D-Tree Substitution Grammars

Owen Rambow*
AT&T Labs–Research

K. Vijay-Shanker†
University of Delaware

David Weir‡
University of Sussex

There is considerable interest among computational linguists in lexicalized grammatical frameworks; lexicalized tree adjoining grammar (LTAG) is one widely studied example. In this paper, we investigate how derivations in LTAG can be viewed not as manipulations of trees but as manipulations of tree descriptions. Changing the way the lexicalized formalism is viewed raises questions as to the desirability of certain aspects of the formalism. We present a new formalism, d-tree substitution grammar (DSG). Derivations in DSG involve the composition of d-trees, special kinds of tree descriptions. Trees are read off from derived d-trees. We show how the DSG formalism, which is designed to inherit many of the characterestics of LTAG, can be used to express a variety of linguistic analyses not available in LTAG.

1. Introduction

There is considerable interest among computational linguists in lexicalized grammatical frameworks. From a theoretical perspective, this interest is motivated by the widely held assumption that grammatical structure is projected from the lexicon. From a practical perspective, the interest stems from the growing importance of word-based corpora in natural language processing. Schabes (1990) defines a lexicalized grammar as a grammar in which every elementary structure (rules, trees, etc.) is associated with a lexical item and every lexical item is associated with a finite set of elementary structures of the grammar. Lexicalized tree adjoining grammar (LTAG) (Joshi and Schabes 1991) is a widely studied example of a lexicalized grammatical formalism.[1]

In LTAG, the elementary structures of the grammar are phrase structure trees. Because of the extended domain of locality of a tree (as compared to a context-free string rewriting rule), the elementary trees of an LTAG can provide possible syntactic contexts for the lexical item or items that anchor the tree, i.e., from which the syntactic structure in the tree is projected. LTAG provides two operations for combining trees: substitution and adjunction. The substitution operation appends one tree at a frontier node of another tree. The adjunction operation is more powerful: it can be used to insert one tree within another. This property of adjoining has been widely used in the LTAG literature to provide an account for long-distance dependencies. For example,

* ATT Labs-Research, B233 180 Park Ave, PO Box 971, Florham Park, NJ 07932-0971, USA. E-mail: rambow@research.att.com
† Department of Computer and Information Science University of Delaware Newark, Delaware 19716. E-mail: vijay@udel.edu
‡ School of Cognitive and Computing Sciences University of Sussex Brighton, BN1 6QH E. Sussex UK. E-mail: david.weir@cogs.susx.ac.uk

1 Other examples of lexicalized grammar formalisms include different varieties of categorial grammars and dependency grammars. Neither HPSG nor LFG are lexicalized in the sense of Schabes (1990).

Downloaded from http://www.mitpressjournals.org/doi/pdf/10.1162/089120101300346813 by guest on 28 March 2024

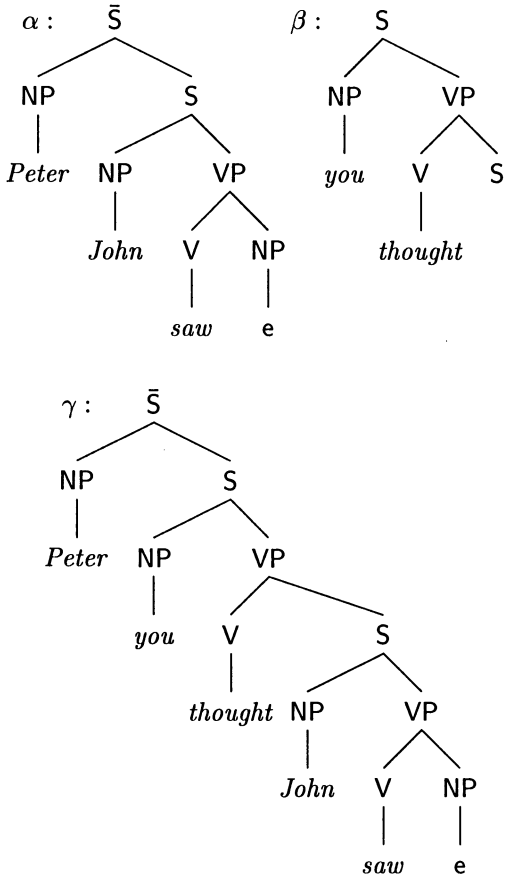


Figure 1
Example of adjunction.

Figure 1 shows a typical analysis of topicalization.[2] The related nodes for the filler and the gap in the elementary tree α are moved further apart when the tree γ is obtained by adjoining the auxiliary tree β within α. This shows that adjunction changes the structural relationship between some of the nodes in the tree into which adjunction occurs.

In LTAG, the lexicalized elementary objects are defined in such a way that the structural relationships between the anchor and each of its dependents change during the course of a derivation through the operation of adjunction, as just illustrated. This approach is not the only possibility. An alternative would be to define the relationships between the nodes of the elementary objects in such a way that these relationships hold throughout the derivation, regardless of how the derivation proceeds.

This perspective on the LTAG formalism was explored in Vijay-Shanker (1992) where, following the principles of d-theory parsing (Marcus, Hindle, and Fleck 1983), LTAG was seen as a system manipulating *descriptions* of trees rather than as a tree

2 The same analysis holds for *wh*-movement, but we use topicalization as an example in order to avoid the superficial complication of the auxiliary needed in English questions. Sometimes, topicalized sentences sound somewhat less natural than the corresponding *wh*-questions, which are always structurally equivalent.

Downloaded from http://www.mitpressjournals.org/doi/pdf/10.1162/089120101300346813 by guest on 28 March 2024

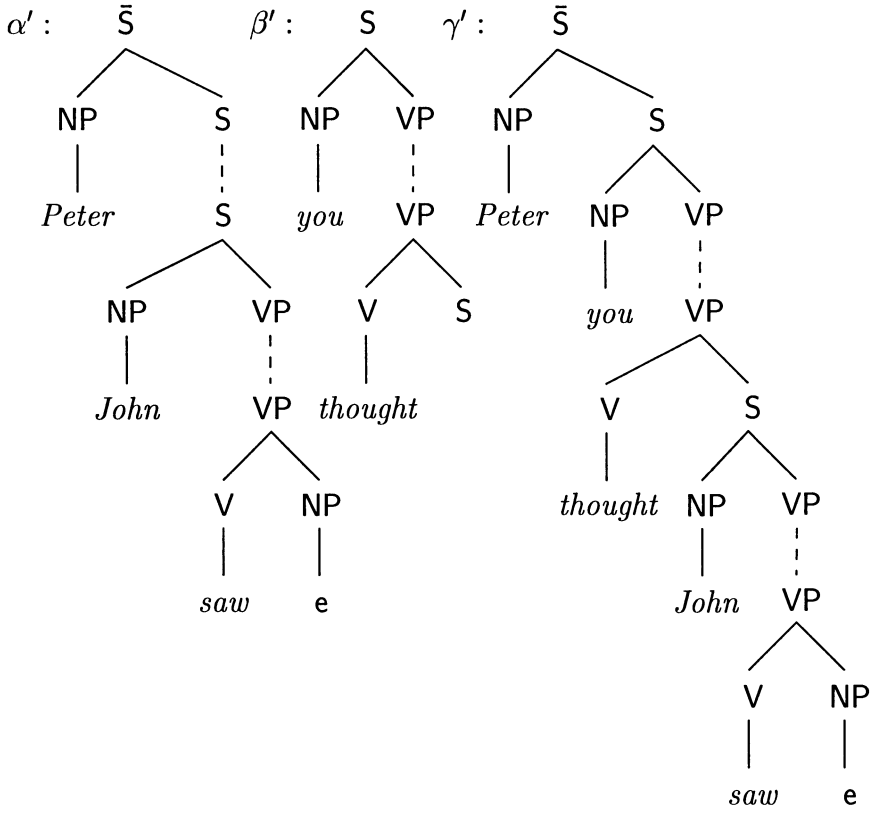


Figure 2
Adjunction example revisited.

rewriting formalism. Elementary objects are descriptions of possible syntactic contexts for the anchor, formalized in a logic for describing nodes and the relationships (dominance, immediate dominance, linear precedence) that hold between them.

From this perspective, instead of positing the elementary tree α in Figure 1, we can *describe* the projection of syntactic structure from the transitive verb. This description is presented pictorially as α' in Figure 2. The solid lines indicate immediate domination, whereas the dashed lines indicate a domination of arbitrary length. The description α' not only partially describes the tree α (by taking the dominations to be those of length 0) but also any tree (such as γ) that can be derived by using the operations of adjunction and substitution starting from α. In fact, α' describes exactly what is common among these trees.

By expressing elementary objects in terms of tree descriptions, we can describe syntactic structure projected from a lexical item in a way that is independent of the derivations in which it is used. This is achieved by employing composition operations that produce descriptions that are compatible with the descriptions being combined. For instance, adjoining, seen from this perspective, serves to further specify the underspecified dominations. In Figure 2, the description γ' is obtained by additionally stating that the domination between the two nodes labeled S in α' is now given by the domination relation between the two nodes labeled S in β'.

As we will explore in this paper, changing the way the lexicalized formalism is viewed, from tree rewriting to tree descriptions, raises questions as to the desirability

Downloaded from http://www.mitpressjournals.org/doi/pdf/10.1162/089120101300346813 by guest on 28 March 2024

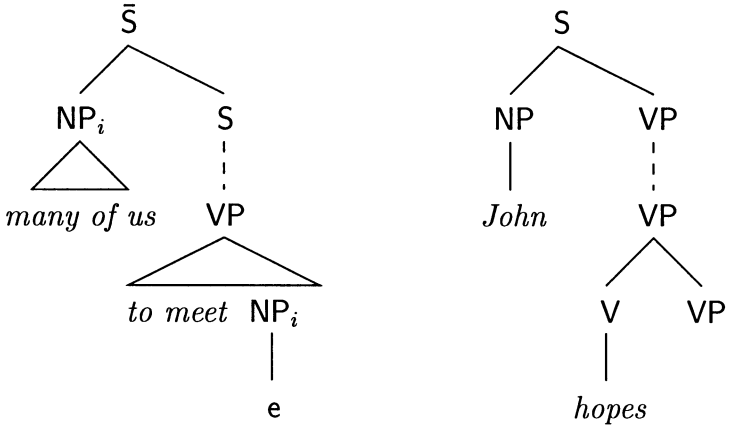


Figure 3
A problem for LTAG.

of certain aspects of the formalism. Specifically, we claim that the following two aspects of LTAG appear unnecessarily restrictive from the perspective of tree description:

1. In LTAG, the root and foot of auxiliary trees must be labeled by the same nonterminal symbol. This is not a minor issue since it derives from one of the most fundamental principles of LTAG, **factoring of recursion**. This principle states that auxiliary trees express factored out recursion, which can be reintroduced via the adjunction operation. It has had a profound influence on the way that the formalism has been applied linguistically.[3] An example of how this can create problems is shown in Figure 3. In this case, the "adjoined" tree has a root labeled S and a foot labeled VP, something that is not permissible in LTAG. Note that without this constraint, the combination would appear to be exactly like adjoining. We consider this aspect in more detail in Section 4.1.

2. The adjunction operation embeds all of the adjoined tree within that part of the tree at which adjunction occurs. This is illustrated in γ' (Figure 2) where both parts (separated by domination) of β' appear within one underspecified domination relationship in α'.
The foot node of tree β in Figure 1 corresponds to a required argument of the lexical anchor, *thought*. The adjunction operation accomplishes the role of expanding this argument node. Unlike the substitution operation, where an entire tree is inserted below the argument node, with adjunction, only a subtree of α appears below the argument node; the remainder appears in its entirety above the root node of β. However, if we view the trees as descriptions, as in Figure 2, and if we take the expansion of the foot node as the main goal served by adjunction, it is not clear why the composition should have anything to say about the domination relationship between the other parts of the two objects being combined. In the description approach, in order to obtain γ' we (in a

3 Note that in feature-based LTAG there is no restriction that the two feature structures be the same, or even that they be compatible.

Downloaded from http://www.mitpressjournals.org/doi/pdf/10.1162/089120101300346813 by guest on 28 March 2024

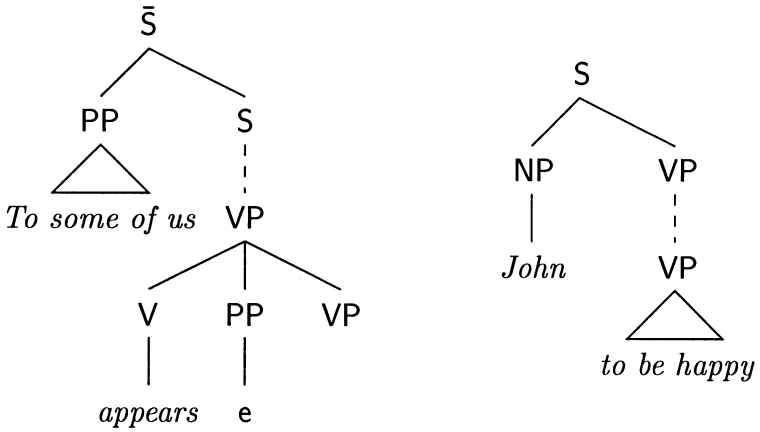


Figure 4
Another problem for LTAG.

sense to be made precise later) substitute the second component of α' (rooted in S) at the foot node of β'. This operation does not itself entail any further domination constraints between the components of α' and β' that are not directly involved in the substitution, specifically, the top components of α' and β'. In the trees described it is possible for either one to dominate the other.[4] However, adjunction further stipulates that the rest of α' will appear above all of β'. This additional constraint makes certain analyses unavailable for the LTAG formalism (as is well known). For instance, given the two lexical projections in Figure 4, the subtrees must be interleaved in a fashion not available with adjoining to produce the desired result. This aspect of adjoining is the focus of the discussion in Section 4.2.

In this paper, we describe a formalism based on tree descriptions called d-tree substitution grammars (DSG).[5] The elementary tree descriptions in DSG can be used to describe lexical items and the grammatical structure they project. Each elementary tree description can be seen as describing two aspects of the tree structure: one part of the description specifies phrase structure rules for lexical projections, and a second part of the description states domination relationships between pairs of nodes. DSG inherits from LTAG the extended domain of locality of its elementary structures, and, in DSG as in LTAG, this extended domain of locality allows us to develop a lexicalized grammar in which lexical items project grammatical structure, including positions for arguments. But DSG departs from LTAG in that it does not include factoring of recursion as a constraint on the makeup of the grammatical projections. Furthermore, in DSG, arguments are added to their head by a single operation that we call **generalized substitution**, whereas in LTAG two operations are used: adjunction and substitution.

DSG is intended to be a simple framework with which it is possible to provide analyses for those cases described with LTAG as well as for various cases in which extensions of LTAG have been needed, such as different versions of multicomponent

4 Of course, the node labels further restrict possible dominance in this case.
5 This paper is based on Rambow, Vijay-Shanker, and Weir (1995), where DSG was called DTG (d-tree grammar).

Downloaded from http://www.mitpressjournals.org/doi/pdf/10.1162/089120101300346813 by guest on 28 March 2024

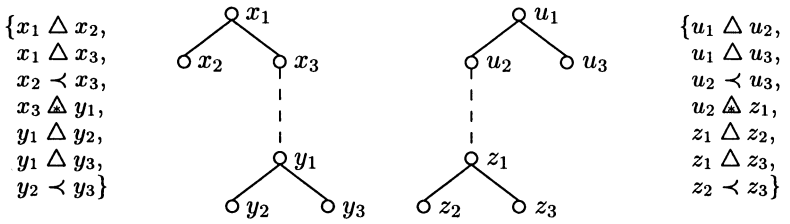


Figure 5
A pair of tree descriptions (which are also d-trees).

LTAG. Furthermore, because the elementary objects are expressed in terms of logical descriptions, it has been possible to investigate the characteristics of the underspecification that is used in these descriptions (Vijay-Shanker and Weir 1999).

In Section 2, we give some formal definitions and in Section 3 discuss some of the formal properties of DSG. In Section 4, we present analyses in DSG for various linguistic constructions in several languages, and compare them to the corresponding LTAG analyses. In Section 5, we discuss the particular problem of modeling syntactic dependency. We conclude with a discussion of some related work and summary.

2. Definition of DSG

D-trees are the primitive elements of a DSG. D-trees are descriptions of trees, in particular, certain types of expressions in a tree description language such as that of Rogers and Vijay-Shanker (1992). In this section we define tree descriptions and substitution of tree descriptions (Section 2.1) and d-trees (Section 2.2) together with some associated terminology and the graphical representation (Section 2.3). We then define d-tree substitution grammars, along with derivations of d-tree substitution grammars (Section 2.4) and languages generated by these grammars (Section 2.5), and close with an informal discussion of path constraints (Section 2.6).

2.1 Tree Descriptions and Substitution

In the following, we are interested in a tree description language that provides at least the following binary predicate symbols: $\triangle$, $\blacktriangle$, and $\prec$. These three predicate symbols are intended to be interpreted as the immediate domination, domination, and precedence relations, respectively. That is, in a tree model, the literal $x \triangle y$ would be interpreted as node (referred to by the variable) x immediately dominates node (referred to by) y, the literal $x \blacktriangle y$ would be interpreted such that x dominates y, and $x \prec y$ indicates that x is to the left of y. In addition to these predicate symbols, we assume there is a finite set of unary function symbols, such as label, which are to be used to describe node labeling. Finally, we assume the language includes the equality symbol.

We will now introduce the notion of tree description.

Definition

A **tree description** is a finite set (conjunction) of positive literals in a tree description language.

In order to make the presentation more readable, tree descriptions are usually presented graphically rather than as logical expressions. Figure 5 gives two tree descriptions, each presented both graphically and in terms of tree descriptions. We introduce

Downloaded from http://www.mitpressjournals.org/doi/pdf/10.1162/089120101300346813 by guest on 28 March 2024

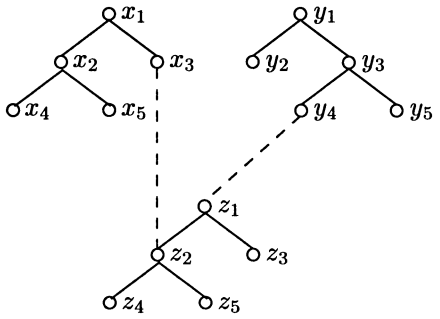


Figure 6
A tree description (which is also a d-tree) with three components.

the conventions used in the graphical representations in more detail in Section 2.3.

Note that with a functor for each feature, feature structure labels can be specified as required. Although feature structures will be used in the linguistic examples presented in Section 4, for the remainder of this section we will assume that each node is labeled with a symbol by the function label. Furthermore, we assume that these symbols come from two pairwise distinct sets of symbols, the terminal and nonterminal labels. (Note that the examples in this section do not show labels for nodes, but rather their names, while the examples in subsequent sections show the labels.)

In the following, we consider a tree description to be **satisfiable** if it is satisfied by a finite tree model. For our current purposes, we assume that a tree model will be defined as a finite universe (the set of nodes) and will interpret the predicate symbols: $\triangle$, $\trianglelefteq$, and $\prec$ as the immediate domination, domination, and precedence relations, respectively. For more details on the notion of satisfiability and the definition of tree models, see Backofen, Rogers, and Vijay-Shanker (1995), where the axiomatization of their theory is also discussed.[6]

We use $d \Rightarrow d'$ to indicate that the description d' logically follows from d, in other words, that d' is known in d.[7] Given a tree description d, we say x dominates y in d if $d \Rightarrow x \trianglelefteq y$ (similarly for the immediate domination and precedes relations).

We use $vars(d)$ to denote the set of variables involved in the description d. For convenience, we will also call the variables in $vars(d)$ the **nodes** of description d. For a tree description d, a node $x \in vars(d)$ is a **frontier node** of d if for all $y \in vars(d)$ such that $x \neq y$, it is not the case that $d \Rightarrow x \trianglelefteq y$. Only frontier nodes of the tree description can be labeled with terminals. A frontier node labeled with a nonterminal is called a **substitution node**.

A useful notion for tree description is the notion of components. Given a tree description d, consider the binary relation on $vars(d)$ corresponding to the immediate domination relations specified in d; i.e., the relation $\{\langle x, y\rangle \mid x, y \in vars(d), d \Rightarrow x \triangle y\}$. The transitive, symmetric, reflexive closure of this relation partitions $vars(d)$ into equivalence classes that we call **components**. For example, the nodes in the tree description in Figure 6 fall into the three components: $\{x_1, x_2, x_3, x_4, x_5\}$, $\{y_1, y_2, y_3, y_4, y_5\}$, and $\{z_1, z_2, z_3, z_4, z_5\}$. In particular, note that y_4 and z_1 (likewise x_3 and z_2) are not in the same components despite the fact that y_4 dominates z_1 is known in that de-

6 Note that the symbol $\trianglelefteq$ in this paper replaces the symbol $\triangle^*$ used in Backofen, Rogers, and Vijay-Shanker (1995).

7 In other words, $d \Rightarrow d'$ iff $d \wedge \neg d'$ is not satisfied by any tree model.

Downloaded from http://www.mitpressjournals.org/doi/pdf/10.1162/089120101300346813 by guest on 28 March 2024

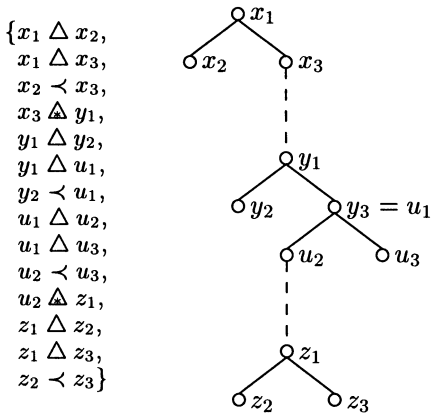


Figure 7
Result of substitution by tree description root.

scription. This is because the reflexive, symmetric, and transitive closure of the immediate domination relation known in the description will not include these pairs of nodes.

We say that x is the **root of a component** if it dominates every node in its component, and we say that x is on the **frontier of a component** if the only node in its component that it dominates is itself. Note that x can be on the frontier of a component of d without being a frontier node of a tree description. For example, in Figure 6, x_3 is a frontier of a component but not a frontier of the tree description. In contrast, z_3 is both a frontier of a component as well as a frontier of the tree description. We say that x is the **root of a tree description** if it dominates every node in the tree description. Note that it need not be the case that every tree description has a root. For example, according to the definition of tree descriptions, the description in Figure 6 is a tree description and does not have a root. Although we know that either x_1 or y_1 dominates all nodes in a tree model of the tree description, we don't know which.

We can now define the **substitution operation on tree descriptions** that will be used in DSG. We use $d_1[y/x]$ to denote the description obtained from d_1 by replacing all instances in d_1 of the variable x by y.

Definition
Let d_1 and d_2 be two tree descriptions. Without loss of generality, we assume that $vars(d_1) \cap vars(d_2) = \phi$. Let $x \in vars(d_1)$ be a root of a component of d_1 and $y \in vars(d_2)$ be a substitution node in the frontier of d_2. Let d be the description $d_1 \cup d_2[x/y]$. We say that d is obtained from d_1 and d_2 by **substituting** x **at** y.

Note that in addition, we may place restrictions on the values of the labeling functions for x and y in the above definition. Typically, for a node labeling function such as label we require $\mathsf{label}(x) = \mathsf{label}(y)$, and for functions that return feature structures we require unifiability (with the unification being the new value of the feature function for y).

Figure 7 shows the result of substituting the root u_1 of the tree description on the right of Figure 5 at the substitution node y_3 of the tree description on the left of Figure 5.

Figure 8 shows the result of substituting a node that is not the root of the tree

Downloaded from http://www.mitpressjournals.org/doi/pdf/10.1162/089120101300346813 by guest on 28 March 2024

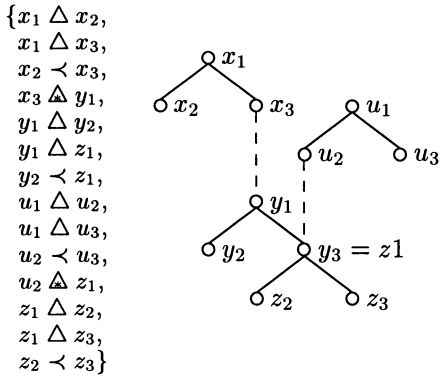


Figure 8
Result of substitution by component root.

description but the root z_1 of a *component* of the tree description on the right of Figure 5 at the substitution node y_3 of the tree description on the left of Figure 5.

2.2 D-Trees

D-trees are certain types of tree descriptions: not all tree descriptions are d-trees. In describing syntactic structure, we are interested in two kinds of **primitive tree descriptions**. The first kind of primitive tree description, which we call **parent-child descriptions**, involves $n+1$ ($n \geq 1$) variables, say $x, x_1, \ldots, x_n$, and in addition to specifying categorial information associated with these variables, specifies tree structure of the form

$$\{x \triangle x_1, \ldots, x \triangle x_n, x_1 \prec x_2, \ldots, x_{n-1} \prec x_n\}$$

A parent-child description corresponds to a phrase structure rule in a context-free grammar, and by extension, to a phrase structure rule in X-bar theory, to the instantiation of a rule schema in HPSG, or to a c-structure rule in LFG. As in a context-free grammar, in DSG we assume the siblings $x_1, \ldots, x_n$ are totally ordered by precedence.[8]

The second kind of primitive description, which we call a **domination description**, has the form $\{x \trianglelefteq y\}$, where x and y are variables. In projecting from a lexical item to obtain the elementary objects of a grammar, this underspecified domination statement allows for structures projected from other lexical items to be interspersed during the derivation process.

Definition

A **d-tree** is a satisfiable description in the smallest class of tree descriptions obtained by closing the primitive tree descriptions under the substitution operation.

For example, Figure 9 shows how the d-tree in Figure 6 is produced by using six parent-child descriptions and two domination descriptions. The ovals show cases of substitution; the circle represents a case of two successive substitutions. Figure 10 shows a tree description that is not a d-tree: it is not a parent-child description, nor

8 One could, of course, relax this constraint and assume that they are only partially ordered. However, for now, we do not consider such an extension. See Section 4.4 for a discussion.

Downloaded from http://www.mitpressjournals.org/doi/pdf/10.1162/089120101300346813 by guest on 28 March 2024

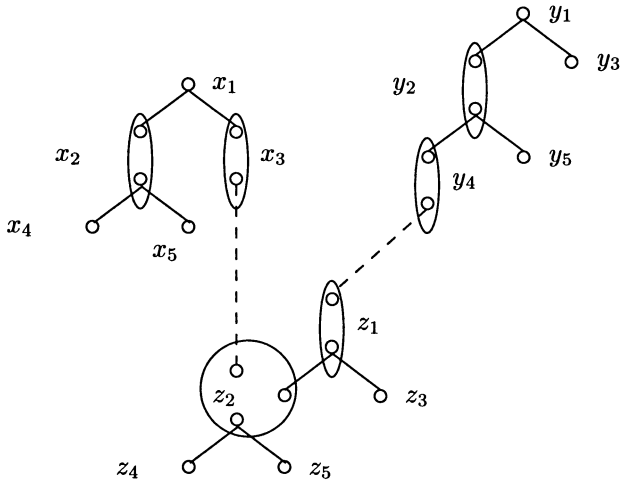


Figure 9
Derivation of an elementary d-tree.

$\{x \trianglelefteq y,$
$x \trianglelefteq z,$
$y \prec z\}$

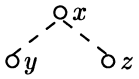


Figure 10
A description that is *not* a d-tree.

can it be derived from two domination descriptions by substitution, since substitution can only occur at the frontier nodes.

A d-tree d is **complete** if it does not contain any substitution nodes, i.e., all the frontier nodes of the description d are labeled by terminals. Given a d-tree d, we say that a pair of nodes, x and y (variables in $vars(d)$), are related by an **i-edge** if $d \Rightarrow x \triangle y$. We say that x is an **i-parent** and y is an **i-child**. Given a d-tree d, we say that a pair of nodes, x and y, are related by a **d-edge** if it is known from d that x dominates y, it is not known from d that x immediately dominates y, and there is no other variable in d that is known to be between them. That is, a pair of nodes x and y, $x \neq y$, are related by a d-edge if $d \Rightarrow x \trianglelefteq y$, $d \not\Rightarrow x \triangle y$, and for all $z \in vars(d)$, if $d \Rightarrow (x \trianglelefteq z \wedge z \trianglelefteq y)$ then $z = x$ or $z = y$. If x and y are related by a d-edge, then we say that they are **d-parent** and **d-child**, respectively. Note that a node in a d-tree (unlike a node in a tree description) cannot be both an i-parent and a d-parent at the same time.

2.3 Graphical Presentation of a D-Tree

We usually find it more convenient to present d-trees graphically. When presenting a d-tree graphically, i-edges are represented with a solid line, while d-edges are represented with a broken line. All immediate dominance relations are always represented graphically, but only the domination relations corresponding to d-edges are shown explicitly in graphical presentations.

By definition of d-trees, each component of a d-tree is fully specified with respect to immediate domination. Thus, all immediate domination relations between nodes in a component are indicated by i-edges. Also, by definition, components must be fully specified with respect to precedence. That is, for any two nodes u and v within

Downloaded from http://www.mitpressjournals.org/doi/pdf/10.1162/089120101300346813 by guest on 28 March 2024

a component we must know whether u precedes v or vice versa. In fact, *all* precedence information derives from precedence among siblings (two nodes immediately dominated by a common node). This means that all the precedence in a description can be expressed graphically simply by using the normal left-to-right ordering among siblings.

Another important restriction on d-trees has to do with how components are related to one another. As we said above, a frontier node of a component of a d-tree can be a d-parent but not an i-parent, and only frontier nodes of a component can serve as d-parents. However, by definition, a frontier node of a d-tree can neither be a d-parent nor an i-parent. Graphically, this restriction can be characterized as follows: edges specifying domination (d-edges) must connect a node on the *frontier of a component* with a node of another component. Furthermore, nodes on the frontier of a component can have at most one d-child.

Recall that not every set of positive literals involving $\triangle$, $\triangleq$, and $\prec$ is a legal d-tree. In particular, we can show that a description is a d-tree if and only if it is logically equivalent to descriptions that, when written graphically, would have the appearance described above.

2.4 D-Tree Substitution Grammars

We can now define d-tree substitution grammars.

Definition

A **d-tree substitution grammar** (DSG) G is a 4-tuple $(V_T, V_N, \Upsilon, d_S)$, where V_T and V_N are pairwise distinct terminal and nonterminal alphabets, respectively, Υ is a finite set of elementary d-trees such that the functor label assigns each node in each d-tree in Υ a label in $V_T \cup V_N$ and such that only d-tree frontier nodes take labels in V_T, and d_S is a characterization of the labels that can appear at the root of a derived tree.

Derivations in DSG are defined as follows. Let $G = (V_T, V_N, \Upsilon, d_S)$ be a DSG. Furthermore:

- Let $\Upsilon_0(G) = \Upsilon$.
- Let $\Upsilon_{i+1} = \Upsilon_i \cup \{d |\ d$ is satisfiable and d results from combining a pair of d-trees in Υ_i by substitution at a node x such that $\mathsf{label}(x) \in V_N\}$.

The **d-tree language** $\Upsilon(G)$ generated by G is defined as follows.

$$\Upsilon(G) = \{\, d \in \Upsilon_i \mid i \geq 0,\ d \text{ is complete} \,\}$$

In a **lexicalized** DSG, there is at least one terminal node on the frontier of every d-tree; this terminal is (these terminals are) designated the **anchor(s)** of the d-tree. The remaining frontier nodes (of the description) and all internal nodes are labeled by nonterminals. Nonterminal nodes on the *frontier of a description* are called **substitution** nodes because these are nodes at which a substitution must occur (see below). Finally, we say that a d-tree d is **sentential** if d has a single component and the label of the root of d is compatible with d_S.

2.5 Reading D-Trees

A description d is a tree if and only if it has a single component (i.e., it does not have any d-edges). Therefore, the process of reading off trees from d-trees can be viewed as

Downloaded from http://www.mitpressjournals.org/doi/pdf/10.1162/089120101300346813 by guest on 28 March 2024

a nondeterministic process that involves repeatedly removing d-edges until a d-tree with a single component results.

In defining the process of removing a d-edge, we require first, that no i-edges be added which are not already specified in the components, and second, that those i-edges that are distinct prior to the process of reading off remain distinct after the removal of the d-edges. This means that each removal of a d-edge results in equating exactly one pair of nodes. These requirements are motivated by the observation that the i-edges represent linguistically determined structures embodied in the elementary d-trees that cannot be created or reduced *during* a derivation.

We now define the **d-edge removal algorithm**. A d-edge represents a domination relation of length zero or more. Given the above requirements, at the end of the composition process, we can, when possible, get a *minimal* reading of a d-edge to be a domination relation of length zero. Thus, we obtain the following procedure for removing d-edges: Consider a d-edge with a node x as the d-parent and with a d-child y. By definition of d-trees, x is on the frontier of a component. The d-child y can either be a root of a component or not. Let us first consider the case in which y is a root of a component. To remove this d-edge, we equate x with y.[9] This gives us the minimal reading that meets the above requirement (that no i-edges are added which are not already specified in the components, and that those i-edges that are distinct prior to the process of reading off remain distinct after the removal of the d-edge). Now we consider the alternate case in which the d-child is not the root of its component. Let z be the root of the component containing y. Now both z and x are known to dominate y and hence in any model of the description, either z will dominate x or vice versa. Equating x with y (the two nodes in the d-edge under consideration) has the potential of requiring the collapsing of i-edges (e.g., i-edges between x and its parent and y and its parent in the component including z). As a consequence of our requirement, the only way to remove the d-edge is by equating the nodes x and z. If we equated x with any other node dominated by z (such as y), we would also be collapsing i-edges from two distinct components and equating more than one pair of nodes, contrary to our requirement. The removal of the d-edge by equating x and z can also be viewed as adding a d-edge from x to z (which, as mentioned, is compatible with the given description and does not have the potential for collapsing i-edges). Now since this d-edge is between a frontier of a component and the root of another, it can be removed by equating the two nodes.

Definition

A tree t can be **read off** from a d-tree d iff t is obtained from d by removing the d-edges of d in any order using the d-edge removal algorithm.

By selecting d-edges for removal in different orders, different trees can be produced. Thus, in general, we can read off several trees from each d-tree in $\Upsilon(G)$. For example, the d-tree in Figure 6 can produce two trees: one rooted in x_1 (if we choose to collapse the edge between y_4 and z_1 first) and one rooted in y_1 (if we choose to collapse the edge between x_3 and z_2 first). The fact that a d-tree can have several minimal readings can be exploited to underspecify different word orderings (see Section 4.4).

9 This additional equality to obtain the minimal readings is similar to unification of the so-called top and bottom feature structures associated with a node in tree adjoining grammars, which happens at the end of a derivation. In DSG, if the labeling specifications on x and y are incompatible, then the additional equality statement does not lead to any minimal tree model, just as in TAG, a derivation cannot terminate if the top and bottom feature structures associated with a node do not unify.

Downloaded from http://www.mitpressjournals.org/doi/pdf/10.1162/089120101300346813 by guest on 28 March 2024

Thus, while a single d-tree may describe several trees, only some of these trees will be read off in this way. This is because of our assumptions about what is being implicitly stated in a d-tree—for example, our requirement that i-edges can be neither destroyed nor created in a derivation. Assumptions such as these about the implicit content of d-trees constitute a theory of how to read off from d-trees. Variants of the DSG formalism can be defined, which differ with respect to this theory.

We now define the tree and string languages of a DSG.

Definition

Let G be a DSG. The **tree language** $T(G)$ generated by G is the set of trees that can be read off from sentential d-trees in $\Upsilon(G)$.

Definition

The **string language** generated by G is the set of terminal strings on the frontier of trees in $T(G)$.

2.6 DSG with Path Constraints

In DSG, domination statements are used to express domination paths of arbitrary length. There is no requirement placed on the nodes that appear on such paths. In this section, we informally define an extension to DSG that allows for additional statements constraining the paths.

Path constraints can be associated with domination statements to constrain which nodes, in terms of their labels, can or cannot appear within a path instantiating a d-edge.[10] Path constraints do not directly constrain the length of the domination path, which still remains underspecified. Path constraints are specified in DSG by associating with domination statements a set of labels that defines which nodes cannot appear within this path.[11] Suppose we have a statement $x \trianglelefteq y$ with an associated path constraint set, P, then logically this pair can be understood as $x \trianglelefteq y \wedge \forall z(z \neq x \wedge z \neq y \wedge x \trianglelefteq z \wedge z \trianglelefteq y) \rightarrow \mathsf{label}(z) \notin P$.

Note that during the process of derivation involving substitution, the domination statements in the two descriptions being composed continue to exist and do not play any role in the composition operation itself. The domination statements only affect the reading off process. For this reason, we can capture the effect of path constraints by merely defining how they affect the reading off process. Recall that the reading off process is essentially the elimination of d-edges to arrive at a single component d-tree. If there is a d-edge between x and y, we consider two situations: is the d-child y the root of a component, or not? When y is the root of a component, then x and y are collapsed. Clearly any path constraint on this d-edge has no effect. However, when y is not the root of a component, and z is the root of the component containing y, then the tree we obtain from the reading off process is one where x dominates z and not where z properly dominates x. That is, in this case, we replace the d-edge between x and y with a d-edge between x and z, which we then eliminate in the reading off process by equating x and z. But in order to replace the d-edge between x and y with a d-edge between x and z, we need to make sure that the path between z and y does not violate the path constraint associated with the d-edge between x and y.

10 In Rambow, Vijay-Shanker, and Weir (1995), path constraints are called "subsertion-insertion constraints."

11 Rambow (1996) uses regular expressions to specify path constraints.

Downloaded from http://www.mitpressjournals.org/doi/pdf/10.1162/089120101300346813 by guest on 28 March 2024

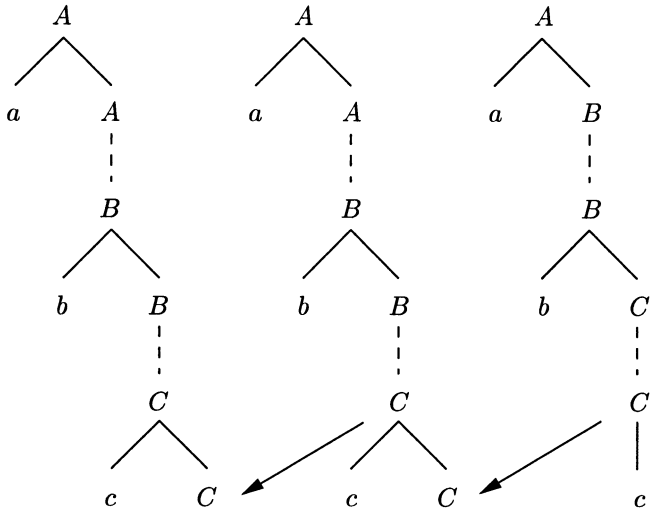


Figure 11
Counting to three: A derivation.

3. Properties of the Languages of DSG

It is clear that any context-free language can be generated by DSG (a context-free grammar can simply be reinterpreted as a DSG). It is also easy to show that the weak generative capacity of DSG exceeds that of context-free grammars. Figure 11 shows three d-trees (including two copies of the same d-tree) that generate the non-context-free language $\{ a^n b^n c^n \mid n \geq 1 \}$. Figure 12 shows the result of performing the first of two substitutions indicated by the arrows (top) and the result of performing both substitutions (bottom). Note that although there are various ways that the domination edges can be collapsed when reading off trees from this d-tree, the order in which we collapse domination edges is constrained by the need to consistently label nodes being equated. This is what gives us the correct order of terminals.

Figure 13 shows a grammar for the language

$$\{ w \in \{ a, b, c \}^* \mid w \text{ has an equal nonzero number of } a\text{'s, } b\text{'s and } c\text{'s} \},$$

which we call Mix. This grammar is very similar to the previous one. The only difference is that node labels are no longer used to constrain word order. Thus the domination edges can be collapsed in *any* order.

Both of the previous two examples can be extended to give a grammar for strings containing an equal number of any number of symbols simply by including additional components in the elementary d-trees for each symbol to be counted. Hence, DSG generates not only non-context-free languages but also non–tree adjoining languages, since LTAG cannot generate the language $\{ a^n b^n c^n d^n e^n \mid n \geq 1 \}$ (Vijay-Shanker 1987).

However, it appears that DSG cannot generate all of the tree adjoining languages, and we conjecture that the classes are therefore incomparable (we offer no proof of this claim in this paper). It does not appear to be possible for DSG to generate the copy language $\{ ww \mid w \in \{ a, b \}^* \}$. Intuitively, this claim can be motivated by the observation that nonterminal labels can be used to control the ordering of a bounded number of terminals (as in Figure 12), but this cannot be done in an unbounded way, as would be required for the copy language (since the label alphabet is finite).

Downloaded from http://www.mitpressjournals.org/doi/pdf/10.1162/089120101300346813 by guest on 28 March 2024

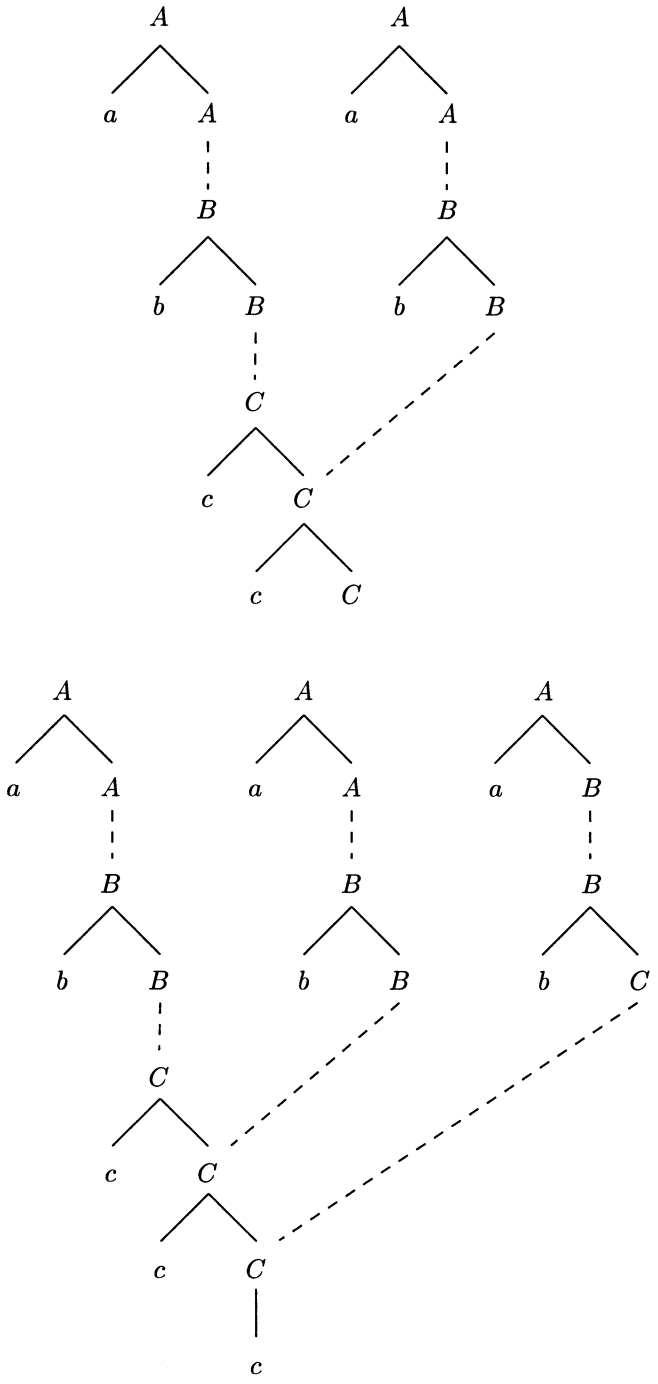


Figure 12
Counting to three: After substituting one tree (above) and the derived d-tree (below).

DSG is closely related (and weakly equivalent) to two equivalent string rewriting systems, UVG-DL and {}-LIG (Rambow 1994a, 1994b). In UVG-DL, several context-free rewrite rules are grouped into a set, and dominance links may hold between

Downloaded from http://www.mitpressjournals.org/doi/pdf/10.1162/089120101300346813 by guest on 28 March 2024

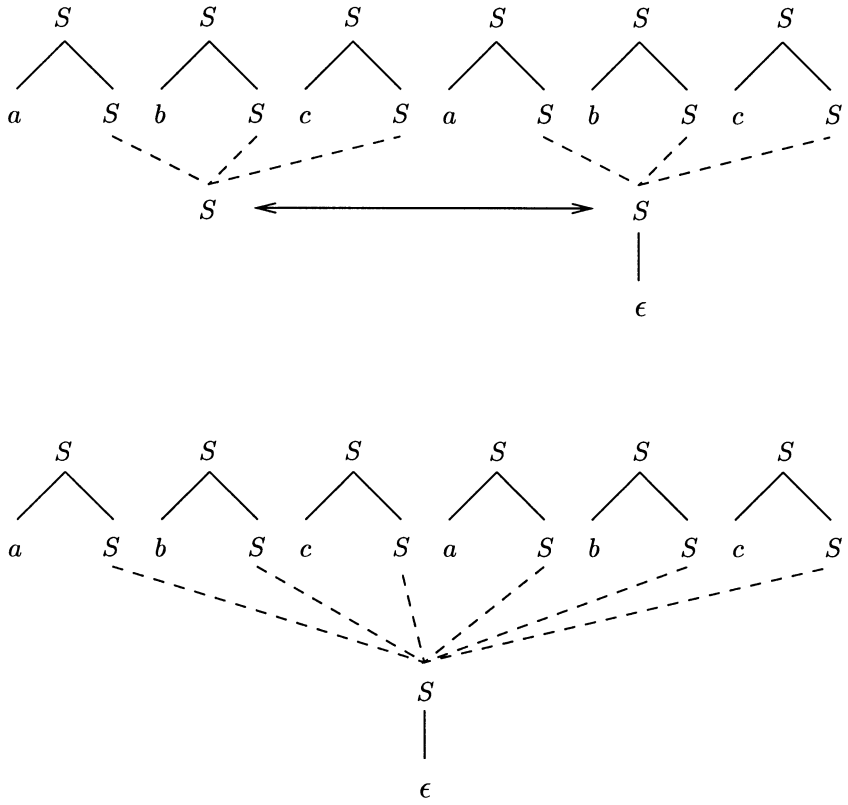


Figure 13
A grammar for Mix.

right-hand-side nonterminals and left-hand-side nonterminals of different rules from the same set. In a derivation, the context-free rules are applied as usual, except that all rules from an instance of a set must be used in the derivation, and at the end of the derivation, the dominance links must correspond to dominance relations in the derivation tree. {}-LIG is a multiset-valued variant of Linear Index Grammar (Gazdar 1988). UVG-DL and {}-LIG, when lexicalized, generate only context-sensitive languages.

Finally, Vijay-Shanker, Weir, and Rambow (1995), using techniques developed for UVG-DL (Rambow 1994a; Becker and Rambow 1995), show that the languages generated by lexicalized DSG can be recognized in polynomial time. This can be shown with a straightforward extension to the usual bottom-up dynamic programming algorithm for context-free grammars. In the DSG case, the nonterminals in the chart are paired with multisets. The nonterminals are used to verify that the immediate dominance relations (i.e., the parent-child descriptions) hold, just as in the case of CFG. The multisets record the domination descriptions whose lower (dominated) node has been found but whose upper (dominating) node still needs to be found in order for the parse to find a valid derivation of the input string (so-called open domination descriptions). The key to the complexity result is that the size of the multisets is linearly bounded by the length of the input string if the grammar is lexicalized, and the number of multisets of size n is polynomial in n. Furthermore, if the number of open domination descriptions in any chart entry is bounded by some constant independent

Downloaded from http://www.mitpressjournals.org/doi/pdf/10.1162/089120101300346813 by guest on 28 March 2024

of the length of the input string (as is plausible for many natural languages including English), the parser performs in cubic time.

4. Some Linguistic Analyses with DSG

In Section 1, we saw that the extended domain of locality of the elementary structures of DSG—which DSG shares with LTAG—allows us to develop lexicalized grammars in which the elementary structures contain lexical items and the syntactic structure they project. There has been considerable research in the context of LTAG on the issue of how to use the formalism for modeling natural language syntax—we mention as salient examples XTAG-Group (1999), a wide-coverage grammar for English, and Frank (1992, forthcoming), an extensive investigation from the point of view of theoretical syntax. Since DSG shares the same extended domain of locality as LTAG, much of this research carries over to DSG. In this section, we will be presenting linguistic analyses in DSG that follow some of the elementary principles developed in the context of LTAG. We will call these conventions the **standard LTAG practices** and summarize them here for convenience.

- Each elementary structure contains a lexical item (which can be multiword) and the syntactic structure it projects.
- Each elementary structure for a syntactic head contains syntactic positions for its arguments. (In LTAG, this means substitution or foot nodes; in DSG, this means substitution nodes.)
- When combining two elementary structures, a syntactic relation between their lexical heads is established. For example, when substituting the elementary structure for lexical item l_1 into an argument position of the elementary structure for lexical item l_2, then l_1 is in fact an argument of l_2.

In Section 1 we also saw that the adjoining operation of LTAG has two properties that appear arbitrary from a tree description perspective. The first property is the recursion requirement, which states that the root and foot of an auxiliary tree must be identically labeled. This requirement embodies the principle that auxiliary trees are seen as factoring recursion. The second property, which we will refer to as the **nesting property of adjunction**, follows from the fact that the adjoining operation is not symmetrical. All the structural components projected from one lexical item (corresponding to the auxiliary tree used in an adjoining step) are included entirely between two components in the other projected structure. That is, components of only one of the lexically projected structures can get separated in an adjoining step.

In this section, we examine some of the ramifications of these two constraints by giving a number of linguistic examples for which they appear to preclude the formulation of an attractive analysis. We show that the additional flexibility inherent in the generalized substitution operation is useful in overcoming the problems that arise.

4.1 Factoring of Recursion

We begin by explaining why, in LTAG, the availability of analyses for long-distance dependencies is limited by the recursion requirement. Normally, substitution is used in LTAG to associate a complement to its head, and adjunction is used to associate a modifier. However, adjunction rather than substitution must be used with complements involving long-distance dependencies, e.g., in *wh*-dependencies and raising

Downloaded from http://www.mitpressjournals.org/doi/pdf/10.1162/089120101300346813 by guest on 28 March 2024

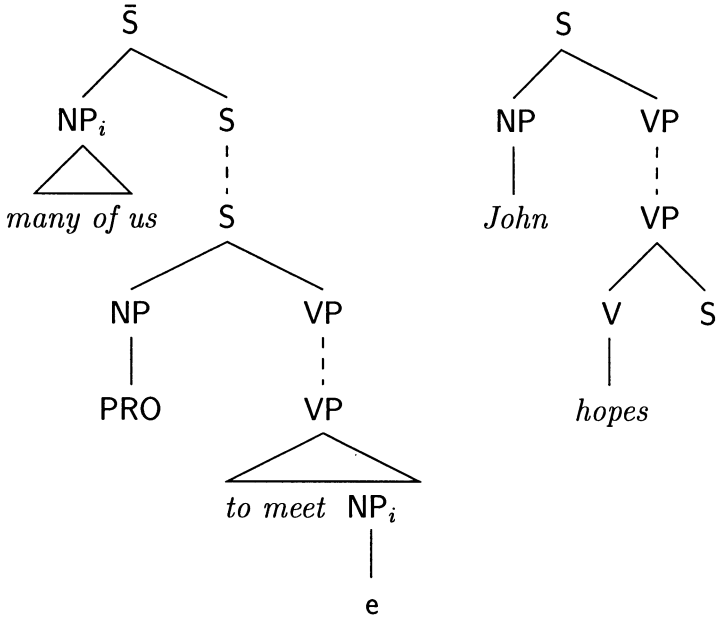


Figure 14
S-analysis for extraction from infinitival complements.

constructions. Such auxiliary trees are called **predicative auxiliary trees**.[12] In a predicative auxiliary tree, the foot node should be one of the nonterminal nodes on the frontier that is included due to argument requirements of the lexical anchor of the tree (as determined by its active valency). However, the recursion requirement means that all frontier nonterminal nodes that do not have the same label as the root node must be designated as substitution nodes, which may mean that no well-formed auxiliary tree can be formed.

Let us consider again the topicalized sentence used as an example in Section 1, repeated here for convenience:

(1) Many of us, John hopes to meet

A possible analysis is shown in Figure 3 in Section 1. We will refer to this analysis as the VP-complement analysis. Note that the individual pieces of the structures projected from lexical items follow standard LTAG practices. Because of the recursion requirement, the tree on the right is not (a description of) an auxiliary tree. To obtain an auxiliary tree in order to give a usual TAG-style account of long-distance dependencies, the complement of the *equi*-verb (control verb) *hopes* must be given an S label, which in turn imposes a linguistic analysis using an empty (PRO) subject as shown in Figure 14 (or, at any rate, an analysis in which the infinitival *to meet* projects to S).

The VP-complement analysis has been proposed within different frameworks, and has been adopted as the standard analysis in HPSG (Pollard and Sag 1994). However, because this would require an auxiliary tree rooted in S with a VP foot node, the recursion requirement precludes the adoption of such an analysis in LTAG. We are

12 This term is from Schabes and Shieber (1994). Kroch (1987) calls such trees complement auxiliary trees.

Downloaded from http://www.mitpressjournals.org/doi/pdf/10.1162/089120101300346813 by guest on 28 March 2024

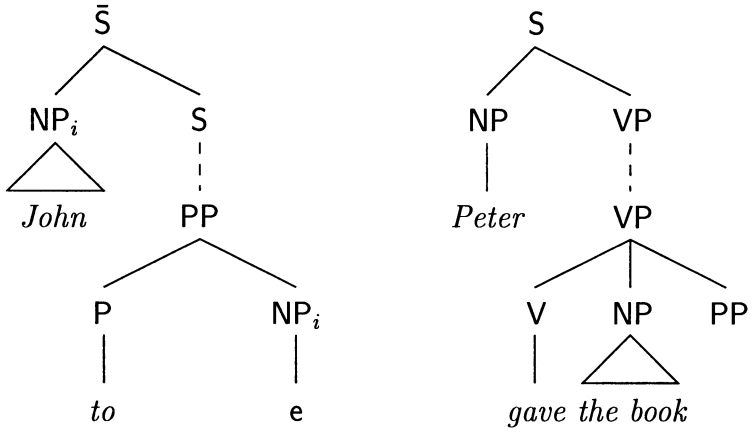


Figure 15
HPSG analysis of *give* expressed as trees.

not suggesting that one linguistic analysis is better than another, but instead we point out that the formal mechanism of LTAG precludes the adoption of certain linguistically motivated analyses. Furthermore, this mechanism makes it difficult to express entire grammars originally formulated in other formalisms in LTAG; for example, when compiling a fragment of HPSG into TAG (Kasper et al. 1995). In fact, the compilation produces structures just like those (described) in Figure 3. Kasper et al. (1995) consider the tree on the right of Figure 3 to be an auxiliary tree with the VP sibling of the anchor determined to be the foot node. Technically, the tree on the right of Figure 3 cannot be an auxiliary tree. Kasper et al. (1995) overcome the problem by making the node label a feature (with all nodes having a default label of no significance). This determination of the foot node is independent of the node labels of the frontier nodes. Instead, the foot node is chosen because it shares certain crucial features (other than label!) with the root node. These shared features are extracted from the HPSG rule schema and are used to define the localization of dependencies in the compiled TAG grammar. See Kasper et al. (1995) for details.

A similar example involves analyses for sentences such as (2), which involve extraction from argument PPs.

(2) John, Peter gave the book to

Figure 15 shows the structures obtained by using the method of Kasper et al. (1995) for compiling an HPSG fragment to TAG-like structures. In contrast to traditional TAG analyses (in which the elementary tree contains the preposition and its PP, with the NP complement of the preposition as a substitution node), the PP argument of the ditransitive verb is not expanded.[13] Instead the PP tree anchored by the preposition is substituted. However, because of the extraction, DSG's notion of substitution rather than LTAG substitution would need to be used.

These examples suggest that the method for compiling an HPSG fragment into TAG-like structures discussed in Kasper et al. (1995) can be simplified by compiling HPSG to a DSG-like framework.

13 Recall that we are not, in this section, advocating one analysis over another; rather, we are discussing the range of options available to the syntactician working in the TAG framework.

Downloaded from http://www.mitpressjournals.org/doi/pdf/10.1162/089120101300346813 by guest on 28 March 2024

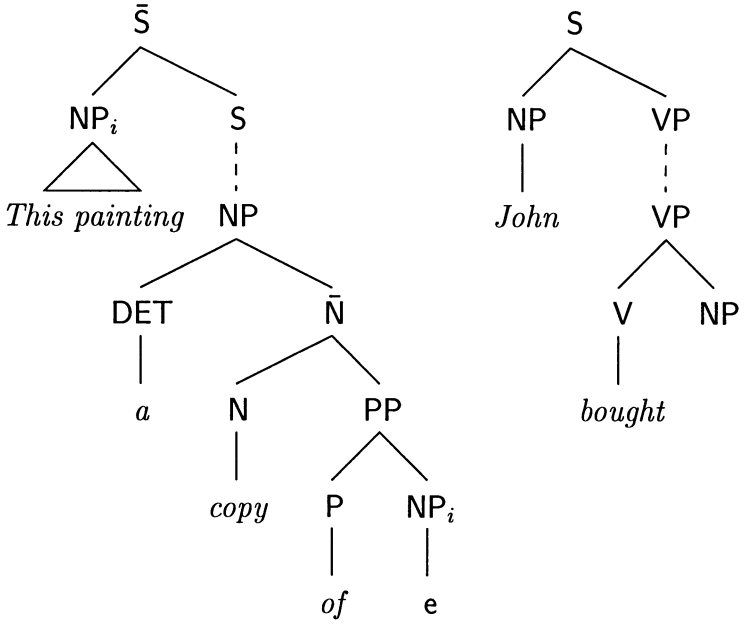


Figure 16
Extraction from *picture*-NPs.

We have shown a number of examples where some, but not all, of the possible linguistic analyses can be expressed in LTAG. It could be claimed that a formal framework limiting the range of possible analyses constitutes a methodological advantage rather than a disadvantage. However, as is well known, there are several other examples in English syntax where the factoring of recursion requirement in fact eliminates *all* plausible LTAG analyses. The only constraint assumed here is that extraction is localized in elementary trees. One such example in English is extraction out of a "*picture*-NP" (a noun which takes a prepositional complement from which extraction into the main sentence is possible), as illustrated in the following example:

(3) This painting, John bought a copy of

Following the standard LTAG practices, we would obtain the structures described in Figure 16. As these descriptions show, the recursion constraint means that adjoining cannot be used to provide this analysis of extraction out of NPs. See Kroch (1989) for various examples of such constructions in English and their treatment using an extension of TAG called multicomponent tree adjoining grammars. (We return to analyses using multicomponent TAG in Section 4.2.)

However, we now show that all of these cases can be captured uniformly with generalized substitution (see Figure 17). The node labeled X in β arises due to the argument requirements of the anchor (the verb) and when X = S, β is a predicative auxiliary tree in LTAG. The required derived phrase structure in these cases is described by γ. To obtain these trees, it would suffice to simply substitute the component rooted in X of α at the node labeled X in β. While in general, such a substitution would not constrain the placement of the upper component of β, because of the labels of the relevant nodes, this substitution will always result in γ. The use of substitution at argument nodes not only captures the situations where adjoining or multicompo-

Downloaded from http://www.mitpressjournals.org/doi/pdf/10.1162/089120101300346813 by guest on 28 March 2024

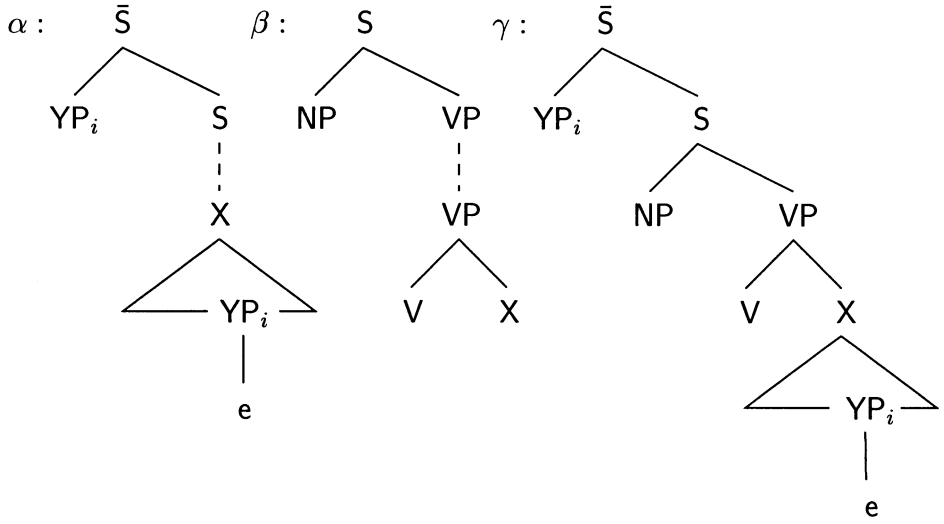


Figure 17
General case of extraction.

nent adjoining is used for these examples, it also allows the DSG treatment to be uniform, and is applicable even in cases where there is no extraction (e.g., the upper component of α is not present).

We end this discussion of the nature of foot nodes by addressing the question of how the choice of foot nodes limits illicit extractions. In the TAG approach, the designation of a foot node specifically rules out extraction from any structure that gets attached to any other frontier node (other arguments), or from structures that are adjoined in (adjuncts). However, as has been pointed out before (Abeillé 1991), the choice of foot nodes is not always determined by node labels alone, for example in the presence of sentential subjects or verbs such as *déduire*, which can be argued to have two sentential objects. In these cases some additional linguistic criteria are needed in order to designate the foot node. These same linguistic criteria can be used to designate frontier nodes from which extraction is possible; extraction can be regulated through the use of features. We also note that in moving to a multicomponent TAG analysis, an additional regulatory mechanism becomes necessary in any case to avoid extractions out of subjects (and, to a lesser degree, out of adjuncts). We refer the interested reader to Rambow, Vijay-Shanker, and Weir (1995) and Rambow and Vijay-Shanker (1998) for a fuller discussion.

4.2 Interspersing of Components

We now consider how the nesting constraint of LTAG limits the TAG formalism as a descriptive device for natural language syntax. We contrast this with the case of DSG, which, through the use of domination in describing elementary structures projected from a lexical item, allows for the interleaving of components projected from lexical items during a derivation.

Consider the raising example introduced in Section 1 repeated here as (4a), along with its nontopicalized version (4b), which indicates a possible original position for

Downloaded from http://www.mitpressjournals.org/doi/pdf/10.1162/089120101300346813 by guest on 28 March 2024

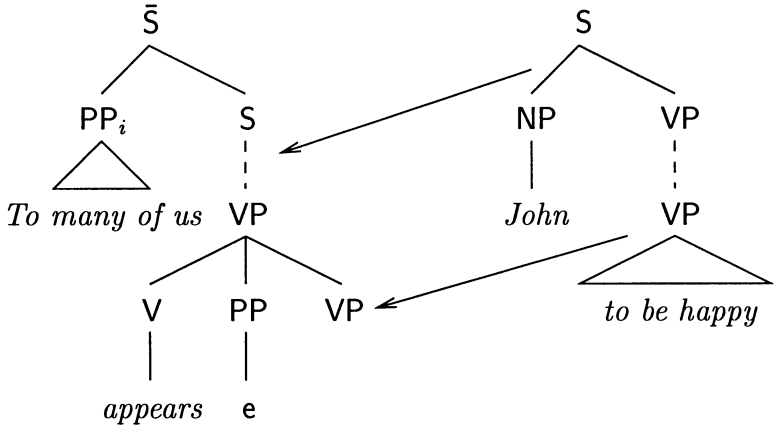


Figure 18
Topicalization out of the clause of a raising verb.

the topicalized phrase.[14]

(4) a. To many of us, John appears to be happy

b. John appears to many of us to be happy

Following standard LTAG practices of localizing argument structure (even in the presence of topicalization) and the standard LTAG analysis for the raising verb *appear*, the descriptions shown in Figure 18 could be proposed. Because of the nesting property of adjunction, the interleaving required to obtain the relevant phrase structure for the sentence (4a) cannot be realized using LTAG with the assumed lexical projections (or any other reasonable structures where the topicalized PP and the verb *appear* are in the same projected structure). In contrast, with these projections, using generalized substitution in DSG (i.e., equating the VP argument node of the verb and the root of the infinitival VP), the only possible derived tree is the desired one.

We will now consider an example that does not involve a *wh*-type dependency:

(5) Didn't John seem to like the gift?

Following the principles laid out in Frank (1992) for constructing the elementary trees of TAG, we would obtain the projections described in Figure 19 (except for the node labels). Note in particular the inclusion of the auxiliary node with the cliticized negation marker in the projection of the raising verb *seem*. Clearly the TAG operations could never yield the necessary phrase structure given this localization. Once again, the use of generalized substitution in DSG would result in the desired phrase structure.

An alternative to the treatment in Frank (1992) is implemented in the XTAG grammar for English (XTAG-Group 1999) developed at the University of Pennsylvania. The XTAG grammar does not presuppose the inclusion of the auxiliary in the projection of the main verb. Rather, the auxiliary gets included by separately adjoining a tree

14 Throughout this section, we underline the embedded clause with all of its arguments, such as here, the raised subject.

Downloaded from http://www.mitpressjournals.org/doi/pdf/10.1162/089120101300346813 by guest on 28 March 2024

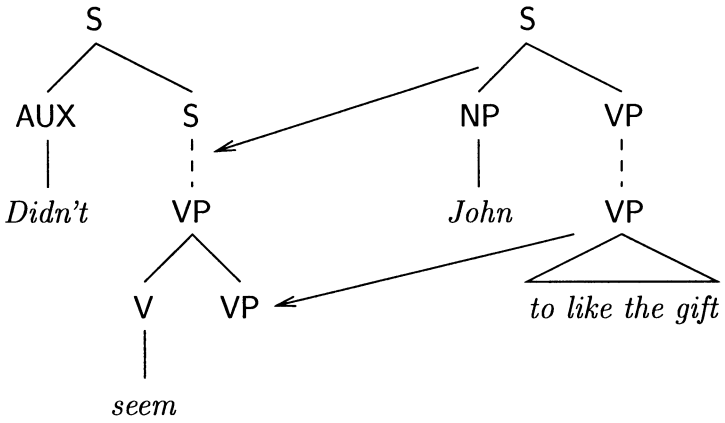


Figure 19
Raising verb with a fronted auxiliary.

projected from the auxiliary verb. The adjunction of the auxiliary is forced through a linguistically motivated system of features. A treatment such as this is needed to avoid using multicomponent adjoining. In our example, the auxiliary, along with the negation marker, is adjoined into the tree projected by the embedded verb *like*, which may be considered undesirable since semantically, it is the matrix verb *seem* that is negated. We take this example to show once more that TAG imposes restrictions on the linguistic analyses that can be expressed in it. Specifically, there are constructions (which do not involve long-distance phenomena) for which one of the most widely developed and comprehensive theories for determining the nature of localization in elementary trees—that of Frank (1992)—cannot be used because of the nature of the TAG operation of adjunction. In contrast, the operations of DSG allow this theory of elementary lexical projections to be used.

In English, the finite verb appears before the subject only in questions (and in some other contexts such as neg-inversion), but in other languages, this word order is routine, leading to similar problems for an LTAG analysis. In V1 languages such as Welsh, the subject appears in second position after the finite verb in the standard declarative sentence. The raised subject behaves in the same manner as the matrix subject, as observed in Harley and Kulick (1998) and illustrated in (6), from Hendrick (1988):

(6) a. Mae Siôn yn gweld Mair
Is John seeing Mary
John is seeing Mary

b. Mae Siôn yn digwydd bod yn gweld Mair
Is John happening be seeing Mary
John happens to be seeing Mary

In German, a V2 language, the finite verb appears in second position in matrix clauses. The first position may be occupied by any constituent (not necessarily the subject). When the subject is not in initial position, it follows the finite verb, both in

Downloaded from http://www.mitpressjournals.org/doi/pdf/10.1162/089120101300346813 by guest on 28 March 2024

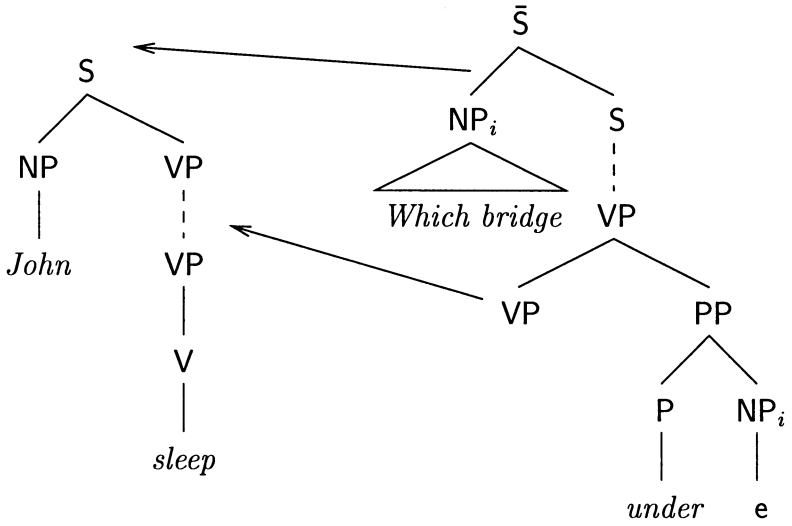


Figure 20
Licit extraction from an adjunct in English.

simplex sentences and in raising constructions:

(7) a. Leider wird es ständig regnen
unfortunately will it$_{NOM}$ continually rain
Unfortunately, it will rain continually

b. Oft schien es uns ständig zu regnen
often seemed it$_{NOM}$ us$_{DAT}$ continually to rain
Often it seemed to us to rain continually

In the German example, a separate adjunction of the tensed verb (as in the XTAG analysis of the English auxiliary) is not a viable analysis at all, since the tensed verb is not an auxiliary but the main (raising) verb of the matrix clause.

We now return to examples that do not include raising, but only *wh*-dependencies.

(8) a. John slept under the bridge

b. Which bridge did John sleep under?

Most LTAG analyses would treat the prepositional phrase in (8a) as an adjunct and use an intransitive frame for the verb. However, the related sentence (8b) cannot be analyzed with TAG operations in the same way, because the projected structures from the verb and the preposition would have to be as shown in Figure 20. The interspersing of components from these projections to obtain the desired tree cannot be obtained using adjoining. Clearly, with the appropriate generalized substitutions in DSG, this tree alone will be derived with these lexical projections.

Related problems arise in languages in which a *wh*-moved element does not invariably appear in sentence-initial position, as it does in English. For example, in

Downloaded from http://www.mitpressjournals.org/doi/pdf/10.1162/089120101300346813 by guest on 28 March 2024

Kashmiri, the *wh*-element ends up in second position in the presence of a topic. This is the case even if the *wh*-element comes from the embedded clause and the topic from the matrix clause. (The data is from Bhatt, [1994].)

(9) a. rameshan kyaa dyutnay tse

Ramesh$_{ERG}$ what$_{NOM}$ gave you$_{DAT}$

What did you give Ramesh?

b. rameshan kyaa$_i$ chu baasaan ki me kor t$_i$

Ramesh$_{ERG}$ what is believe$_{NPERF}$ that I$_{ERG}$ do

What does Ramesh believe that I did?

Another example comes from Rumanian. Rumanian differs from English in that it allows multiple fronted *wh*-elements in the same clause. Leahu (1998) illustrates this point with the examples in (10) (her (8a) and (11a)); (10a) shows multiple *wh*-movement in the same clause, while (10b) shows multiple *wh*-words in one clause that originate from different clauses, resulting again in an interspersed order.

(10) a. Cine$_i$ cui$_j$ t$_i$ promite o masina t$_j$?

who to whom promises a car

Who promises a car to whom?

b. Cine$_i$ pe cine$_j$ a zis t$_i$ ca a vazut t$_j$?

who whom has said that has seen

Who has said he has seen whom?

The examples discussed in this section show a range of syntactic phenomena in English and in other languages that cannot be analyzed using the operations of TAG. We conclude that complex interspersing is a fairly common phenomenon in natural language. As in the case of factoring of recursion, sometimes we find that the definition of adjunction precludes certain linguistically plausible analyses but allows others; in other cases, TAG does not seem to allow any linguistically plausible analysis at all. However, in each case, we can use standard LTAG practices for projecting structures from lexical items and combine the resulting structures using the generalized substitution operation of DSG to obtain the desired analyses, thus bringing out the underlying similarity of related constructions both within languages and cross-linguistically.

4.3 Linguistic Use of Path Constraints

In the examples discussed so far, we have not had the need to use path constraints. The d-edges seen so far express any domination path. Recall that path constraints can be associated with a d-edge to express certain constraints on what nodes, in terms of their labels, cannot appear within a path instantiating a d-edge.

Downloaded from http://www.mitpressjournals.org/doi/pdf/10.1162/089120101300346813 by guest on 28 March 2024

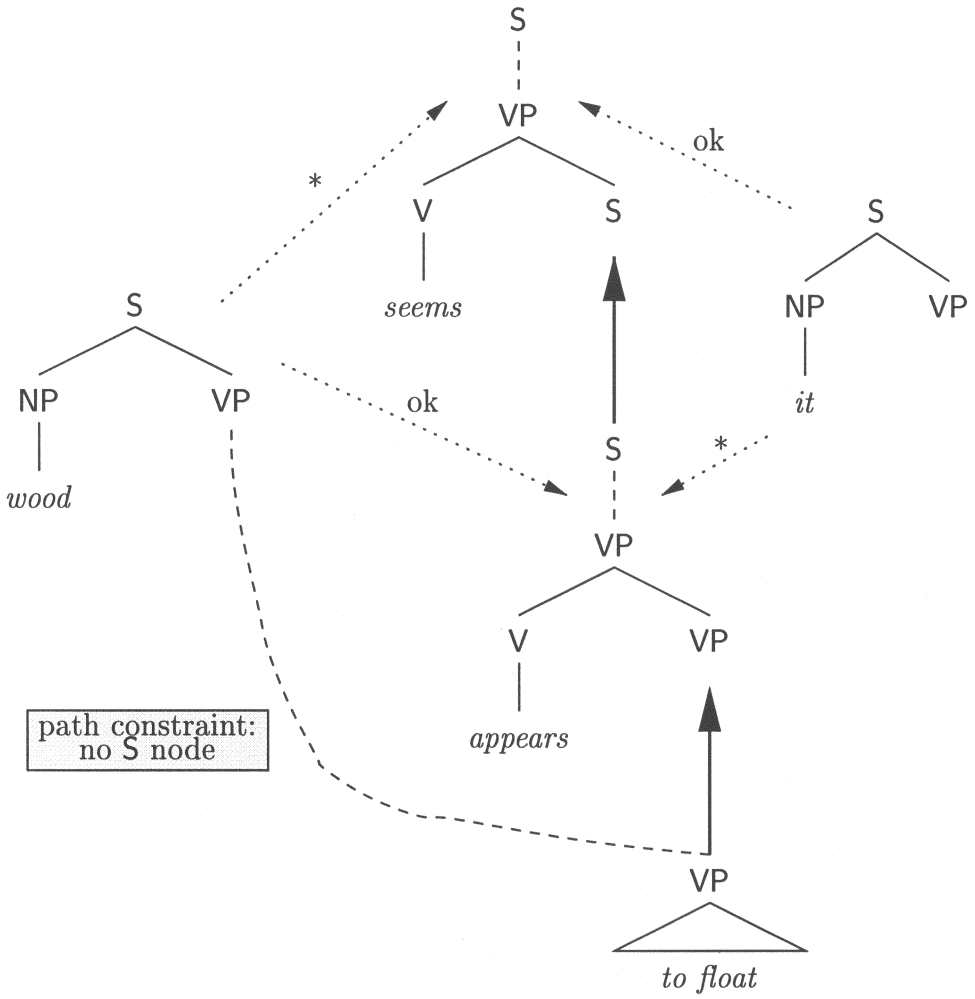


Figure 21
Path constraints are needed to rule out ungrammatical super-raising.

As an example of the use of path constraints, let us consider the well-known case of "super-raising":

(11) a. It seems wood appears to float

b. *Wood seems it appears to float

c. Wood seems to appear to float

In (11a), the subject of *float*, *wood*, has raised to the *appears* clause, while the raising verb *seem* does not trigger raising and has an expletive *it* as its subject. In (11b), *wood* has raised further, and *appear* now has an expletive subject; (11b) is completely ungrammatical. If we make the intermediate raising verb *appear* nonfinite (and hence without a subject), as in (11c), the sentence is again grammatical.

Now consider the DSG analysis for (11a) shown in Figure 21. The d-tree for *seem* has an S substitution node, since *seems* takes a finite complement with a subject. *Appear*,

Downloaded from http://www.mitpressjournals.org/doi/pdf/10.1162/089120101300346813 by guest on 28 March 2024

since it is finite, projects to S, but takes a VP complement since its complement, the *float* clause, is nonfinite and has no overt subject.[15] We furthermore assume that the raising verbs *seem* and *appear* do not select for subjects, but that the expletive subject *it* is freely available for inclusion in their d-trees, since expletive *it* is semantically vacuous and merely fulfills syntactic requirements (such as subject-verb agreement), not semantic ones. We substitute the *float* d-tree into the *appear* d-tree, and the result into the *seem* d-tree, as indicated by the solid arrows in Figure 21. Given the reading off process, this derived d-tree can be seen to express two possibilities, depending on where the *wood* component and the expletive *it* end up. These two possibilities correspond to (11a) and (11b).

To exclude the ungrammatical result, we use the path constraints discussed in Section 2.6. Let us make the uncontroversial assumption that as we project from a verb, we will project to a VP before projecting to an S. But we will interpret this notion of projection as also applying to the d-edges between nodes labeled VP: we annotate the d-edge between the VP nodes in the *float* tree (and in fact in all trees, of course) as having a path constraint that does not allow an S node on this path. This is, after all, what we would expect in claiming that the *float* tree represents a structure lexically projected from the verb *float*.[16] Given this additional grammatical expression, after the substitution at the S node of the *seems* tree, it is no longer possible to read off from the d-tree in Figure 21 a tree whose yield is the ungrammatical (11b). The only possible way of reading off from the derived d-tree yields (11a).

What is striking is that this particular path constraint disallowing S nodes between VP nodes in structures projected from a verb can be used in other cases as well. In fact, this same path constraint on its own, when applied to the English examples considered so far, predicts the correct arrangement of all components among the two d-trees being combined, regardless of whether the nesting constraint of adjoining must be met (extraction out of clausal or VP complements, extraction from NP or PP complements), or not (extraction from the clause of a raising verb, raising verb with fronted auxiliary, or extraction from an adjunct). For example, in Figure 18, after substituting the *to be happy* component at the VP node of the *appears* d-tree, a path constraint on the d-edge between the two VP nodes of the *to be happy* tree makes it impossible for the *to any of us* component to intervene, thus leaving the interspersed tree as the only possible result of the reading off process, even if we relaxed the requirement on label equality for the removal of d-edges during the reading off process.

Note that while the same path constraints apply in all cases, in LTAG, as we have seen, the nesting constraint of adjoining precludes deriving the correct order in some cases, and the use of extensions such as multicomponent adjoining has been suggested. In fact, because there are both situations in which the arrangement of components of the lexically projected structures corresponds to adjoining and situations in which this arrangement is inappropriate, Vijay-Shanker (1992) raises the question of whether the definition of the formalism should limit the arrangement of components of the lexically projected structures, or whether the possible arrangements should be derived from the linguistic theory and from intuitions about the nature of the elementary objects of a grammar. This subsection partially addresses this question and shows

15 The point we are making in this section relies on there being some distinction between the labels of the roots of the *appear* and *float* clauses, a linguistically uncontroversial assumption. Here, we use the categorial distinction between S and VP for convenience only; we could also have assumed a difference in feature content.

16 Bleam (2000) uses informal path constraints in much the same way in order to restrict Spanish clitic climbing in an LTAG analysis.

Downloaded from http://www.mitpressjournals.org/doi/pdf/10.1162/089120101300346813 by guest on 28 March 2024

how the path constraint expressing the nature of projection from a lexical item can be used to derive the arrangements of components corresponding to adjoining in some cases as well as predict when the nesting condition of adjoining is too limiting in the others.

4.4 Underspecification of Linear Precedence

In our proposed tree description language, we provide for underspecified dominance but not for underspecified linear precedence. As a consequence, in the graphical representations of d-trees, we assume that sister nodes are always ordered as shown. This may seem arbitrary at first glance, especially since in many linguistic frameworks and theories it is common to specify linear precedence (LP) separately from syntactic structure (GPSG, HPSG, LFG, ID/LP-TAG [Joshi 1987] and FO-TAG [Becker, Joshi, and Rambow 1991], various dependency-based formalisms, and so on). This separate specification of LP rules allows for underspecified LP rules, which is useful in cases in which word order is not fully fixed.

In principle, an underspecification of LP could easily be added to DSG without profoundly changing its character or formal properties. The reason we have not done so is that in all cases, the same effect can be achieved using underspecified dominance alone, though at the cost of forcing a linguistic analysis that uses binary branching phrase structure trees rather than *n*-ary branching ones. We will illustrate the point using examples from German, which allows for scrambling of the arguments.

Consider the following German examples.[17]

(12) a. daß die Kinder dem Lehrer das Buch geben
that [the children]$_{NOM}$ [the teacher]$_{DAT}$ [the book]$_{ACC}$ give
that the children give the teacher the book

b. daß dem Lehrer die Kinder das Buch geben

c. daß dem Lehrer das Buch die Kinder geben

All orders of the three arguments are possible, resulting in six possible sentences (three of which are shown in (12)). In DSG, we can express this by giving the lexical entry for *geben* shown in Figure 22.[18] The arguments of the verb have no dominance specified among them, so that when using this d-tree (which is of course not yet a tree) in a derivation, we can choose whichever dominance relations we want when we read off a tree at the end of the derivation. As a result, we obtain any ordering of the arguments.

As mentioned previously, while we can derive any ordering, we cannot, in DSG, obtain a flat VP structure. However, our analysis has an advantage when we consider "long scrambling," in which arguments from two lexical verbs intersperse. (In German, only certain matrix verbs allow long scrambling.) If we have the subject-control verb *versuchen* 'to try', the nominative argument is the overt subject of the matrix clause, while the dative and accusative arguments are arguments of the embedded clause. Nonetheless, the same six word orders are possible (we again underline the embedded

17 We give embedded clauses starting with the complementizer in order to avoid the problem of V2. For a discussion of V2 in a framework like DSG, see Rambow (1994a) and Rambow and Santorini (1995).

18 We label all projections from the verb (except the immediate preterminal) VP. We assume that relevant levels of projection are distinguished by the feature content of the nodes. This choice has mainly been made in order to allow us to derive verb-second matrix clause order using the same d-trees, which is also why the verb is in a component of its own.

Downloaded from http://www.mitpressjournals.org/doi/pdf/10.1162/089120101300346813 by guest on 28 March 2024

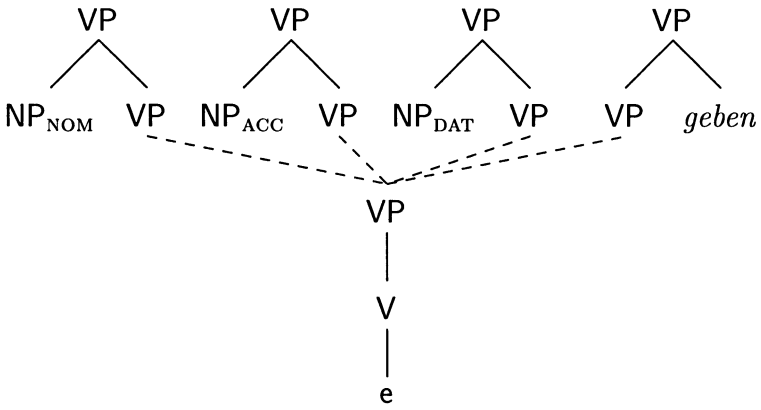


Figure 22
D-tree for German verb *geben* 'to give'.

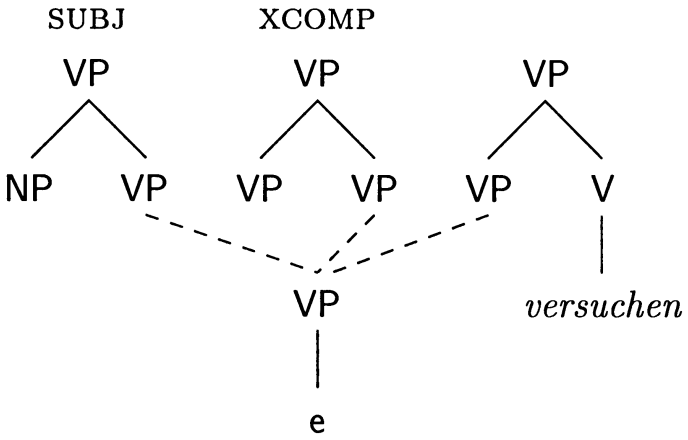


Figure 23
D-tree for German verb *versuchen* 'to try'.

clause material):

(13) a. daß die Kinder dem Lehrer das Buch zu geben versuchen
that [the children]$_{NOM}$ [the teacher]$_{DAT}$ [the book]$_{ACC}$ to give try
that the children try to give the teacher the book

b. daß dem Lehrer die Kinder das Buch zu geben versuchen

c. daß dem Lehrer das Buch die Kinder zu geben versuchen

We can represent the matrix verb as shown in Figure 23, and a derivation as shown in Figure 24. It is clear that we can still obtain all possible word orders, and that this

Downloaded from http://www.mitpressjournals.org/doi/pdf/10.1162/089120101300346813 by guest on 28 March 2024

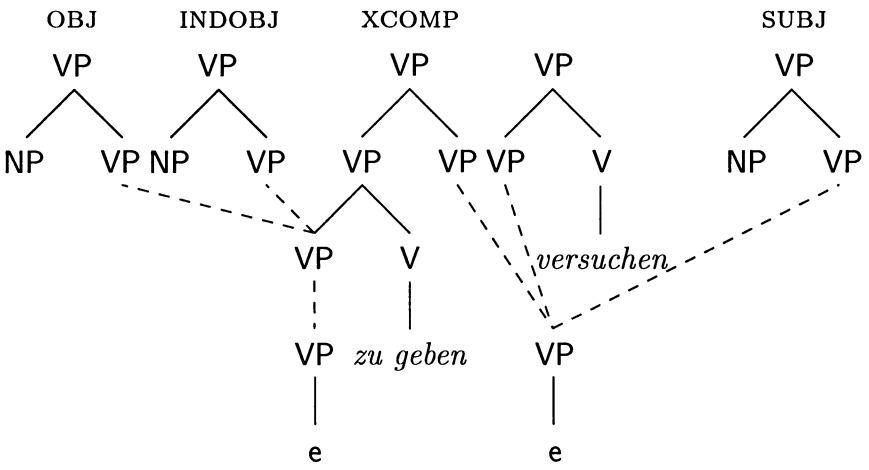


Figure 24
DSG derivation for a complex sentence.

would be impossible using simple LP rules that order sister nodes.[19] (It would also be impossible in LTAG, but see Joshi, Becker, and Rambow [2000] for an alternate discussion of long scrambling in LTAG.)

5. Modeling Syntactic Dependency

In the previous sections, we have presented DSG and have shown how it can be used to provide analyses for a range of linguistic phenomena. In this section, we conclude our introduction of DSG by discussing the relationship between derivations in DSG and **syntactic dependency**. Recently, syntactic dependency has emerged as an important factor for applications in natural language processing.

In lexicalized formalisms such as LTAG, the operations of the formalism (i.e., in the case of LTAG, substitution and adjunction) relate structures associated with two lexical items. It is therefore natural to interpret these operations as establishing a direct syntactic relation between the two lexical items, i.e., a relation of syntactic dependency. There are at least two types of syntactic dependency: a relation of complementation (predicate-argument relation) and a relation of modification (predicate-adjunct relation).[20] Syntactic dependency represents an important linguistic intuition, provides a uniform interface to semantics, and is, as Schabes and Shieber (1994) argue, important in order to support statistical parameters in stochastic frameworks. In fact,

19 This kind of construction has been extensively analyzed in the Germanic syntax literature. Following the descriptive notion of "coherent construction" proposed by Bech (1955), Evers (1975) proposes that in German (and in Dutch, but not in English) a biclausal structure undergoes a special process to produce a monoclausal structure, in which the argument lists of the two verbs are merged and the verbs form a morphological unit. This analysis has been widely adopted (in one form or another) in the formal and computational syntax literature by introducing special mechanisms into the underlying formal system. If the special mechanism produces a single (flat) VP for the new argument list, then LP rules for the simplex case can also apply to the complex case. However, the DSG analysis has the advantage that it does not involve a special mechanism, and the difference between German and English complex clauses is related simply to the difference in word orders allowable in the simplex case (i.e., German but not English allows scrambling). Furthermore, the DSG analysis correctly predicts some "interleaved" word orders to be grammatical. See Rambow (1995) for details.

20 In addition, we may want to identify the relation between a function word and its lexical headword (e.g., between a determiner and a noun) as a third type of relation.

Downloaded from http://www.mitpressjournals.org/doi/pdf/10.1162/089120101300346813 by guest on 28 March 2024

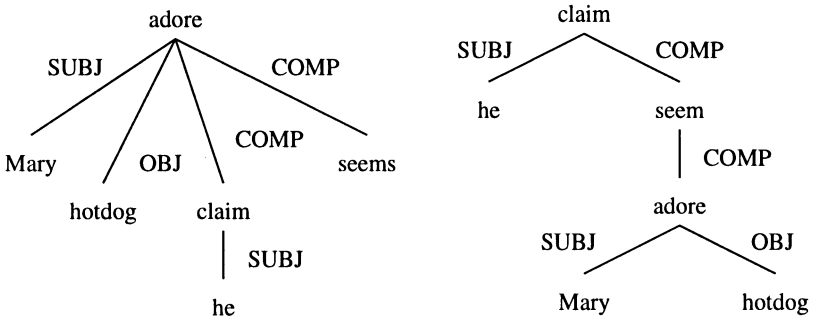


Figure 25
LTAG derivation tree for (14) (left); dependency tree for (14) (right).

recent advances in parsing technology are due to the explicit stochastic modeling of dependency information (Collins 1997).

Purely CFG-based approaches do not represent syntactic dependency, but other frameworks do, e.g. the f-structure (functional structure) of LFG (Kaplan and Bresnan 1982), and dependency grammars (see, for example, Mel'čuk [1988]), for which syntactic dependency is the sole basis for representation. As observed by Rambow and Joshi (1997), for LTAG, we can see the derivation structure as a dependency structure, since in it lexemes are related directly.

However, as we have pointed out in Section 4.1, the LTAG composition operations are not used uniformly: while substitution is used only to add a (nominal) complement, adjunction is used both for modification and (clausal) complementation.[21] Furthermore, there is an inconsistency in the directionality of the substitution operation and those uses of adjunction for clausal complementation: in LTAG, nominal complements are substituted into their governing verb's tree, while the governing verb's tree is adjoined into its own clausal complement. The fact that adjunction and substitution are used in a linguistically heterogeneous manner means that (standard) LTAG derivation trees do *not* provide a direct representation of the dependencies between the words of the sentence, i.e., of the predicate-argument and modification structure. In DSG, this problem is overcome straightforwardly, since DSG uses generalized substitution for *all* complementation (be it nominal or clausal), while still allowing long-distance effects.[22]

There is a second, more serious problem with modeling syntactic dependency in LTAG, as can be seen from the following example:

(14) Hot dogs he claims Mary seems to adore

The problem is that in the standard LTAG derivation, we adjoin both the trees for *claim* and *seem* into the tree for *adore* (Figure 25, left), while in the (commonly assumed) dependency structure, *seem* depends on *claim*, and *adore* depends on *seem* (Figure 25, right). The problem is in fact related to the interleaving problem discussed in Section 4.2, and can easily be solved in DSG by proposing a structure such as that

21 Clausal complementation cannot be handled uniformly by substitution because of the existence of syntactic phenomena such as long-distance *wh*-movement in English.

22 Modification can be handled by some other operation, such as sister adjunction (Rambow, Vijay-Shanker, and Weir 1995), and is thus distinguished from complementation. We do not discuss modification in this paper.

Downloaded from http://www.mitpressjournals.org/doi/pdf/10.1162/089120101300346813 by guest on 28 March 2024

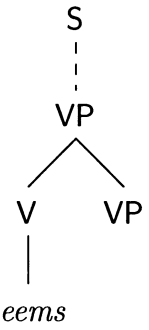


Figure 26
Elementary d-tree for finite *seems*.

in Figure 26 for *seems*, which we have already seen in Figure 21. (This structure can be justified on linguistic grounds independently from the dependency considerations, by assuming that all finite verbs—whether raising or not—project to at least S [= IP]. Raising verbs simply lack a subject of their own, but the S node is justified by the finiteness of the verb, not by the presence or absence of a subject.) Thus, DSG can be used to develop grammars in which the derivation faithfully and straightforwardly reflects syntactic dependency.[23]

6. Related Work

In this section, we mention some related theoretical work and some application-oriented work that is based on DSG.

On the theoretical side, Kallmeyer (1996, 1999) presents an independently conceived formalism called tree description grammar (TDG). TDG is similar to DSG: in both formalisms, descriptions of trees are composed during derivations through conjunction and equation of nodes. Furthermore, like DSG, TDG does not allow the conflation of immediate dominance structure specified in elementary structures. However, TDG allows for more than one node to be equated in a derivation step: nodes are "marked" and all marked nodes are required to be equated with other nodes in a derivation step. (Equating more than one pair of nodes in each derivation step shifts some of the work done in reading off in DSG to the derivation in TDG.) In DSG, we have designed a simple generative system based on tree descriptions involving dominance, using an operation that directly correspond to the linguistic notion of complementation. Additional mechanisms, such as the marking of nodes and their simultaneous involvement in a derivation step, are not available in DSG.

Hepple (1998) relates DSG to a system he has previously proposed in which deductions in implicational linear logic are recast as deductions involving only first-order formulas (Hepple 1996). He shows how this relation can be exploited to give derivations in DSG a functional semantics.

There is an ongoing effort to evaluate the theoretical proposals presented in this paper through the development of a wide-coverage DSG-based parsing system that provides analysis in a broadly HPSG style (Carroll et al. 2000). One aspect of this work involves exploiting the extended domain of locality that DSG shares with TAG in order

23 Candito and Kahane (1998) propose to use derivations in DSG to model semantic (rather than syntactic) dependency.

Downloaded from http://www.mitpressjournals.org/doi/pdf/10.1162/089120101300346813 by guest on 28 March 2024

to maximize localization of syntactic dependencies within elementary tree descriptions, thereby avoiding the need for unification during parsing (Carroll et al. 1999).

Nicolov and Mellish (2000) use DSG as the formalism in a generation application. The principal motivation for using DSG is that DSG is a lexicalized formalism which can provide derivations that correspond to the traditional notion of (deep) syntactic dependency (see Section 5), which is often considered to be the input to the syntactic component of a generation system.

7. Conclusions

We have introduced the grammar formalism of d-tree substitution grammars by showing how it emerges from a tree-description-theoretic analysis of tree adjoining grammars. Derivations in DSG involve the composition of d-trees, special kinds of tree descriptions. Trees are read off from derived d-trees.

We have shown that the DSG formalism can be used to express a variety of linguistic analyses, including styles of analysis that do not appear to be available with the LTAG approach, and analyses for constructions that appear to be beyond the descriptive capacity of LTAG. Furthermore, linguistic analyses of syntactic phenomena are uniform, both language-internally and cross-linguistically. Finally, DSG allows for a consistent modeling of syntactic dependency.

References

Abeillé, Anne. 1991. *Une grammaire lexicalisée d'arbres adjoints pour le français*. Ph.D. thesis, Université Paris 7.

Backofen, Rolf, James Rogers, and K. Vijay-Shanker. 1995. A first-order axiomatization of the theory of finite trees. *Journal of Language, Logic, and Information*, 4(1):5–39.

Bech, Gunnar. 1955. *Studien über das deutsche Verbum infinitum*. Det Kongelige Danske videnskabernes selskab. Historisk-Filosofiske Meddelelser, bd. 35, nr. 2 (1955) and bd. 36, nr. 6 (1957). Munksgaard, Kopenhagen. Second unrevised edition published 1983 by Max Niemeyer Verlag, Tübingen (Linguistische Arbeiten 139).

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)*, pages 21–26.

Becker, Tilman and Owen Rambow. 1995. Parsing non-immediate dominance relations. In *Proceedings of the Fourth International Workshop on Parsing Technologies*, pages 26–33, Prague.

Bhatt, Rakesh. 1994. *Word Order and Case in Kashmiri*. Ph.D. thesis, University of Illinois, Urbana-Champaign.

Bleam, Tonia. 2000. Clitic climbing and the power of tree adjoining grammar. In Anne Abeillé and Owen Rambow, editors, *Tree Adjoining Grammars: Formalisms, Linguistic Analyses and Processing*. CSLI Publications, pages 193–220. Paper initially presented in 1995.

Candito, Marie-Hélène and Sylvain Kahane. 1998. Defining DTG derivations to get semantic graphs. In *Proceedings of the Fourth International Workshop on Tree Adjoining Grammars and Related Frameworks (TAG+4)*, IRCS Report 98–12, pages 25–28. Institute for Research in Cognitive Science, University of Pennsylvania.

Carroll, John, Nicolas Nicolov, Olga Shaumyan, Martine Smets, and David Weir. 1999. Parsing with an extended domain of locality. In *Ninth Conference of the European Chapter of the Association for Computational Linguistics (EACL'99)*, pages 217–224.

Carroll, John, Nicolas Nicolov, Olga Shaumyan, Martine Smets, and David Weir. 2000. Engineering a wide-coverage lexicalized grammar. In *Proceedings of the Fifth International Workshop on Tree Adjoining Grammars and Related Frameworks*, pages 55–60.

Collins, Michael. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of the 35th Annual Meeting*, Madrid, Spain, July. Association for Computational Linguistics.

Evers, Arnold. 1975. *The Transformational Cycle in Dutch and German*. Ph.D. thesis, University of Utrecht. Distributed by the Indiana University Linguistics Club.

Downloaded from http://www.mitpressjournals.org/doi/pdf/10.1162/089120101300346813 by guest on 28 March 2024

Frank, Robert. 1992. *Syntactic Locality and Tree Adjoining Grammar: Grammatical, Acquisition and Processing Perspectives*. Ph.D. thesis, Department of Computer and Information Science, University of Pennsylvania.

Frank, Robert. Forthcoming. *Phrase Structure Composition and Syntactic Dependencies*. MIT Press, Cambridge.

Gazdar, G. 1988. Applicability of indexed grammars to natural languages. In U. Reyle and C. Rohrer, editors, *Natural Language Parsing and Linguistic Theories*. D. Reidel, Dordrecht, pages 69–94.

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)*, IRCS Report 98–12, pages 62–65. Institute for Research in Cognitive Science, University of Pennsylvania.

Hendrick, R. 1988. *Anaphora in Celtic and Universal Grammar*. Kluwer Academic Publishers, Dordrecht.

Hepple, Mark. 1996. A compilation-chart method for linear categorical deduction. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING'96)*, pages 537–542.

Hepple, Mark. 1998. On same similarities between D-Tree Grammars and type-logical grammars. In *Proceedings of the Fourth International Workshop on Tree Adjoining Grammars and Related Frameworks (TAG+4)*, IRCS Report 98–12, pages 66–69. Institute for Research in Cognitive Science, University of Pennsylvania.

Joshi, Aravind K. 1987. Word-order variation in natural language generation. Technical Report, Department of Computer and Information Science, University of Pennsylvania.

Joshi, Aravind K., Tilman Becker, and Owen Rambow. 2000. A new twist on the competence/performance distinction. In Anne Abeillé and Owen Rambow, editors, *Tree Adjoining Grammars: Formalisms, Linguistic Analysis, and Processing*. CSLI Publications, pages 167–182.

Joshi, Aravind K. and Yves Schabes. 1991. Tree-adjoining grammars and lexicalized grammars. In Maurice Nivat and Andreas Podelski, editors, *Definability and Recognizability of Sets of Trees*. Elsevier.

Kallmeyer, Laura. 1996. Tree description grammars. In D. Gibbon, editor, *Natural Language Processing and Speech Technology. Results of the 3rd KONVENS Conference*, pages 332–341, Berlin. Mouton de Gruyter.

Kallmeyer, Laura. 1999. *Tree Description Grammars and Underspecified Representations*. Ph.D. thesis, University of Tübingen. Available as Technical Report No. 99–08 from the Institute for Research in Cognitive Science at the University of Pennsylvania.

Kaplan, Ronald M. and Joan W. Bresnan. 1982. Lexical-functional grammar: A formal system for grammatical representation. In J. W. Bresnan, editor, *The Mental Representation of Grammatical Relations*. MIT Press, Cambridge, MA.

Kasper, Robert, Bernd Kiefer, Klaus Netter, and K. Vijay-Shanker. 1995. Compilation of HPSG and TAG. In *Proceedings of the Annual Meeting*, pages 92–99. Association for Computational Linguistics.

Kroch, Anthony. 1987. Subjacency in a tree adjoining grammar. In Alexis Manaster-Ramer, editor, *Mathematics of Language*. John Benjamins, Amsterdam, pages 143–172.

Kroch, Anthony. 1989. Asymmetries in long distance extraction in a Tree Adjoining Grammar. In Mark Baltin and Anthony Kroch, editors, *Alternative Conceptions of Phrase Structure*. University of Chicago Press, Chicago, pages 66–98.

Leahu, Manuela. 1998. *Wh*-dependencies in Romanian and TAG. In *Proceedings of the Fourth International Workshop on Tree Adjoining Grammars and Related Frameworks (TAG+4)*, pages 92–95, IRCS Report 98–12, Institute for Research in Cognitive Science, University of Pennsylvania.

Marcus, Mitchell, Donald Hindle, and Margaret Fleck. 1983. D-theory: Talking about talking about trees. In *Proceedings of the 21st Annual Meeting*, Cambridge, MA. Association for Computational Linguistics.

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

Nicolov, Nicolas and Christopher Mellish. 2000. Protector: Efficient generation with lexicalized grammars. In Ruslan Mitkov and Nicolas Nicolov, editors, *Recent Advances in Natural Language Processing (RANLP vol. II)*. John Benjamins, Amsterdam and Philadelphia, pages 221–243.

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

Rambow, Owen. 1994a. *Formal and Computational Aspects of Natural Language Syntax*. Ph.D. thesis, Department of Computer and Information Science, University of Pennsylvania, Philadelphia.

Downloaded from http://www.mitpressjournals.org/doi/pdf/10.1162/089120101300346813 by guest on 28 March 2024

Available as Technical Report 94-08 from the Institute for Research in Cognitive Science (IRCS) and also at ftp://ftp.cis.upenn.edu/pub/rambow/thesis.ps.Z.

Rambow, Owen. 1994b. Multiset-valued linear index grammars. In *Proceedings of the 32nd Annual Meeting*, pages 263–270. Association for Computational Linguistics.

Rambow, Owen. 1995. Coherent constructions in German: Lexicon or syntax? In Glyn Morrill and Richard Oehrle, editors, *Formal Grammar: Proceedings of the Conference of the European Summer School in Logic, Language, and Information*, pages 213–226, Barcelona.

Rambow, Owen. 1996. Word order, clause union, and the formal machinery of syntax. In Miriam Butt and Tracy Holloway King, editors, *Proceedings of the First LFG Conference*. On-line version at http://www-csli.stanford.edu/publications/LFG/lfg1.html.

Rambow, Owen and Aravind Joshi. 1997. A formal look at dependency grammars and phrase-structure grammars, with special consideration of word-order phenomena. In Leo Wanner, editor, *Recent Trends in Meaning-Text Theory*. John Benjamins, Amsterdam and Philadelphia.

Rambow, Owen and Beatrice Santorini. 1995. Incremental phrase structure generation and a universal theory of V2. In J. N. Beckman, editor, *Proceedings of NELS 25*, pages 373–387, Amherst, MA. GSLA.

Rambow, Owen and K. Vijay-Shanker. 1998. *Wh*-islands in TAG and related formalisms. In *Proceedings of the Fourth International Workshop on Tree Adjoining Grammars and Related Frameworks (TAG+4)*, pages 147–150, IRCS Report, 98–12. Institute for Research in Cognitive Science, University of Pennsylvania.

Rambow, Owen, K. Vijay-Shanker, and David Weir. 1995. D-Tree Grammars. In *Proceedings of the 33rd Annual Meeting*, pages 151–158. Association for Computational Linguistics.

Rogers, James and K. Vijay-Shanker. 1992. Reasoning with descriptions of trees. In *Proceedings of the 30th Annual Meeting*, pages 72–80. Association for Computational Linguistics.

Schabes, Yves. 1990. *Mathematical and Computational Aspects of Lexicalized Grammars*. Ph.D. thesis, Department of Computer and Information Science, University of Pennsylvania.

Schabes, Yves and Stuart Shieber. 1994. An alternative conception of tree-adjoining derivation. *Computational Linguistics*, 20(1):91–124.

Vijay-Shanker, K. 1987. *A Study of Tree Adjoining Grammars*. Ph.D. thesis, 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 David Weir. 1999. Exploring the underspecified world of Lexicalized Tree Adjoining Grammars. In *Proceedings of the Sixth Meeting on Mathematics of Language*.

Vijay-Shanker, K., David Weir, and Owen Rambow. 1995. Parsing D-Tree Grammars. In *Proceedings of the Fourth International Workshop on Parsing Technologies*, pages 252–259. ACL/SIGPARSE.

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

Downloaded from http://www.mitpressjournals.org/doi/pdf/10.1162/089120101300346813 by guest on 28 March 2024