon, Alexandra. 1997. Un analyseur déterministe pour les grammaires d'arbres adjoints lexicalisées. In Proceedings 4e TALN. Grenoble, France.

Kinyon, Alexandra. 1999. The Psycholinguistic Relevance of LTAGs. In Proceedings CLIN. Utrecht.

Kroch, Anthony. 1987. Subjacency in a Tree Adjoining Grammar. In Mathematics of Language, ed. A. Manaster-Ramer. 143–172. Amsterdam: John Benjamins.

Kroch, Anthony. 1989. Asymmetries in Long Distance Extraction in a Tree Adjoining Grammar. In Alternative Conceptions of Phrase Structure, ed. Mark Baltin and Anthony Kroch. 66–98. University of Chicago Press.

Kroch, Anthony, and Aravind K. Joshi. 1985. The Linguistic Relevance of Tree Adjoining Grammar. Technical Report MS-CS-85-16. Department of Computer and Information Sciences, University of Pennsylvania.

Kroch, Anthony, and Owen Rambow. 1994. Defective Complements. In 3^e Colloque International sur les Grammaires d'Arbres Adjoints (TAG+3). Rapport Technique TALANA-RT-94-01. Université Paris 7.

Kroch, Anthony, and Beatrice Santorini. 1991. The Derived Constituent Structure of the West Germanic Verb Raising Construction. In Principles and parameters in comparative grammar, ed. R. Freidin. 269–338. Cambridge, Mass.: MIT Press.

Kuno, Susumu. 1973. Constraints on internal clauses and sentential subject. Linguistic Inquiry 3:363–385.

Lavelli, Alberto, and Giorgio Satta. 1991. Bidirectional Parsing of Lexicalized Tree Adjoining Grammars. In Fifth Conference of the European Chapter of the Association for Computational Linguistics (EACL'91), 27–32. Berlin.

Lee, Young-Suk. 1993. Scrambling as case-driven obligatory movement. Doctoral dissertation, University of Pennsylvania. Available as Technical Report 93–06 from the Institute for Research in Cognitive Science at the University of Pennsylvania.

Lopez, Patrice. 1999. Analyse d'noncs oraux pour le dialogue homme-machine l'aide de grammaires lexicalises d'arbres. Doctoral dissertation, University Nancy I.

Lopez, Patrice, Christine Fay-Varnier, and Azim Roussanaly. 1999. Sous-langages d'application et LTAG: le système EGAL. In Actes de la Sixime Confrence Annuelle sur le Traitement Automatique des Langues Naturelles (TALN99), 223–232. Cargeèse, Corsica. ATALA.

Lopez, Patrice, and David Roussel. 1998. Which Rules for the Robust Parsing of Spoken Utterances with Lexicalized Tree Adjoining Grammars? In Proceedings of the Fourth International Workshop on Tree Adjoining Grammars and Related Frameworks (TAG+4), 96–99. IRCS Report 98–12. Institute for Research in Cognitive Science, University of Pennsylvania.

Magerman, David M. 1995. Statistical Decision-Tree Models for Parsing. In Proceedings of the 33^{rd} Annual Meeting of the Association for Computational Linguistics.

Makino, Takaki, Minoru Yoshida, Kentaro Torisawa, and Jun'ichi Tsujii. 1998. LiLFeS — Towards a Practical HPSG Parser. In 36th Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics (COLING-ACL'98), 807–811. Montreal.

Maxwell, John T., and Ronald M. Kaplan. 1993. The Interface between Phrasal and Functional Constraints. Computational Intelligence 19(4):571–590.

McCord, Michael C. 1990. Slot Grammar: A System for Simpler Construction of Practical Natural Language Grammars. In Natural Language and Logic: Proc. of the International Scientiøc Symposium, Hamburg, Germany, ed. R. Studer. 118–145. Berlin, Heidelberg: Springer.

McDonald, David D., and Marie W. Meteer. 1990. The implications of Tree Adjoining Grammar to Generation. In Natural Language Generation, ed. G. Kempen. Dordrecht.

McDonald, David D., and James Pustejovsky. 1985. TAG's as a Grammatical Formalism for Generation. In Proceedings of the 23th Annual Meeting of the Association for Computational Linguistics (ACL'85). Chicago.

Mel'čuk, Igor A. 1988. Dependency Syntax: Theory and Practice. New York: State University of New York Press.

Meteer, Marie. 1992. Expressibility and the Problem of EŒcient Text Planning. Communication in Artificial Intelligence Series. Pinter.

Minnen, Guido. 1994. Predictive Left-to-Right Parsing of a Restricted Variant of TAG (LD/LP). Computational Intelligence 10(4):535–548.

Moortgat, Michael. 1988. Categorial Investigations: Logical and Linguistic Aspects of Lambek Calculus. Dordrecht, Netherlands.: Foris.

Morrill, G. 1994. Type Logical Grammar. Dordrecht, Netherlands.: Kluwer Academic Publishers.

Nasr, Alexis. 1995. A Formalism and a Parser for Lexicalised Dependency Grammars. In 4th International Workshop on Parsing Technologies, 186–195. Prague.

Nederhof, Mark-Jan. 1998. An alternative LR algorithm for TAGs. In 36th Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics (COLING-ACL'98), 946–952. Montréal, Canada.

Nederhof, Mark-Jan. 1999. Models of Tabulation for TAG Parsing. In Proceedings MOL'6, 143–158.

Nederhof, Mark-Jan, Anoop Sarkar, and Giorgio Satta. 1998. Prefix Probabilities from Stochastic Tree Adjoining Grammars. In 36th Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics (COLING-ACL'98), 953–959. Montreal.

Neumann, Günther. 1998. Automatic Extraction of Stochastic Lexicalized Tree Adjoining Grammars from Treebanks. In Proceedings of the Fourth International Workshop on Tree Adjoining Grammars and Related Frameworks (TAG+4), 120–123. IRCS Report 98–12. Institute for Research in Cognitive Science, University of Pennsylvania.

Nicolov, Nicolas, and Christopher Mellish. 2000. PROTECTOR: Efficient Generation with Lexicalized Grammars. In Recent Advances in Natural Language Processing (RANLP vol.II), ed. Ruslan Mitkov and Nicolas Nicolov. 221–243. Amsterdam and Philadelphia: John Benjamins.

Palmer, Martha, and Joseph Rosenzweig. 1996. Capturing Motion Verb Generalizations with Synchronous TAGs. In Proceedings of AMTA-96. Montreal, Quebec, October.

Paroubek, Patrick, Yves Schabes, and Aravind K. Joshi. 1992. XTAG – A Graphical Workbench for Developing Tree-Adjoining Grammars. In Third Conference on Applied Natural Language Processing. Trento, Italy.

Peters, S., and R. W. Ritchie. 1973. On the Generative Power of Transformational Grammars. Inf. Sci. 6.

Pollard, Carl. 1984. Generalized phrase structure grammars, head grammars and natural language. Doctoral dissertation, Stanford University, Stanford, CA.

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

Pollard, Carl, and Ivan Sag. 1994. Head-Driven Phrase Structure Grammar. Chicago: University of Chicago Press.

Poller, Peter. 1994. Incremental Parsing with LD/TLP–TAGs. Computational Intelligence 10:4:549–562.

Poller, Peter, and Tilman Becker. 1998. Two-step TAG Parsing Revisited. In Proceedings of the Fourth International Workshop on Tree Adjoining Grammars and Related Frameworks (TAG+4), 143–146. Institute for Research in Cognitive Science (IRCS), University of Pennsylvania, Philadelphia, PA, USA.

Prolo, Carlos A. 2000. An efficient LR parser generator for Tree Adjoining Grammars. In Proceedings of the 6th International Workshop on Parsing Technologies. Trento, Italy.

Rambow, Owen. 1994a. Formal and Computational Aspects of Natural Language Syntax. Doctoral dissertation, Department of Computer and Information Science, University of Pennsylvania, Philadelphia. Available as Technical Report 94-08 from the Institute for Research in Cognitive Science (IRCS).

Rambow, Owen. 1994b. Multiset-Valued Linear Index Grammars. In 32nd Meeting of the Association for Computational Linguistics (ACL'94). ACL.

Rambow, Owen. 1996. Word Order, Clause Union, and the Formal Machinery of Syntax. In Proceedings of the First LFG Conference, ed. Miriam Butt and Tracy Holloway King. On-line version at http://www-csli.stanford.edu/publications/LFG/lfg1.html.

Rambow, Owen, and Aravind Joshi. 1994a. A Formal Look at Dependency Grammars and Phrase-Structure Grammars, with Special Consideration of Word-Order Phenomena. In Current Issues in Meaning-Text Theory, ed. Leo Wanner. London: Pinter. To appear.

Rambow, Owen, and Aravind Joshi. 1994b. A Processing Model for Free Word Order Languages. In Perspectives on Sentence Processing, ed. Jr. Charles Clifton, Lyn Frazier, and Keith Rayner. 267–301. Hillsdale, NJ: Lawrence Erlbaum Associates.

Rambow, Owen, and Young-Suk Lee. 1994. Word Order Variation and Tree Adjoining Grammar. Computational Intelligence 10(4):386–400.

Rambow, Owen, and Giorgio Satta. 1996. Synchronous Models of Language. In 34th Meeting of the Association for Computational Linguistics (ACL'96), 116–123. ACL.

Rambow, Owen, K. Vijay-Shanker, and David J. Weir. 1995. D-Tree Grammars. In 33rd Meeting of the Association for Computational Linguistics (ACL'95), 151–158. ACL.

Resnik, Philip. 1992. Probabilistic Tree-Adjoining Grammar as a Framework for Statistical Natural Language Processing. In Proceedings of the Fourteenth International Conference on Computational Linguistics (COLING '92). Nantes, France, July.

Roche, Emmanuel, and Yves Schabes (ed.). 1997. Finite-State Language Processing. Cambridge, Mass.: MIT Press.

Rogers, James. 1994. Capturing CFLs With Tree Adjoining Grammars. In 32nd Meeting of the Association for Computational Linguistics (ACL'94). ACL.

Rogers, Jim, and K. Vijay-Shanker. 1994. Obtaining Trees from their Descriptions: An Application to Tree-Adjoining Grammar. Computational Intelligence 10(4):471–485.

Sag, Ivan. 1997. English relative clauses. Journal of linguistics 33(2):431–483.

Sarkar, Anoop. 1998. Conditions on Consistency of Probabilistic Tree Adjoining Grammars. In 36th Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics (COLING-ACL'98), 1164–1170. Montreal.

Satta, Giorgio. 1994. Tree-Adjoining Grammar Parsing and Boolean Matrix Multiplication. Computational Linguistics 20(2):173–192.

Satta, Giorgio, and William Schuler. 1998. Restrictions on Tree Adjoining Languages. In 36th Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics (COLING-ACL'98), 1176–1182. Montreal.

Schabes, Yves. 1989. Computational and Mathematical Studies of Lexicalized Grammars. Technical report. Department of Computer and Information Science, University of Pennsylvania: Department of Computer and Information Science.

Schabes, Yves. 1990. Mathematical and Computational Aspects of Lexicalized Grammars. Doctoral dissertation, Department of Computer and Information Science, University of Pennsylvania.

Schabes, Yves. 1992. Stochastic Lexicalized Tree-Adjoining Grammars. In Proceedings of the 14th International Conference on Computational Linguistics (COLING'92).

Schabes, Yves. 1994. Left-to-Right Parsing in Lexicalized Tree-Adjoining Grammars. Computational Intelligence 10(4):506–524.

Schabes, Yves, and Aravind K. Joshi. 1988. An Early-Type Parsing Algorithm for Tree Adjoining Grammars. In Proceedings of the 26^{th} Meeting of the Association for Computational Linguistics. Buffalo, June.

Schabes, Yves, and Stuart B. Shieber. 1994. An Alternative Conception of Tree-Adjoining Derivation. Computational Linguistics 1(20):91–124.

Schabes, Yves, and K. Vijay-Shanker. 1990. Deterministic Left to Right Parsing of Tree Adjoining Languages. In 28th Meeting of the Association for Computational Linguistics (ACL'90). Pittsburgh.

Schabes, Yves, and Richard C. Waters. 1995. Tree insertion grammar: a cubic-time, parsable formalism that lexicalizes context-free grammar without changing the trees produced. Computational Linguistics 21(4):479–513.

Schuler, William. 1999. Preserving Semantic Dependencies in Synchronous Tree Adjoining Grammar. In 37th Meeting of the Association for Computational Linguistics (ACL'99), 88–95. College Park, MD.

Seki, Hiroyuki, Takashi Matsumura, Mamoru Fujii, and Tadao Kasami. 1991. On multiple context-free grammars. Theoretical Computer Science 88:191–229.

Seki, Hiroyuki, Ryichi Nakanishi, Yuichi Kaji, Sachiko Ando, and Tadao Kasami. 1993. Parallel Multiple Context-Free Grammars, Finite State Translation Systems, and Polynomial-Time Recognizable Subclasses of Lexical-Functional Grammar. In 31st Meeting of the Association for Computational Linguistics (ACL'93), 121–129. Columbus, OH. ACL.

Shieber, Stuart B. 1985. Evidence against the context-freeness of natural language. Linguistics and Philosophy 8:333–343.

Shieber, Stuart B. 1994. Restricting the Weak Generative Capacity of Synchronous Tree Adjoining Grammar. Computational Intelligence 10(4):371–385.

Shieber, Stuart B., and Yves Schabes. 1990. Synchronous Tree Adjoining Grammars. In Proceedings of the 13th International Conference on Computational Linguistics. Helsinki.

Shieber, Stuart B., and Yves Schabes. 1991. Generation and Synchronous Tree Adjoining Grammars. Computational Intelligence 4(7):220–228.

Srinivas, B., Christine Doran, and Seth Kulick. 1995. Heuristics and Parse Ranking. In Proceedings of the 4^{th} Annual International Workshop on Parsing Technologies. Prague, September.

Srinivas, B., and Aravind Joshi. 1999. Supertagging: An Approach to Almost Parsing. Computational Linguistics 25(2):237–266.

Steedman, Mark. 1990. Gapping as Constituent Coordination. Linguistics and Philosophy 13:207–263.

Steedman, Mark. 1996. Surface Structure and Interpretation. MIT Press.

Steedman, Mark. 1998. The Syntactic Process. MIT Press.

Stone, Matthew, and Christine Doran. 1997. Sentence planning Using TAG. In Proceedings of the Joint conference ACL/EACL '97. Madrid, Spain.

Stone, Matthew, and Bonnie Webber. 1998. Textual Economy through Close Coupling of Syntax and Semantics. In Proceedings INLG'98, 178–187.

van Noord, Gertjan. 1994. Head-Corner Parsing for TAG. Computational Intelligence 10(4):525–534.

Vijay-Shanker, K. 1987. A study of Tree Adjoining Grammars. Doctoral dissertation, Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA, December.

Vijay-Shanker, K. 1992. Using Descriptions of Trees in a Tree Adjoining Grammar. Computational Linguistics 18(4):481–518.

Vijay-Shanker, K., and Aravind K. Joshi. 1989. Long distance dependencies in LFG and TAG. In 27th Meeting of the Association for Computational Linguistics (ACL'89). Vancouver, B.C.

Vijay-Shanker, K., and Yves Schabes. 1992. Structure Sharing in Lexicalized Tree-Adjoining Grammars. In Proceedings of the 14th International Conference on Computational Linguistics (COLING'92), 205–211.

Vijay-Shanker, K., and David J. Weir. 1993. The Use of Shared Forests in Tree Adjoining Grammar Parsing. In Sixth Conference of the European Chapter of the Association for Computational Linguistics (EACL'93), 384–393. Utrecht.

Vijay-Shanker, K., and David J. Weir. 1994. The Equivalence of Four Extensions of Context-Free Grammars. Math. Syst. Theory 27:511–546.

Vijay-Shanker, K., David J. Weir, and Aravind K. Joshi. 1987. Characterizing structural descriptions produced by various grammatical formalisms. In 25th Meeting of the Association for Computational Linguistics (ACL'87). Stanford, CA.

Wahlster, W., E. Andre, W. Finkler, H.-J. Profitlich, and T. Rist. 1993. Plan-Based Integration of Natural Language and Graphics Generation. Artiøcial Intelligence 63:387–427.

Webber, Bonnie L., and Aravind K. Joshi. 1998. Anchoring a Lexicalized Tree-Adjoining Grammar for Discourse. In Proceedings of COLING-ACL'98 Workshop on Discourse Relations and Discourse Markers. Montreal.

Weir, David J. 1988. Characterizing Mildly Context-Sensitive Grammar Formalisms. Doctoral dissertation, Department of Computer and Information Science, University of Pennsylvania.

Xia, Fei, Martha Palmer, K.Vijay-Shanker, and Joseph Rosenzweig. 1998. Consistent Grammar Development Using Partial-Tree Descriptions for Lexicalized Tree-Adjoining Grammars. In Proceedings of the Fourth International Workshop on Tree Adjoining Grammars and Related Frameworks (TAG+4), 180–183. IRCS Report, No. 98–12. Institute for Research in Cognitive Science, University of Pennsylvania.

XTAG-Group, The. 1999. A Lexicalized Tree Adjoining Grammar for English. Technical Report `http://www.cis.upenn.edu/~xtag/tech-report/tech-report.html`. The Institute for Research in Cognitive Science, University of Pennsylvania.

Yang, Gijoo, Kathleen F. McCoy, and K. Vijay-Shanker. 1991. From Functional Specification to Syntactic Structures: Systemic Grammar and Tree Adjoining Grammar. Computational Intelligence 4(7):207–219.

Zaenen, Annie, and Ronald M. Kaplan. 1995. Formal devices for linguistic generalizations: West Germanic word order in LFG. In Formal Issues in Lexical-Functional Grammar, ed. Mary Dalrymple, Ronald M. Kaplan, John Maxwell, and Annie Zaenen. 215–239. Stanford, CA: CSLI Publications.

Index