May 1994

Formal and Computational Aspects of Natural Language Syntax

Owen Rambow
University of Pennsylvania

University of Pennsylvania Institute for Research in Cognitive Science Technical Report No. IRCS-94-08.

Formal and Computational Aspects of Natural Language Syntax

Abstract

This thesis explores issues related to using a restricted mathematical formalism as the formal basis for the representation of syntactic competence and the modeling of performance. The specific contribution of this thesis is to examine a language with considerably freer word-order than English, namely German, and to investigate the formal requirements that this syntactic freedom imposes. Free word order (or free constituent order) languages can be seen as a test case for linguistic theories, since presumably the stricter word order can be subsumed by an apparatus that accounts for freer word order.

The formal systems investigated in this thesis are based on the tree adjoining grammar (TAG) formalism of Joshi et al. (1975). TAG is an appealing formalism for the representation of natural language syntax because its elementary structures are phrase structure trees, which allows the linguist to localize linguistic dependencies such as agreement, subcategorization, and filler-gap relations, and to develop a theory of grammar based on the lexicon.

The main results of the thesis are an argument that simple TAGs are formally inadequate, and the definition of an extension to TAG that is. Every aspect of the definition of this extension to TAG, called V-TAG, is specifically motivated by linguistic facts, not by formal considerations. A formal investigation of V-TAG reveals that (when lexicalized) it has restricted generative capacity, that it is polynomial parsable, and that it forms an abstract family of languages. This means that it has desirable formal properties for representing natural language syntax. Both a formal automaton and a parser for V-TAG are presented.

As a consequence of the new system, a reformulation of the linguistic theory that has been proposed for TAG suggests itself. Instead of including a transformational step in the theory of grammar, all derivations are performed within mathematically defined formalisms, thus limiting the degrees of freedom in the linguistic theory, and making the theory more appealing from a computational point of view. This has several interesting linguistic consequences; for instance, functional categories are expressed by feature content (not node labels), and head movement is replaced by the adjunction of heads. The thesis sketches a fragment of a grammar of German, which covers phenomena such as scrambling, extraposition, topicalization, and the V2 effect.

Finally, the formal automaton for V-TAG is used as a model of human syntactic processing. It is shown that this model makes several interesting predictions related to free word order in German.

Comments

University of Pennsylvania Institute for Research in Cognitive Science Technical Report No. IRCS-94-08.

The Institute For Research In Cognitive Science

Formal and Computational Aspects of Natural Language Syntax (Ph.D. Dissertation)

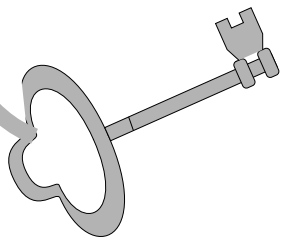
by

Owen Rambow

University of Pennsylvania
3401 Walnut Street, Suite 400C
Philadelphia, PA 19104-6228

May 1994

Site of the NSF Science and Technology Center for Research in Cognitive Science



University of Pennsylvania
Founded by Benjamin Franklin in 1740

IRCS Report 94-08

Formal and Computational Aspects of Natural Language Syntax

Owen Rambow

A Dissertation
in
Computer and Information Science

Presented to the Faculties of the University of Pennsylvania in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy

1994

Aravind K. Joshi and Anthony Kroch
Supervisors of Dissertation

This research was partially funded by ARO grants DAAG29-84-K-0061 and DAAL03-89-C0031PRI, DARPA grants N00014-85-K0018 and N00014-90-J-1863, NSF grants MCS-82-19196 and IRI 90-16592, and Ben Franklin Grant 91S.3078C-1 at the University of Pennsylvania.

Acknowledgements

I have been lucky to have two extremely supportive advisors. Aravind Joshi's deep insight into and interest in the field have provided the intellectual framework in which this thesis work is inscribed. Tony Kroch's undogmatic methodological rigor has formed my understanding of linguistic research. This thesis is profoundly marked by their scholarship and advice, and would not have been possible without it.

I would like to thank the members of my thesis committee for their insightful comments, questions, and criticisms: Jean Gallier, Mitch Marcus, and Mark Steedman. I would especially like to thank the two external members of my committee, Beatrice Santorini (Northwestern University) and Hans Uszkoreit (Universität des Saarlandes), whose detailed comments and questions have gone far beyond the call of duty and whose feedback has been instrumental in shaping various parts of this thesis.

It has been a pleasant surprise to find that academic research is not a solitary enterprise. My interest in computational linguistics arose from an early joint project with Tanya Korelsky and Richard Kittredge; I would like to thank them for exposing me to linguistic theory and to computational linguistics, and for sending me off to Penn for graduate school. This thesis originated with a joint project with Tilman Becker and Aravind Joshi to examine the syntax of German in a tree adjoining grammar framework. Collaboration with Tilman Becker and Michael Niv led to a better understanding of the formal properties of word order variation. Together with Young-Suk Lee, I have explored the linguistic aspects of using tree adjoining grammar for free word order languages. A joint project with Bob Frank and Young-Suk Lee helped me gain a better understanding of the syntactic properties of scrambling. Joint work with Giorgio Satta has dealt with some mathematical properties of formalisms used for linguistic purposes. Finally, I have worked with Tilman Becker on a parsing algorithm for one of the formalisms discussed in this thesis. Throughout the thesis, work that is directly based on joint work is clearly marked as such; however, the entire thesis would have been impossible without the stimulating effect of intellectual interaction. I am extremely grateful to the researchers with whom I have had the fortune of collaborating. These collaborations have been made possible to a large extent by the positive environment for interdisciplinary research at the Institute for Research in Cognitive Science at the University of Pennsylvania.

This thesis work has been greatly helped by discussions with Anne Abeillé, Bob Frank, Beryl Hoffman, Sabine Iatridou, Bernhard Rohrbacher, Yves Schabes, K. Vijay-Shanker, and David Weir. Michael Hegarty and Caroline Heycock have been extremely generous in spending time explaining various aspects of linguistic theory to me. I would also like to thank Lauri Karttunen,

Bob Kasper, and Mike Reape for helpful discussions and crucial comments. Finally, I have profited from the courses taught by Ellen Prince and Bonnie Webber, even though the lessons learned, about the importance of linguistic and extra-linguistic context, go beyond the scope of this thesis.

It is in order to acknowledge the multiplicity of authors, ideas and influences that have shaped this thesis that I will use the first person plural pronoun in the thesis. Needless to say, I assume full responsibility for all errors and infelicities.

I would like to thank John Bacon, Tilman Becker, Michael Hegarty, Beryl Hoffman, Philip Resnik, Giorgio Satta, and K. Vijay-Shanker for reading various drafts of various parts of this thesis and providing crucial early feedback. I am grateful to Tilman Becker, Mischa Czarny, Tanya Korelsky, Young-Suk Lee, Aileen Rambow, Bernhard Rohrbacher, and Beatrice Santorini for native speaker judgments.

John Bacon has been patient and supportive, and has provided perspective; without him, graduate school would have been less fun.

Finally, I would like to thank my parents, Gerhard and Susan Rambow, and my sister, Aileen, for their interest in my work, and their encouragement and support.

Abstract

Formal and Computational Aspects of Natural Language Syntax

Author: Owen Rambow

Supervisors: Aravind K. Joshi and Anthony Kroch

This thesis explores issues related to using a restricted mathematical formalism as the formal basis for the representation of syntactic competence and the modeling of performance. The specific contribution of this thesis is to examine a language with considerably freer word-order than English, namely German, and to investigate the formal requirements that this syntactic freedom imposes. Free word order (or free constituent order) languages can be seen as a test case for linguistic theories, since presumably the stricter word order can be subsumed by an apparatus that accounts for freer word order.

The formal systems investigated in this thesis are based on the tree adjoining grammar (TAG) formalism of Joshi et al. (1975). TAG is an appealing formalism for the representation of natural language syntax because its elementary structures are phrase structure trees, which allows the linguist to localize linguistic dependencies such as agreement, subcategorization, and filler-gap relations, and to develop a theory of grammar based on the lexicon.

The main results of the thesis are an argument that simple TAGs are formally inadequate, and the definition of an extension to TAG that is. Every aspect of the definition of this extension to TAG, called V-TAG, is specifically motivated by linguistic facts, not by formal considerations. A formal investigation of V-TAG reveals that (when lexicalized) it has restricted generative capacity, that it is polynomial parsable, and that it forms an abstract family of languages. This means that it has desirable formal properties for representing natural language syntax. Both a formal automaton and a parser for V-TAG are presented.

As a consequence of the new system, a reformulation of the linguistic theory that has been proposed for TAG suggests itself. Instead of including a transformational step in the theory of grammar, all derivations are performed within mathematically defined formalisms, thus limiting the degrees of freedom in the linguistic theory, and making the theory more appealing from a computational point of view. This has several interesting linguistic consequences; for instance, functional categories are expressed by feature content (not node labels), and head movement is replaced by the adjunction of heads. The thesis sketches a fragment of a grammar of German, which covers phenomena such as scrambling, extraposition, topicalization, and the V2 effect.

Finally, the formal automaton for V-TAG is used as a model of human syntactic processing. It is shown that this model makes several interesting predictions related to free word order in German.

List of Abbreviations

BEPDA ..... Definition 19, page 98
Bottom-Up Embedded Pushdown Automaton

{}-BEPDA ..... Definition 28, page 113
Multiset-Indexed Bottom-Up Embedded Pushdown Automaton

CCG ..... page 122
Combinatory-Categorial Grammar

{}-CCG ..... page 122
Set Combinatory-Categorial Grammar

CETM ..... page 149
Condition on Elementary Tree Minimality

CFG ..... passim
Context-Free Grammar

CKY ..... page 83
Coocke-Kasami-Younger Parser for Context Free Grammar

DG ..... page 127
Dependency Grammar

EPDA ..... page 96
Embedded Pushdown Automaton

ETF ..... Definition 11, page 76 and Definition 18, page 89
Extended Two Form

FO-TAG ..... page 43
Free-Order Tree Adjoining Grammar

GB ..... page 131
Government-and-Binding Theory

GPSG ..... page 198
Generalized Phrase Structure Grammar

SNF ..... Definition 12, page 76
Single Nonterminal Form

TAG ..... Definition 1, page 37
Tree Adjoining Grammar

UMG ..... page 62
Unordered Matrix Grammar

USCG ..... page 62
Unordered Scattered Context Grammar

UVG ..... Definition 4, page 63
Unordered Vector Grammar

UVG-DL ..... Definition 14, page 85
Unordered Vector Grammar with Dominance Links

UVG-DL-Δ ..... Definition 15, page 87
Unordered Vector Grammar with Dominance Links and Integrity

V2 ..... page 10
Verb-Second Phenomenon

VG ..... page 62
Vector Grammar

VMC-TAG ..... Definition 20, page 102
Vector Multi-Component Tree Adjoining Grammar

VMC-TAG-DL ..... Definition 21, page 103
Vector Multi-Component Tree Adjoining Grammar with Dominance Links

V-TAG ..... Definition 21, page 103
Vector Multi-Component Tree Adjoining Grammar with Dominance Links (same as VMC-TAG-DL)

V-TAG-Δ ..... Definition 22, page 104
Vector Multi-Component Tree Adjoining Grammar with Dominance Links and Integrity

Contents

Chapter 1

Introduction: A Formal System for Free Word Order

In this introductory chapter, we outline the motivation for this thesis (Section 1.1), and then provide a quick overview over the main components of the thesis (Section 1.2). The structure of the document is summarized in Section 1.3.

1.1 Mathematical Models for Natural Language Syntax

The use of mathematically defined formalisms for the representation of natural language syntax has three advantages.

- First, the formalism forces the linguist to represent linguistic theory in a precise manner. The conclusions that follow from linguistic stipulations (axioms) can be determined rigorously. Put differently, a mathematical formalism may help enforce methodological rigor, which is presumably required of any scientific endeavor, but is difficult to maintain.
- Second, the mathematical formalism puts at the disposal of the linguist a particular set of means of expression. Different formalisms have different means, and some may be better suited for capturing linguistic intuitions than others, even if they are in some sense equivalent or inter-translatable.
- Third, the mathematical formalism may be restricted in its expressive power. This means that the set of structures (trees, strings) that the formalism can generate is restricted, and the task of the linguist – explaining why certain structures are not legal in natural language – may partly be taken over by the definition of the formalism itself. Furthermore, computational models of syntactic processing can be derived that are restricted in their time complexity. This latter issue is crucial for applications in natural language processing.

Let us consider some examples. We may think of a logical representation of Government and Binding Theory[1] (GB) (Chomsky, 1981), such as that provided by Stabler (1992), as an example of a use of a mathematical formalism for the representation of natural language syntax which has the first of the advantages mentioned above. However, first-order predicate logic (FOPL) does not provide special expressive means, nor is it formally restricted. Head-Driven Phrase-Structure Grammar (HPSG) (Pollard and Sag, 1987; Pollard and Sag, 1994) makes use of a special expressive device, namely feature structures and the associated operation of unification. In addition to imposing methodological rigor, unification also makes available analyses that are not as easily expressed in a theory expressed, for example, in FOPL. However, the formalism used by HPSG is not restricted in its formal power, either. Similarly, Lexical-Functional Grammar (LFG) (Kaplan and Bresnan, 1981) proposes very specific expressive means, but does not limit the formal power of the mathematical system. Finally, there are numerous linguistic theories that are embedded in mathematically restricted formal systems, for example Generalized Phrase Structure Grammar (GPSG) (Gazdar et al., 1985), Combinatory-Categorial Grammar (CCG) (Steedman, 1985; Steedman, 1991), and linguistic theories based on tree adjoining grammar (TAG) (Kroch, 1987; Frank, 1992).

In this thesis, we will adopt the third approach. There is little use in attempting to justify this choice *a priori*. Should it turn out that an interesting linguistic theory can be elaborated in a mathematically restricted formalism, then that result is significant, since it means that linguistic theories that do not use a mathematically restricted formalism are missing an important generalization about the nature of competence syntax. Needless to say, the task of elaborating a complete linguistic theory is one that surpasses the scope of this thesis, and this thesis is therefore to be seen as a part of an ongoing research effort.

There have been a large number of proposals for using formally restricted mathematical systems for the representation of natural language syntax. Most prominent among them is the suggestion that context-free grammar (CFG) provides an adequate but highly restricted framework. The most elaborate linguistic framework expressed in CFG is Generalized Phrase Structure Grammar (GPSG) (Gazdar et al., 1985). There are two types of problems with CFGs:

- There are natural languages which have been shown not to be context-free, under certain assumptions about the nature of competence syntax which are now widely accepted (Shieber, 1985).
- The context-free string rewriting rule, which is the elementary structure of a CFG, does not allow for the elaboration of a syntactic theory that is based on the lexicon (since there must be rules that are not associated with lexical items) and it does not allow the linguist to localize linguistic relations such as subcategorization and agreement, forcing the adoption of additional machinery such as feature-passing conventions.

These two shortcomings have led to an interest in mathematical systems that are somewhat more powerful than CFGs, but not overly so. Linguistic theories expressed in such frameworks include CCG, an extension of Bar-Hillel categorial grammar (which is context-free). A second example

[1] Unless specified otherwise, we will use this term broadly and include in it the variations introduced by the *Barriers* framework of Chomsky (1986a).

is constituted by the linguistic theories that have been developed on the bases of TAG. The TAG formalism (Joshi et al., 1975; Joshi, 1985) is a tree-rewriting system, so that the elementary structures of the grammar are trees. The extended domain of locality of the elementary structures (extended with respect to the string rewriting rules of a CFG) has two important consequences:

- The grammar can be lexicalized, meaning that we can associate an elementary structure of the mathematical formalism with a lexical item and *vice versa*. As Schabes (1990) shows, this is not possible in a CFG. (In fact, the process of lexicalizing a CFG naturally leads to a TAG.) This allows us to develop a syntactic theory which is oriented towards the lexicon.
- Linguistic relations such as selection and agreement can be stated within the elementary structures of the grammar. Furthermore, in a transformational linguistic theory, certain types of transformations, including so-called *wh*-movement and head movement, can be limited to these elementary structures, meaning that those parts of the grammar that concern such transformations can be greatly simplified, since they can only apply to a much smaller domain. This approach has been pursued by Kroch (1987) and Frank (1992).

In this thesis, we adopt the intuition behind the TAG approach, namely that the elementary structures used for the representation of natural language syntax should represent phrase structure, and should be sufficiently large to allow for the localization of linguistic relations such as selection, agreement, and filler-gap dependencies. However, we make no further *a priori* assumptions about the form of these elementary structures.

The specific contribution of this thesis is to examine a language with considerably freer word-order than English, namely German, and to investigate the formal requirements that this syntactic freedom imposes. Free word order (or free constituent order) languages can be seen as a test case for linguistic theories, since presumably the stricter word order can be subsumed by an apparatus that accounts for freer word order.

1.2 Overview of the Thesis

1.2.1 The Formal Requirements of Word Order Variation

The linguistic phenomenon that lies at the base of this thesis is that of free constituent order in languages like German, which is commonly referred to as "scrambling". The following example is from (Haider, 1991).

(1) ...daß [eine hiesige Firma] [meinem Onkel] [die Möbel]
...that [a local company]$_{\text{NOM}}$ [my uncle]$_{\text{DAT}}$ [the furniture]$_{\text{ACC}}$
[vor drei Tagen] [ohne Voranmeldung] zugestellt hat
three days ago without advance warning delivered has
...that a local company delivered the furniture to my uncle three days ago without advance warning

The five constituents in brackets, the three arguments of the verb and two adjuncts, can appear in any order, giving us 5! = 120 possible word orders. What happens when there are embedded

clauses? Scrambling in German is strictly clause-bound in the sense that arguments (or adjuncts) cannot move[2] out of embedded clauses.

(2) * Natürlich hoffe ich dich$_i$, daß er t$_i$ mag
naturally hope I you$_{\text{ACC}}$ that he likes

Intended meaning: Of course I hope that he likes you

However, a large number of German verbs that subcategorize for embedded infinitival clauses allow for so-called *coherent* constructions (the term was first proposed by Bech (1955) and has since been widely adopted in the linguistic literature). In a coherent construction, the verbs in certain respects appear to act as a single verb. In particular, the arguments of both verbs (or, in cases of multiple embeddings, of all verbs) can appear in any order.

(3) Ich glaube, daß [dem Kunden]$_i$ [den Kühlschrank]$_j$ bisher noch niemand t$_i$[[t$_j$
I think that [the client]$_{\text{DAT}}$ the refrigerator$_{\text{ACC}}$ so far as yet no-one$_{\text{NOM}}$
zu reparieren] zu versuchen] versprochen hat
to repair to try promised has

I think that so far, no-one yet has promised the client to repair the refrigerator

Such permutations do not preclude a different type of word order variation, called *topicalization* in the literature, in which a single constituent appears in sentence-initial position just before the matrix finite verb. In (4), the embedded verb, *geben* 'to give', has two overt nominal arguments, one of which has topicalized into sentence-initial position, and the other of which has scrambled beyond the matrix subject.

(4) [Dieses Buch]$_i$ hat [den Kindern]$_j$ bisher noch niemand [PRO t$_j$ t$_i$ zu geben]
[this book]$_{\text{ACC}}$ has [the children]$_{\text{DAT}}$ so far yet [no-one]$_{\text{NOM}}$ to give
versucht
tried

So far, no-one has tried to give this book to the children

The question that arises and that this thesis addresses is: does this word order variation raise any problems for a formal treatment of natural language syntax? If the universal component of syntactic competence allows for such freedom in word order, what sort of formalisms are required for expressing it? In order to address these questions, we observe that the localization of linguistic dependencies, in particular subcategorization, within an elementary structure places a restriction on derivations (as opposed to derived structures): arguments must be directly associated (by a rewriting operation such as adjunction or substitution) with their predicate during a derivation. Using this restriction on derivations, it can easily be shown that sentences such as (4) cannot be derived by a simple TAG. Furthermore, this restriction eliminates a class of systems form consideration which are known as "local".

[2]The term "move" is used in a purely descriptive sense, no particular type of linguistic analysis is implied.

Evers (1975) proposes an influential analysis in the framework of transformational grammar of embedded clauses in Dutch and German that captures the intuition behind the notion of "coherence" by proposing that verbs that form coherent constructions in fact undergo a process by which they form a single, morphologically complex verb with an extended subcategorization frame ("clause union"). Hence, in German coherent constructions in fact form a single clause. This analysis does not affect the arguments that we present here about the formal requirements of German scrambling. Clearly, we would want to model the formation of the complex verb in the formal system, since otherwise we would have to posit an infinite lexicon, which is undesirable. But once we model this process in the formal system (even if we define it to be part of a "pre-syntactic" stage), then the arguments about predicate-argument structure that we have given above apply to the clause-union analysis as well.

1.2.2 Formal Systems

If TAG is formally insufficient to handle the word order facts that we need to capture, then we will need to modify TAG, and we will want to do so in a manner that preserves those features of TAG that make it attractive as a formalism for expressing natural language syntax in the first place. There are two options:

- We can relax the relation of linear precedence between nodes in an elementary tree.
- We can relax the relation of immediate dominance between nodes in an elementary tree.

This thesis argues that the relaxation of LP rules does not provide sufficient constraints to state restriction on word order in case in which (leftward) scrambling is combined with (rightward) extraposition of embedded clauses. However, the proper constraints can be stated simply (using the notion of c-command) if word order variation is accompanied by a change in immediate dominance structure. We therefore opt for the second way of extending TAG.

We define several formalisms, of which we briefly review three here.

- A **multiset-valued linear index grammar** ({}-LIG), based on index grammar of Aho (1968) and Gazdar (1988), is a CFG in which the nonterminal symbols are augmented with multisets of indices. The context-free string rewrite rules of the grammar specify whether index symbols should be removed or added to these multisets. Upon rewriting a nonterminal symbol, the indices in its multiset are distributed freely among the nonterminals that are newly introduced. However, the indices are never copied, making the system linear.
- In an **unordered vector grammar with dominance links** (UVG-DL), context-free rewrite rules are grouped into sets (or "vectors"). These rules can be connected by so-called dominance links. There is no restriction on the use of rewrite rules during a derivation, except that the same number of rules from each set must be used during a derivation, and that a dominance link must correspond to a dominance relation (not necessarily immediate) in the derivation tree. A sample rule set is shown in Figure 1.1 (the dotted lines represent dominance links).

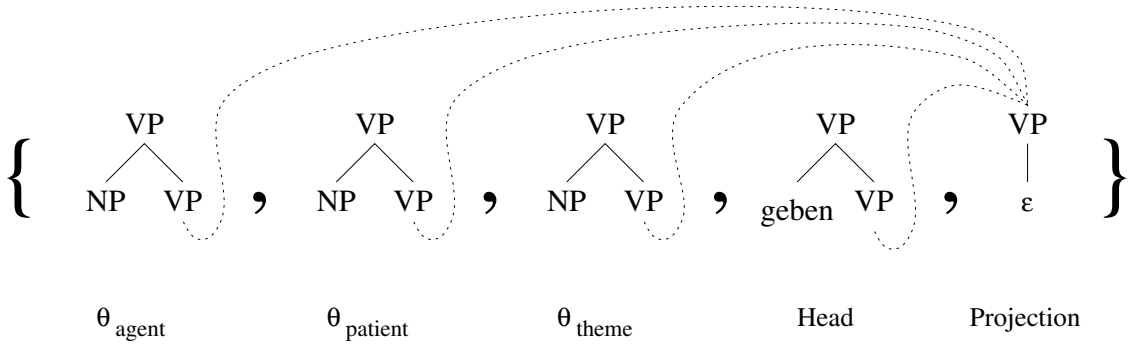


Figure 1.1: Sample UVG-DL rule set

- **Vector TAG** (V-TAG) is the tree-rewriting version of UVG-DL. The elementary structures are sets of trees, which are linked by dominance links. Derivations proceed as in UVG-DL, except that dominance links must correspond to dominance relations in the derived tree. A sample tree set is shown in Figure 1.2 (again, the dotted lines represent dominance links).

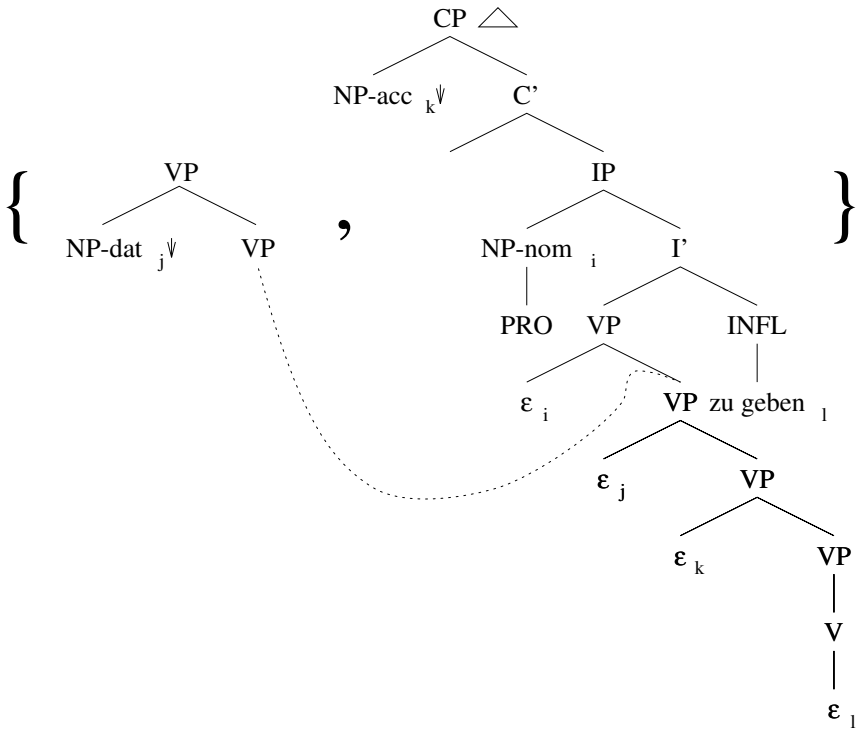


Figure 1.2: Sample tree set in a V-TAG

{}-LIG is used mainly to prove formal results, but it is also useful for the purpose of relating these formalisms to other, previously defined formalisms. We do not actually use {}-LIG in this thesis for linguistic work. The following formal results are proved:

- {}-LIG and UVG-DL are weakly equivalent. The multiset of indices can be used to implement the dominance links and *vice versa*.
- The sets of languages generated by lexicalized {}-LIG (and hence by lexicalized UVG-DL) and by lexicalized V-TAG are contained in the set of context-sensitive languages.
- Lexicalized {}-LIG (and hence lexicalized UVG-DL) and lexicalized V-TAG are polynomially parsable.
- {}-LIG (and hence UVG-DL) form an abstract family of languages.

We also define a formal automaton for {}-LIG and UVG-DL, called the {}-PDA, and a formal automaton for V-TAG, called the {}-BEPDA. The {}-PDA is derived from the pushdown automaton (the automaton of CFG) by adding sets of indices to stack symbols, while the {}-BEPDA is derived from the bottom-up embedded pushdown automaton (an automaton for TAG), also by adding sets of indices to the stack symbols.

1.2.3 Linguistic Interpretation

We use UVG-DL and V-TAG to develop a linguistic theory. These formalisms can be shown to be formally adequate for the relevant linguistic data. The choice of a mathematical formalism for the representation of syntactic competence does not determine the linguistic theory that is implemented in it. V-TAG would allow for a "conservative" extension of the approach of Kroch (1987) and Frank (1992): for example, local transformations could transform a tree (representing some form of deep or D-structure) into a set of trees such as that shown in Figure 1.2. However, we do not choose that approach, and instead this thesis explores an approach in which the transformational step is eliminated altogether. The trees or tree sets, elementary structures in a TAG- or V-TAG-based theory of grammar, are themselves derived formally and non-transformationally from even more elementary structures, which can be thought of as sets of very small trees (i.e., rules), which are in fact rule sets in a UVG-DL. This approach is suggested by the fact that sets of trees are required independently in order to derive the whole range of word order variations. An approach in which we start with a single tree (D-structure), then derive a set of trees, and then derive a single tree again seems unnecessarily complex. Furthermore, it is desirable to eliminate non-formalized transformational steps for two reasons:

- From a practical perspective, a greater part of the linguistic theory can be efficiently implemented.
- From a theoretical perspective, the elimination of transformational steps from the theory removes a possible locus of application of principles and parameters, and thus restricts the freedom in formulating such principles and parameters, while increasing the predictive power of the theory.

The proposal that this thesis pursues is that the lexicon of a natural language can be represented as a set of context-free string-rewriting rules, as shown in Figure 1.1, i.e., as a formal grammar in the UVG-DL formalism. Sentences are then derived in two steps.

- First, during the *lexical derivation*, the rules are used to derive a (partial) structure that roughly corresponds to a lexical item and its extended projection in the sense of Grimshaw (1993). This step includes operations that, in transformational analyses, correspond to *wh*-movement, head movement, and grammatical function-changing operations such as passivization.
- In a second step, called the *syntactic derivation*, the structures derived during the lexical derivation are combined to form sentences.

The thesis sketches a German grammar in this framework that handles the following phenomena:

- Local and non-local scrambling.
- Extraposition and the "Third Construction".
- Local and long-distance *wh*-movement, and the V2 effect.

1.2.4 A Processing Model

In languages in which there is considerable word-order freedom, one can observe a great variation in the acceptability of different word orders. Generally, this variability is taken to be the result of at least two factors: pragmatic constraints and processing constraints. This thesis does not address the issue of pragmatic (or semantic) constraints on word order and on the acceptability of specific word orders. It does present a processing model that makes predictions about the degradation of acceptability based on processing constraints.

Building on previous work (Joshi, 1990), we use {}-BEPDA, the formal automaton of V-TAG, as a model for human syntactic processing. This means that the competence syntax and performance model are closely linked, in the sense that the competence grammar determines the performance model. This approach is shown to make a wide range of predictions, including:

- Long-distance scrambling is degraded over local scrambling.
- Extraposition greatly improves acceptability.
- Long-distance scrambling is degraded with respect to equidistant long-distance topicalization.

However, the model does not make any predictions about clause-internal word-order variation, and appears to exaggerate the difficulty of long-distance scrambling. We discuss some possible ways of improving the model by interpreting the intuition behind the clause union analysis of Evers (1975) as a processing effect.

1.3 Structure of the Thesis

The thesis is structured as follows. The investigation is "data-driven": we present a set of data in Chapter 2 which we would like to find an account for. TAG and various extensions to TAG are discussed as possible formal frameworks for this data in Chapter 3. None of the existing definitions are found quite adequate. In Chapter 4, we define a new extension to TAG, called Vector TAG (V-TAG). Its formal characteristics are explored, and found to be benign. An associated formal automaton is also defined. We develop a linguistic theory based on V-TAG in Chapter 5 and give a fragment of a grammar of German that can account for some of the phenomena introduced in Chapter 2. Finally, in Chapter 6 we use the formal automaton of V-TAG as a model of syntactic processing to account for the wide range of acceptability judgments that we found in Chapter 2. A summary of the main points made in the thesis, and a very brief outlook, can be found in Chapter 7.

Comparison to other work is distributed over several points in the thesis. Reape's "word order domains" are discussed in Section 3.3.6. The formalisms developed in this thesis are compared to categorial systems, to Vijay-Shanker's quasi-trees, to unification-based formalisms such as HPSG, and to dependency grammars in Section 4.7. The linguistic framework proposed in this thesis is compared to previous TAG-based work by Frank and Hegarty, to Karttunen's "radical lexicalism", and to Uszkoreit's GPSG-based work in Section 5.6.

Chapter 2

The Data

This chapter presents the data that subsequent chapters address. We start out by presenting the basic structure of the German sentence (Section 2.1). This will define two types of movement, called *scrambling* and *topicalization*. We present the long-distance version of the former in Section 2.2, and of the latter in Section 2.3. There are important differences between the two types of movement; we discuss these in Section 2.4. Finally, in Section 2.5 we summarize the data that we want our model to account for, and that constitute the criteria for success of this thesis. On two occasions in this chapter, we will present initial linguistic analyses. In Section 2.2.3, we discuss the so-called "Third Contruction" and relate it to the other word order variations we have discussed. In Section 2.3.3, we examine extraction facts from certain types of embedded clauses in order to determine whether one can truly speak of extraction. In both cases, the goal is to gain a better preliminary understanding of the phenomenon in question; it is not meant to be definitive linguistic analysis, and the linguistic analyses do not refer to any particular theoretical framework.

2.1 The Structure of the German Sentence

In German clauses, the verb can appear in one of three positions:

- In declarative clauses without an overt complementizer, the finite verb (main verb or auxiliary) appears in second position:

  (1) Der Lehrer gibt den Kindern dieses Buch
  the teacher$_{\mathrm{NOM}}$ gives the children$_{\mathrm{DAT}}$ this book$_{\mathrm{ACC}}$
  The teacher gives this book to the children

- In interrogative and imperative root clauses, the finite verb appears in initial position.

  (2) Gibt der Lehrer den Kindern dieses Buch?
  gives the teacher$_{\mathrm{NOM}}$ the children$_{\mathrm{DAT}}$ this book$_{\mathrm{ACC}}$
  Does the teacher give this book to the children?

- In clauses with overt complementizers, the verb appears in clause-final position.

  (3) Ich glaube, daß der Lehrer den Kindern dieses Buch gibt
  I think that the teacher$_{\text{NOM}}$ the children$_{\text{DAT}}$ this book$_{\text{ACC}}$ gives
  I think that the teacher gives this book to the children

  Furthermore, if the tensed verb in a finite root clause is an auxiliary, then the main verb appears in clause-final position.

  (4) Der Lehrer hat den Kindern dieses Buch gegeben
  the teacher$_{\text{NOM}}$ has the children$_{\text{DAT}}$ this book$_{\text{ACC}}$ given
  The teacher has given this book to the children

We will assimilate the first two cases, by assuming that in the verb-initial clauses, there is actually some sort of empty operator in initial position. We can then summarize these facts by saying that German is a V2 verb-final language. The "V2" means that the finite verb is in second position unless there is an overt complementizer, while "verb-final" means that in all other cases the verb is in clause-final position. (We do not wish to imply any particular (transformational) analysis by using this terminology.) This divides the clause into two parts: the position in front of the finite verb, the *Vorfeld* (VF), and the positions between the finite and non-finite verbs, the *Mittelfeld* (MF)[1]. This is summed up as follows:

(5) VF V-finite MF (Verb-non-finite)
Comp MF (Verb-non-finite) V-finite

The VF must contain exactly one constituent, which can be any element (an argument or an adjunct, or a projection of the non-finite verb, or an expletive *es* 'it'). In the case of clauses with simple tensed verbs, the final position in the clause for the non-finite verb remains empty. Note that it is also possible (under certain circumstances) to omit the complementizer in embedded clauses, in which case the V-2 effect is obligatory again.

(6) a. Ich glaube, daß der Lehrer den Kindern dieses Buch gegeben hat
I think that the teacher$_{\text{NOM}}$ the children$_{\text{DAT}}$ this book$_{\text{ACC}}$ given has
I think that the teacher has given this book to the children

b. Ich glaube, der Lehrer hat den Kindern dieses Buch gegeben

c. * Ich glaube, der Lehrer den Kindern dieses Buch gegeben hat

This basic division of the German sentence, into VF and MF, gives us two possibilities for word-order variation: we can choose different elements for the VF, and we can vary the order of constituents within the MF. Let us discuss these two options in turn.

[1] The area behind the verbs in clause-final position is called the *Nachfeld* ("afterfield"). In Modern Standard German, the *Nachfeld* can only contain an extraposed subordinate clause (see Section 2.1.3), but never nominal arguments or adjuncts. This terminology, which is now commonly used for descriptive purposes, was first introduced by Drach (1937).

2.1.1 Topicalization

In transformational analyses, the VF element moves there from the MF; this movement is called *topicalization*.[2] The MF contains the remaining arguments and adjuncts. Some more examples follow.

(7) a. Der Lehrer hat den Kindern dieses Buch gegeben
the teacher$_{\text{NOM}}$ has the children$_{\text{DAT}}$ this book$_{\text{ACC}}$ given
The teacher has given this book to the children

b. Dieses Buch hat der Lehrer den Kindern gegeben
this book$_{\text{ACC}}$ has the teacher$_{\text{NOM}}$ the children$_{\text{DAT}}$ given
The teacher has given this book to the children

c. Gestern hat der Lehrer den Kindern dieses Buch gegeben
yesterday has the teacher$_{\text{NOM}}$ the children$_{\text{DAT}}$ this book$_{\text{ACC}}$ given
Yesterday, the teacher gave this book to the children

d. *Gestern der Lehrer hat den Kindern dieses Buch gegeben

e. * Hat der Lehrer den Kindern dieses Buch gegeben

In (7a), the default word order, the subject is in the VF. In (7b), the direct object has topicalized into the VF, so that the subject remains in the MF. In (7c), an adjunct occupies the VF, so that all three arguments of *geben* 'to give' are in the MF. (7d) and (7e) are ruled out because the VF contains two and no elements, respectively.

2.1.2 Scrambling

The movement of arguments and adjuncts within the MF in German is called *scrambling*. The following example is from (Haider, 1991).

(8) ...daß [eine hiesige Firma] [meinem Onkel] [die Möbel]
...that [a local company]$_{\text{NOM}}$ [my uncle]$_{\text{DAT}}$ [the furniture]$_{\text{ACC}}$
[vor drei Tagen] [ohne Voranmeldung] zugestellt hat
three days ago without advance warning delivered has
...that a local company delivered the furniture to my uncle three days ago without advance warning

As Haider points out, "any permutation of these five elements (5! = 120) is grammatically well-ordered". We see that, contrary to the case of topicalization, scrambling of more than one element is possible.

[2]The term "topicalization" is due to Ross (1967) and has been widely used in the literature. Our use of the term in no way implies a particular pragmatic analysis of this construction (e.g., in terms of topics).

2.1.3 Extraposition

While nominal arguments appear obligatorily within the MF, clausal arguments can optionally appear behind the finite verb. This variation is called *extraposition*.

(9) a. Max hat dem Schneider [PRO die Brücke zu reparieren] versprochen
Max has [the tailor]$_{\text{DAT}}$ the bridge to repair promised

Max has promised [the tailor]$_{\text{DAT}}$ to repair the bridge

b. Max hat dem Schneider versprochen, [PRO die Brücke zu reparieren]

c. Weil Max dem Schneider versprochen hat, [PRO die Brücke zu reparieren]
because Max [the tailor]$_{\text{DAT}}$ promised has the bridge to repair

Because Max has promised [the tailor]$_{\text{DAT}}$ to repair the bridge

If the embedded clause in a finite clause with an overt complementizer (for instance, a *daß* clause), extraposition is in fact preferred:

(10) a. ?? Max hat dem Schneider [daß er die Brücke reparieren wird] versprochen
Max has [the tailor]$_{\text{DAT}}$ that he the bridge repair will promised

Max has promised the tailor that he will repair the bridge

b. Max hat dem Schneider versprochen, [daß er die Brücke reparieren wird]

c. Weil Max dem Schneider versprochen hat, [daß er die Brücke reparieren wird]
because Max [the tailor]$_{\text{DAT}}$ promised has that he the bridge repair will

Because Max has promised the tailor that he will repair the bridge

2.2 Long-Distance Scrambling and the "Third Construction"

2.2.1 Long-Distance Scrambling

If there are embedded clauses, certain matrix verbs allow scrambling of elements out of the embedded clauses ("long-distance" scrambling):

(11) a. ...daß niemand [PRO den Kühlschrank zu reparieren] versprochen hat
...that no-one$_{\text{NOM}}$ the refrigerator$_{\text{ACC}}$ to repair promised has
...that no-one has promised to repair the refrigerator

b. ...daß [den Kühlschrank]$_i$ niemand [t$_i$ zu reparieren] versprochen hat
...that the refrigerator$_{\text{ACC}}$ no-one$_{\text{NOM}}$ to repair promised has

However, in German, scrambling can never proceed out of tensed clauses, or, put differently, a constituent may never be scrambled past a complementizer or a finite verb in V2 position:

(12) a. Peter hat versprochen, daß er den Kühlschrank reparieren wird
Peter has promised that he the refrigerator$_{\text{ACC}}$ repair will

Peter has promised that he will repair the refrigerator

b. * Peter hat den Kühlschrank versprochen, daß er reparieren wird
Peter has the refrigerator$_{\text{ACC}}$ promised that he repair will

Intended reading: Peter has promised that he will repair the refrigerator

c. Peter hat gesagt, er wird den Kühlschrank reparieren
Peter has said he will the refrigerator$_{\text{ACC}}$ repair

Peter has said he will repair the refrigerator

d. * Peter hat den Kühlschrank gesagt, er wird reparieren
Peter has the refrigerator$_{\text{ACC}}$ said he will repair

Intended reading: Peter has said he will repair the refrigerator

It has been suggested that embedded (infinitival) clauses undergo "clause union" (Evers, 1975); this approach has been widely accepted in the linguistic literature (but see Section 5.4.1, page 154). If clause union takes place, then in fact there is no long-distance scrambling in German because no clause boundary is crossed. Therefore, one often sees the statement that German has no long-distance scrambling. However, throughout this thesis we will use the term "long-distance scrambling" in a more descriptive way. We will take the term to mean that an argument of a verb appears (within the *Mittelfeld*) to the left of an argument of a higher, less deeply embedded verb (where "higher" can be , theory-independently, interpreted with respect to the semantics of the embeddings). This definition allows us to characterize word order variation without having to adopt a particular theoretical position. We would like to emphasize that none of the empirical claims that we make about scrambling in German in this chapter contradict those made in the linguistic literature we are aware of.

Scrambling within the *Mittelfeld* is quite free. There is no bound on the number of clause boundaries over which an element can scramble:

(13) a. ...daß [den Kühlschrank]$_i$ niemand [[t$_i$ zu reparieren] zu versuchen]
...that the refrigerator$_{\text{ACC}}$ no-one$_{\text{NOM}}$ to repair to try
versprochen hat
promised has
...that no-one has promised to repair the refrigerator

b. ...daß [den Kühlschrank]$_i$ niemand [[[t$_i$ zu reparieren] zu versuchen]
...that the refrigerator$_{\text{ACC}}$ no-one$_{\text{NOM}}$ to repair to try
zu versprechen] bereit ist
to promise ready is
...that no-one is ready to promise to repair the refrigerator

Furthermore, more than one constituent can scramble in the same sentence ("iterability of scrambling"). We have already seen that this is the case for local scrambling (see (8)). However, iterability

also holds in the case of long-distance scrambling: an element scrambled (long-distance or not) from one clause does not preclude an element from another clause from being scrambled.

(14) ...daß [dem Kunden]$_i$ [den Kühlschrank]$_j$ bisher noch niemand t$_i$
...that [the client]$_{\mathrm{DAT}}$ [the refrigerator]$_{\mathrm{ACC}}$ so far as yet no-one$_{\mathrm{NOM}}$
[[t$_j$ zu reparieren] zu versuchen] versprochen hat
to repair to try promised has
...that so far, no-one yet has promised the client to repair the refrigerator

Observe that scrambling does not obey a "path containment" condition (Pesetsky, 1982), which would require that dependencies between moved element and trace be nested, but not crossed.

We conclude that scrambling in German (and other languages) is "doubly unbounded" in the following sense:[3]

1. There is no bound on the distance over which each element can scramble (unboundedness of scrambling).

2. There is no bound on the number of elements that can scramble in one sentence (iterability of scrambling).

This generalization should not be taken to mean that all sentences in which "doubly unbounded" scrambling has occurred will be judged equally acceptable. Clearly, scrambling is constrained by pragmatic and processing factors, and perhaps also by semantic factors. Analyses of the pragmatic and semantic issues involved in scrambling (see, e.g., (Lenerz, 1977; Höhle, 1982; Moltmann, 1990; Diesing, 1992)) are still somewhat sketchy; however, while they provide constraints on word order, they do not provide evidence that the generalization of "double unboundedness" must be abandoned: the contextual and semantic restrictions on word order do not translate into general rules that would categorically rule out certain formally definable orders (such as, say, word orders derived by multiple long-distance scrambling), irrespective of the particular choice of lexemes and context.[4] Similarly, while processing load appears to increase with an increasing number of scrambled elements, the increase in processing load is gradual, and again we do not have any reason to define a particular "cut-off point" beyond which all orders are ungrammatical. (This argument is similar to the argument in favor of allowing unlimited recursion in the competence grammar.) In conclusion, while the issues of pragmatic and semantic constraints and of processing constraints require further study, they should not be brought to bear on the choice of a formalism for the representation of syntactic competence.

Further support for our empirical claim of "double unboundedness" comes from cross-linguistic evidence. Karttunen (1989, p.58) explicitly claims that in Finnish, scrambling may occur over unbounded distances, and more than constituent may scramble, a claim analogous to our claim of double unboundedness for German.

[3]The "double unboundedness" of German scrambling was first observed in (Becker and Rambow, 1990).

[4]The rules worked out by Uszkoreit (1987) are more restrictive: it is impossible to derive the order DO-IO-SUBJ if all three arguments are full NPs. However, according to (8), such an order is possible. We return to a discussion of Uszkoreit (1987) in Section 5.6.4, page 198.

(15) En minä tennistä$_i$ ole aikonut näissä$_j$ ruveta [pelaamaan t$_i$ t$_j$]
not I tennis have intend these-in start play

I did not intend to start to play tennis in these

(15) is "acceptable to a great majority of speakers", and Karttunen argues that because of the variation in judgments, a grammar should not exclude any of the 42 sentences that can be obtained by scrambling the two most embedded arguments. Similarly, Lee (1993, p.6f) presents data that shows that in Korean, scrambling is both unbounded and iterable. Comparable data for Japanese is provided by Saito (1992) (p.69 for unboundedness of long-distance scrambling, and p.82 for iterability). In Russian, unbounded scrambling is also possible, as shown by Comrie (1973) (cited by Müller and Sternefeld (1993, p.467)), and scrambling is iterable (Müller and Sternefeld, 1993, fn. 5). This latter Russian example, reproduced here as (16), is particularly interesting because it shows that the German restriction against scrambling beyond lexically overt complementizers (or finite verbs) is language-specific, and not a cross-linguistic property of scrambling:

(16) ... čto ty$_i$ menja$_j$ vižu [čto t$_i$ ljubiš t$_j$]
... that you$_{\text{NOM}}$ me$_{\text{ACC}}$ I-see that love

...that I see that you love me

Observe also that under the "clause union" hypothesis of Evers (1975) (which, in some form, is widely held), there appears to be no way to block doubly unbounded scrambling, since under that hypothesis, scrambling is a purely local phenomenon. Whichever mechanisms license all permutations in the case of local scrambling (see example (8)) will also license all permutations in the case of long-distance scrambling (in the sense of the term that we have adopted here). We take the linguistic intuition underlying the "clause union" analysis as further support for our empirical claim.

2.2.2 The "Third Construction"

While transformational accounts usually assume that the base-generated default word order of sentences with embedded clauses in German reflects the verb-final character of the language (as in (11)), in actual speech and writing extraposition of the embedded clause is far more common:

(17) ...daß niemand versprochen hat, das Fahrrad zu reparieren
...that no-one promised has the bike$_{\text{ACC}}$ to repair
...that no-one promised to repair the bike

In addition, a so-called "third construction" is possible (den Besten and Rutten, 1989), in which only the embedded verb, but not its nominal argument has been extraposed:

(18) ...daß niemand das Fahrrad versprochen hat zu reparieren
...that no-one$_{\text{NOM}}$ the bike$_{\text{ACC}}$ promised has to repair
...that no-one promised to repair the bike

A similar construction can also be observed if there is an intermediate clause. From the base-generated order in (19), only the verb sequence *zu reparieren zu versuchen* has been extraposed in (20), but not the NP *das Fahrrad*.

(19) ...daß niemand das Fahrrad zu reparieren zu versuchen versprochen hat
...that no-one$_{\text{NOM}}$ the bike$_{\text{ACC}}$ to repair to try promised has
...that no-one promised to try to repair the bike

(20) ...daß niemand das Fahrrad versprochen hat zu reparieren zu versuchen
...that no-one$_{\text{NOM}}$ the bike$_{\text{ACC}}$ promised has to repair to try
...that no-one promised to try to repair the bike

There are five sentence elements that are in principle permutable.[5] The 5! = 120 permutations theoretically possible we can immediately diminish by a factor of 2 × 2, since we will constrain each of the two NPs to precede their respective verbs. The remaining 30 permutations are listed in Figure 2.1. Six of them are ruled out.[6]

2.2.3 Linguistic Analysis of the "Third Construction"

This wide variety of possible word orders that we observed in Section 2.2.2 is interesting from a formal as well as a linguistic point of view. den Besten and Rutten (1989) (for Dutch) and Santorini and Kroch (1990) (for German) present arguments that the third construction, rather than being an effect of clause union, is in fact a syntactic phenomenon that does not involve the morphological process of verb cluster formation. Under this analysis, the construction is related to two independently motivated syntactic operations, namely extraposition and scrambling. In this section, we will briefly present and discuss these analyses of the third construction. The aim of the discussion is to gain a better understanding of the problems that the data in Figure 2.1 present. We first summarize Santorini and Kroch's analysis of extraposition across complex verb sequences. We then show that it accounts for the acceptable sentences shown in Figure 2.1, and that the completely ungrammatical ones can be ruled out. However, for several of the bad sentences there are derivations not considered by Santorini and Kroch. We discuss possible ways of ruling out these sentences, and conclude that it must be a processing issue.

Santorini and Kroch (1990) discuss the third construction in German and propose an analysis based on scrambling and remnant extraposition. They reject Grewendorf's "morphological" analysis (based on (Evers, 1975)) in favor of the "syntactic" analysis proposed by den Besten and Rutten (1989).[7] There, the third construction is derived from a sequence of extraposition of the embedded clause and subsequent long-distance scrambling of its nominal argument. For example, sentence (18), repeated here as (21c), can be derived from (11), repeated here as (21a), via (17), repeated here as (21b), in which the embedded clause has extraposed.

[5] Note that *versprochen hat* is counted as one unit since no element may intervene between the two words, and since they may not be permuted.

[6] The judgments are those of the author, confirmed with other native speakers. They are only meant to be suggestive, of course. Because of processing issues, more rigorously collected experimental data is required; the "grammaticality judgment" is not a sufficient methodological tool.

[7] See Section 5.4.1, page 154 for a more detailed discussion of the "clause union" analysis.

No Extraposition Past verspricht		
(i)	Weil niemand das Fahrrad zu reparieren zu versuchen verspricht	ok
(ii)	Weil das Fahrrad niemand zu reparieren zu versuchen verspricht	?
(iii)	Weil niemand zu versuchen, das Fahrrad zu reparieren, verspricht	?
(iv)	Weil niemand das Fahrrad zu versuchen zu reparieren verspricht	?
(v)	Weil das Fahrrad niemand zu versuchen zu reparieren verspricht	?
(vi)	Weil das Fahrrad zu reparieren niemand zu versuchen verspricht	ok
(vii)	Weil das Fahrrad zu reparieren zu versuchen niemand verspricht	?
(viii)	Weil zu versuchen, das Fahrrad zu reparieren, niemand verspricht	?
(ix)	Weil zu versuchen das Fahrrad niemand zu reparieren verspricht	*
(x)	Weil das Fahrrad zu versuchen niemand zu reparieren verspricht	*
(xi)	Weil zu versuchen niemand das Fahrrad zu reparieren verspricht	*
(xii)	Weil das Fahrrad zu versuchen zu reparieren niemand verspricht	?
Extraposing the versuchen *Clause*		
(xiii)	Weil niemand verspricht, das Fahrrad zu reparieren zu versuchen	ok
(xiv)	Weil niemand das Fahrrad verspricht zu reparieren zu versuchen	?
(xv)	Weil das Fahrrad niemand verspricht zu reparieren zu versuchen	?
(xvi)	Weil niemand verspricht, zu versuchen, das Fahrrad zu reparieren	ok
(xvii)	Weil niemand verspricht, das Fahrrad zu versuchen zu reparieren	?
(xviii)	Weil niemand das Fahrrad verspricht zu versuchen zu reparieren	ok
(xix)	Weil das Fahrrad niemand verspricht zu versuchen zu reparieren	?
(xx)	Weil niemand das Fahrrad zu reparieren verspricht zu versuchen	?
(xxi)	Weil das Fahrrad zu reparieren niemand verspricht zu versuchen	?
(xxii)	Weil das Fahrrad niemand zu reparieren verspricht zu versuchen	?
Extraposing Only the reparieren *Clause*		
(xxiii)	Weil niemand zu versuchen verspricht, das Fahrrad zu reparieren	?
(xxiv)	Weil niemand zu versuchen das Fahrrad verspricht zu reparieren	*
(xxv)	Weil niemand das Fahrrad zu versuchen verspricht zu reparieren	*?
(xxvi)	Weil das Fahrrad niemand zu versuchen verspricht zu reparieren	*?
(xxvii)	Weil zu versuchen niemand das Fahrrad verspricht zu reparieren	*
(xxviii)	Weil zu versuchen niemand verspricht das Fahrrad zu reparieren	*?
(xxix)	Weil zu versuchen das Fahrrad niemand verspricht zu reparieren	*
(xxx)	Weil das Fahrrad zu versuchen niemand verspricht zu reparieren	*?

Figure 2.1: The Data

(21) a. ...daß niemand [das Fahrrad zu reparieren] versprochen hat

b. ...daß niemand t_i versprochen hat, [das Fahrrad zu reparieren]$_i$

c. ...daß niemand [das Fahrrad]$_j$ t_i versprochen hat [t_j zu reparieren]$_i$

The order of the two operations can also be reversed. Instead of first extraposing and then scrambling, one can first (string-vacuously) long-distance scramble the embedded NP into the matrix clause, and then extrapose the remnant clause. The intermediate structure is shown in (22).

(22) ...daß niemand [das Fahrrad]$_i$ [t_i zu reparieren] versprochen hat
...that no-one the bike$_{\text{ACC}}$ to repair promised has
...that no-one promised to repair the bike

Extraposition of the remnant *reparieren* clause then yields sentence (21c) (=(18)). If this analysis is correct, then we should be able to leave constituents in the extraposed clause. This is indeed the case:

(23) Weil niemand das Fahrrad versprochen hat, während der Mittagspause
because no-one$_{\text{NOM}}$ [the bike$_{\text{ACC}}$ promised has during [the noon-break]$_{\text{GEN}}$
zu reparieren
to repair

Because no-one has promised to repair the bike during lunch break

As Santorini and Kroch point out, "this analysis is very attractive since it relates and reduces the third construction to two independently motivated syntactic processes" (p.9). We therefore follow them in adopting Den Besten and Rutten's syntactic approach. This syntactic approach extends to the case of sentences with multiple embedded clauses, such as (xiv) from Figure 2.1, repeated here as (24):

(24) ? ...weil niemand das Fahrrad versprochen hat zu reparieren zu versuchen
...because no-one the bike has promised to repair to try

Here, we can first string-vacuously long-distance scramble the NP *das Fahrrad* out of the *zu reparieren* clause, and then extrapose the *zu versuchen* clause (along with the embedded remnant of the *zu reparieren* clause).

Given the instruments of (nominal and clausal) scrambling and extraposition, can we account for the data in Figure 2.1? Strikingly, all sentences judged to be good or marginal ("ok" or "?") are accounted for, and all sentences judged to be completely unacceptable ("*") are ruled out. We will discuss one sample derivation from each category in more detail.

Consider sentence (xxii) from Figure 2.1, reproduced here as (25):

(25) ? ...weil das Fahrrad niemand zu reparieren versprochen hat zu versuchen
...because the bike no-one to repair has promised to try

In order to derive this sentence, we start with the center-embedded D-Structure shown in (26):

(26) ...weil niemand [PRO [PRO das Fahrrad zu reparieren] zu versuchen] versprochen hat

We first (long-distance) scramble the NP *das Fahrrad* into the initial position of the matrix clause, yielding (27):

(27) ...weil [das Fahrrad]$_i$ niemand [PRO [PRO t$_i$ zu reparieren] zu versuchen] versprochen hat

Then, the remnant of the *zu reparieren* clause is scrambled string-vacuously into the matrix clause.

(28) ...weil [das Fahrrad]$_i$ niemand [PRO t$_i$ zu reparieren]$_j$ [PRO t$_j$ zu versuchen] versprochen hat

Finally, the remnant *versuchen* clause is extraposed, yielding (25).

As an example of a sentence that is ruled out, consider sentence (xxiv) from Figure 2.1, reproduced here as (29):

(29) * ...weil niemand zu versuchen das Fahrrad versprochen hat zu reparieren
...because no-one to try the bike has promised to repair

Here, we could first scramble the NP, then the remnant *reparieren* clause, extrapose the *reparieren* clause and finally scramble the remnant *[t_i zu versuchen]* clause. However, in this case, at least one trace will be unbound during at least one step of the derivation: the *[t_i zu versuchen]* clause must be c-commanded[8] by the extraposed *[t_k zu reparieren]$_j$* clause, which in turn must be c-commanded by the NP *[das Fahrrad]$_k$*. That is impossible, since *das Fahrrad* must be c-commanded by the *versuchen* clause since it is located in the string between the *versuchen* clause and the main verb (the latter ruling out a derivation in which the NP is – for whatever reason – adjoined to the right).

The analyses are summarized in Figure 2.2. The scrambled NP is always *das Fahrrad*. Scrambled or extraposed clauses may or may not contain all of their overt arguments; this is not indicated in the table. Extraposition always occurs with respect to the immediately embedding verb. (SV) means that the scrambling, at that point in the derivation, is string-vacuous. Other derivations may be possible (in particular, other orderings of these operations).

There is a problem. As can be seen from Figure 2.2, four sentences, (xxv), (xxvi), (xxviii), and (xxx), have been ruled "*?", but are given analyses by the scrambling and extraposition approach. Strikingly, one of these sentences is used by Santorini and Kroch to illustrate the contrasting behavior of full and remnant clauses extraposed across complex verb sequences. Their minimal pair, (50a) and (50b), corresponds to (xxiii) and (xxv) from Figure 2.1, reproduced here as (30) and (31). (Santorini and Kroch judge (31) as '*'; they do not make the distinction between '*' and '*?' that we do.)

[8]Throughout this thesis, we assume the first branching node definition of c-command: η_1 c-commands η_2 (η_1, η_2 nodes) if and only if the node immediately dominating η_1 also dominates η_2, but η_1 itself does not. Note that this is a purely structural definition, which does not refer to concepts defined within a linguistic theory.

No Extraposition Past versprochen hat		
(i)	ok	no movement
(ii)	?	scramble NP
(iii)	?	extrapose *reparieren*
(iv)	?	scramble NP (SV) + extrapose *reparieren*
(v)	?	scramble NP + extrapose *reparieren*
(vi)	ok	scramble *reparieren*
(vii)	?	scramble *versuchen*
(viii)	?	scramble *versuchen* + extrapose *reparieren*
(ix)	*	**Ruled out:** unbound trace in *[t_i zu versuchen]*
(x)	*	**Ruled out:** unbound trace in *[t_i zu versuchen]*
(xi)	*	**Ruled out:** unbound trace in *[t_i zu versuchen]*
(xii)	?	scramble NP + extrapose *reparieren* + scramble *versuchen*
Extraposing the versuchen *Clause*		
(xiii)	ok	extrapose *versuchen*
(xiv)	?	scramble NP (SV) + extrapose *versuchen*
(xv)	?	scramble NP + extrapose *versuchen*
(xvi)	ok	extrapose *versuchen* + extrapose *reparieren*
(xvii)	?	extrapose *versuchen* + scramble NP (SV) + extrapose *reparieren*
(xviii)	ok	scramble NP (SV) + extrapose *versuchen* + extrapose *reparieren*
(xix)	?	scramble NP + extrapose *versuchen* + extrapose *reparieren*
(xx)	?	scramble *reparieren* (SV) + extrapose *versuchen*
(xxi)	?	scramble *reparieren* + extrapose *versuchen*
(xxii)	?	scramble NP + scramble *reparieren* (SV) + extrapose *versuchen*
Extraposing Only the reparieren *Clause*		
(xxiii)	?	scramble *reparieren* (SV) + extrapose *reparieren*
(xxiv)	*	**Ruled out:** unbound trace in *[t_i zu versuchen]*
(xxv)	*?	scramble NP (SV) + scramble *reparieren* (SV) + extrapose *reparieren*
(xxvi)	*?	scramble NP + scramble *reparieren* (SV) + extrapose *reparieren*
(xxvii)	*	**Ruled out:** unbound trace in *[t_i zu versuchen]*
(xxviii)	*?	scramble *versuchen* + scramble *reparieren* (SV) + extrapose *reparieren*
(xxix)	*	**Ruled out:** unbound trace in *[t_i zu versuchen]*
(xxx)	*?	scramble NP + scramble *reparieren* (SV) + extrapose *reparieren* + scramble *versuchen*

Figure 2.2: Linguistic Analysis

(30) ok ...weil niemand zu versuchen versprochen hat, das Fahrrad zu reparieren
...because no-one to try has promised the bike to repair

(31) *? ...weil niemand das Fahrrad zu versuchen versprochen hat zu reparieren
...because no-one the bike to try has promise to repair

Santorini and Kroch derive (30) by first string-vacuously scrambling *[PRO das Fahrrad zu reparieren]* out of the *versuchen* clause (yielding a grammatical intermediate construction), and then extraposing it. (This is also the derivation given in Figure 2.2.) In order to rule out (31), they consider two derivations. In the first derivation, NP *das Fahrrad* scrambles out of the *reparieren* clause. Then, the remnant clause, *[PRO t_i zu reparieren]* is barred from extraposing from its position by subjacency, since the (remnant) *reparieren* clause could only extrapose with respect to *zu versuchen*, but not *versprochen hat*. On the other hand, the entire *reparieren* clause could first scramble, and then the NP *das Fahrrad* could scramble out of the scrambled clause, making it possible to extrapose the remnant clause without subjacency violation. But this derivation is ruled out by Huang's "Constraint on Extraction Domain" (CED) (Huang, 1982, p.505), which states that extraction can only occur out of governed domains.[9] However, after scrambling, the *reparieren* clause is no longer governed.

However, there is a third possible derivation, not discussed by Santorini and Kroch. Under this derivation, first the NP scrambles out of the *reparieren* clause, then the remnant clause scrambles, and finally it extraposes. If Huang's CED is interpreted as a condition on rule application rather than as a well-formedness condition on S-structure,[10] then it does not rule out this derivation. Nor is there a subjacency violation. Furthermore, the two intermediate constructions are licit. Finally, the combination of any two of three steps of the derivation is valid:

1. It could be argued that scrambling out of scrambled elements is ruled out on independent grounds (other than the CED). However, this cannot be the case. While (32) and (33) below are marginal, they are definitely better than (31). Both can only have been derived by first scrambling either the NP *das Fahrrad* or the adverb *morgen* 'tomorrow', and then the (remnant) clause. (An alternate derivation under which first the intact *reparieren* clause is scrambled and then the NP or adverb is scrambled further is ruled out by the CED.)

   (32) ? weil das Fahrrad niemand zu reparieren den Kindern versprochen hat
   because the bike no-one to repair the children has promise
   because no-one has promised the children to repair the bike

   (33) ? weil morgen niemand das Fahrrad zu reparieren den Kindern versprochen
   because tomorrow no-one the bike to repair the children promised
   hat
   has
   because no-one has promised the children to repair the bike tomorrow

[9] Of course, if the CED is derived from underlying principles, then the same remarks apply.

[10] Huang discusses this question. The scrambling facts provide evidence that the CED should be interpreted as a constraint on movement since interpreting it as a well-formedness condition would incorrectly rule out (32) and (33) below.

Furthermore, (xxii) in Figure 2.1 is acceptable, and can only be derived if we assume that we first scramble the NP out of the *reparieren* clause, and then string-vacuously scramble the remnant clause.

2. One may argue that extraposing clauses from scrambled positions is impossible. However, that would rule out sentence (xxiii) (Santorini and Kroch's (50a)), which is quite good, and can be derived only under the assumption that extraposition from scrambled positions is licit.

3. Finally, one could argue that only complete clauses can be extraposed, but this is clearly wrong, since in that case the third construction would not exist.

Thus, each step is licit and any combination of two of the three steps is licit. We conclude that the degradation can best be explained by appealing to processing phenomena. This is plausible since the four sentences in this category – (xxv), (xxvi), (xxviii), and (xxx) – appear to be somewhat better than those that are ruled out because of unbound traces (e.g., (xvii)).

To summarize, we have used an analysis based on a combination of scrambling and extraposition to account for the data presented in Section 2.2.2. This analysis adequately accounts for most of the sentences, but it appears that some extremely marginal sentences cannot be ruled out except by other considerations, such as processing facts.

2.3 Long-distance Topicalization

English long-distance *wh*-movement and topicalization data has played a central role in the development of the theory of grammar in Government and Binding Theory (GB) and related frameworks. It is therefore a bit surprising to discover that the equivalent data in an as closely related a language as German is extremely murky and difficult to evaluate. We consider three cases: first, extraction from embedded infinitivals, which is unproblematic; second, extraction from finite complementizer clauses; third, extraction from embedded V2 clauses. In all cases, topicalization obeys the usual island constraints (complex NPs, relative clauses, *wh*-islands; see (Webelhuth, 1989) for a complete examination of the relevant data).

2.3.1 Long-distance Topicalization from Non-Finite Clauses

Long-distance topicalization is fully acceptable out of embedded infinitival clauses, whether or not the embedded clause has extraposed. There is no intonation break (or comma) between fronted element and matrix finite verb.

(34) a. Dieses Buch$_i$ habe ich [PRO den Kindern t_i zu geben] versucht
this book$_{\text{ACC}}$ have I the children$_{\text{DAT}}$ to give tried
This book I have tried to give the children

b. Dieses Buch$_i$ habe ich versucht, [PRO den Kindern t_i zu geben]

We can simultaneously (long-distance) scramble and (long-distance) topicalize NPs out of a subordinate clause. For instance, in (35), the embedded verb, *geben* 'to give', has two overt nominal arguments, one of which has topicalized into sentence-initial position, and the other of which has scrambled beyond the matrix subject.

(35) [Dieses Buch]$_i$ hat [den Kindern]$_j$ bisher noch niemand [PRO t$_j$ t$_i$ zu geben]
[this book]$_{\text{ACC}}$ has [the children]$_{\text{DAT}}$ so far yet [no-one]$_{\text{NOM}}$ to give
versucht
tried

So far, no-one has tried to give this book to the children

2.3.2 Long-distance Topicalization from Complementizer Clauses

It has often been observed that there are regional (or perhaps ideolectal) variations within Standard German concerning the degree to which extraction is allowed. Generally speaking, speakers from the South are more liberal in allowing extraction than speakers from the North. Moreover, many speakers, irrespective of dialect, prefer extraction of non-specific NPs to that of specific NPs, with *wh*-extraction often preferred over topicalization. For some speakers, extraction of adjuncts is better than extraction of arguments.

Since an exhaustive examination of the data is beyond the scope of this investigation, we will simply assume that long-distance topicalization is possible in German, and that restrictions are due to individual differences in sensitivity to pragmatic and semantic factors rather than to restrictions in competence syntax.

Extraction out of embedded *daß*-clauses is exemplified by (36):[11]

(36) Was glaubt Peter daß der Lehrer den Kindern gegeben hat?
what$_{\text{ACC}}$ thinks Peter that [the teacher]$_{\text{NOM}}$ [the children]$_{\text{DAT}}$ given has

What does Peter think that the teacher gave the children?

As in English, there is no limit on the number of intermediate clauses.

(37) Was glaubt Peter daß Heike sagt daß der Lehrer den Kindern
what$_{\text{ACC}}$ thinks Peter that Heike says that [the teacher]$_{\text{NOM}}$ [the children]$_{\text{DAT}}$
gegeben hat?
given has

What does Peter think that Heike says that the teacher gave the children?

[11] We have omitted punctuation marks in the sentences.

2.3.3 Long-distance Topicalization from V2 Clauses

Long-distance *wh*-movement and topicalization are also possible from embedded V2 clauses, as shown in (38):[12]

(38) a. Was glaubst du hat der Meister zu reparieren versprochen?
what believe you has [the master]$_{\mathrm{NOM}}$ to repair promised

What do you think the master has promised to repair?

b. Dieses Fahrrad glaube ich hat der Meister zu reparieren versprochen
[this bike]$_{\mathrm{ACC}}$ believe I has [the master]$_{\mathrm{NOM}}$ to repair promised

This bike I think the master has promised to repair

While at first this case appears to be the analogue of extraction from embedded *daß*-clauses, there is a possible alternate analysis, namely as parenthetical. Such an analysis is motivated by the following constructions, which, presumably, are uncontroversially parentheticals:

(39) a. Dieses Fahrrad hat, glaube ich, der Meister zu reparieren versprochen

b. Dieses Fahrrad hat der Meister, glaube ich, zu reparieren versprochen

Grewendorf (1988, p.83ff) argues against a parenthetical analysis for (38), citing the following evidence.

1. Parentheticals are usually independent sentences.
2. In V2 sentences, parentheticals usually appear after the finite verb.
3. The verb of the parenthetical must match the modality (declarative, interrogative) of the host clause.
4. Intonationally, parentheticals are similar to independent utterances (pauses preceding it and a main accent).
5. The parenthetical is intonationally weakened with respect to the host clause.
6. Parentheticals admit criterial adverbs such as *übrigens* 'incidentally', sentential adverbs such as *wahrscheinlich* 'probably', and focus particles such as *sogar* 'even'.
7. Parentheticals do not behave like matrix clauses with respect to strong crossover phenomena.

Let us examine these differences to true extraction cases in detail. We must first address the question of the definition of the parenthetical (which Grewendorf does not do directly). If we assume that (i) is definitional, then sentences such as (39) cannot be parentheticals, and their proper analysis remains a mystery. If we assume (ii) to be definitional, then we have contributed

[12] We again omit punctuation marks.

nothing of interest to the analysis of extraction from V2-clauses such as (38). Clearly, the distinction in question – between a parenthetical and a "true" matrix clause – must refer to the syntactic analysis itself, and presumably the two options correspond roughly to the two structures sketched in Figure 2.3 (the details of the trees are not of interest here, only the general structure). In a matrix clause, the embedded clause appears as an argument of the verb in the *Mittelfeld* (though extraposed). A parenthetical, on the other hand, is adjoined into an independent sentence, like an adjunct. Presumably some sort of empty operator transfers the subcategorization requirement of the parenthetical verb to the clause into which the parenthetical is adjoined, as in a relative clause without overt relative pronoun.[13] This analysis would also explain the fact that parentheticals that subcategorize for clauses are verb-initial: the VF is occupied by the empty operator.

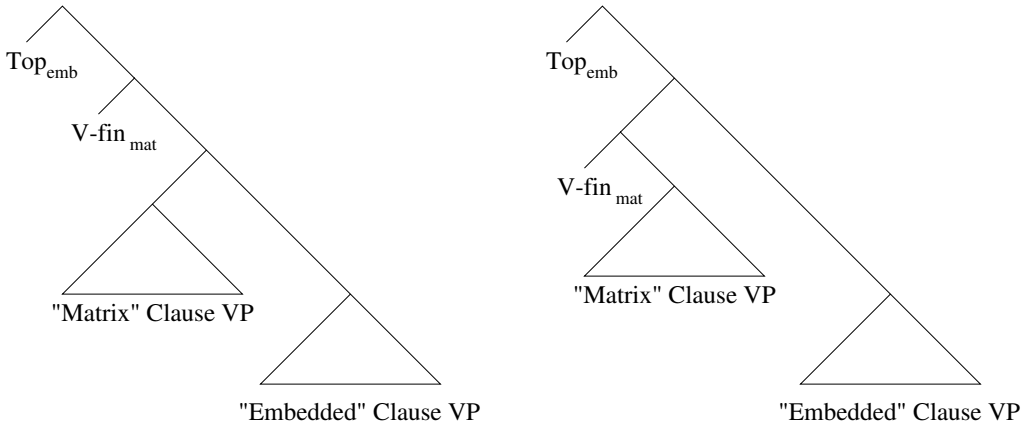


Figure 2.3: Standard embedding (left) and parenthetical (right)

Let us suppose that cases in which the quasi-matrix clause is a full sentence or in which it appears after the finite verb are uncontroversially parentheticals. (This assumption is motivated by the fact that, for such constructions, no plausible syntactic analysis as true matrix clauses is apparent.) The question thus is whether quasi-matrix clauses that neither are full sentences nor appear behind the finite verb are true matrix clauses or not. Let us consider Grewendorf's arguments with respect to these two possible analyses.

As we have seen, points (i) and (ii) beg the question. Grewendorf supports point (iii) by the following contrast (his (7-50), (7-51)):

(40) a. Peter hat – behauptet/*fragt Maria – zwei Maß Bier getrunken
Peter has claims/asks Maria two pints beer drunk

b. Wer hat – *behauptet/fragt Maria – zwei Maß Bier getrunken?
who has claims/asks Maria two pints beer drunk

c. Wer, behauptet/fragt Hans, hat zwei Maß Bier getrunken?

(40a) and (40b) are intended to show that the parenthetical verb must match the modality of the question, while a true matrix verb need not, as shown by (40c). However, the judgment on the

[13] Relative pronouns must be overt in German.

relevant contrast between (40b) and (40c) is not actually conclusive, as shown in the following example:

(41) Was hat, meinen Sie, am meisten zum Sturz des Kanzlers beigetragen?
what has think you at most to-the$_{\text{DAT}}$ fall [the chancellor]$_{\text{GEN}}$ contributed

What, in your opinion, contributed most to the fall of the chancellor?

Furthermore, Grewendorf fails to cite the parallel case for the alleged true matrix clauses, which in fact patterns with parentheticals:

(42) Peter, behauptet/*fragt Hans, hat zwei Maß Bier getrunken

Thus, once the full facts are taken into account, the criterion in (iii) becomes irrelevant.

While points (iv) and (v) – relating to intonation – are potentially crucial questions, we again take the data to be ambiguous. Clearly, sentences with V2-extraction pattern such as (38) can be given the intonation that Grewendorf ascribes to parentheticals, which is to be expected, since he sees such constructions as potentially ambiguous. More importantly, it seems that parentheticals of the type given in (39) can also be pronounced without breaks surrounding the parenthetical, or without "weakening". While intonation may provide crucial clues to the puzzle at hand, we conclude that the data requires rigorous phonological study (rather than mere introspection) in the context of a theory that integrates syntax and intonation.

Grewendorf supports point (vi) by observing the following contrast (adapted from his (7-53) and (7-54)):

(43) a. Peter hat – behauptet übrigens/vermutlich/auch Maria – zwei Maß Bier getrunken
Peter has claims incidentally/presumably/also Maria two pints beer drunk

Peter has drunk – {incidentally/presumably} Maria claims this {as well} – two pints of beer

b. *Wer — behauptet übrigens/vermutlich/auch Maria — hat zwei Maß Bier getrunken?
who claims incidentally/presumably/also Maria has two pints beer drunk

However, two variables have been varied between (43a) and (43b): parenthetical vs. true matrix clause, and topicalization vs. *wh*-movement. If we topicalize beyond an alleged true matrix clause, the contrast disappears, as Grewendorf points out, but this is expected since we may obtain a parenthetical interpretation of the construction in question. The crucial case, questions with an unambiguous parenthetical, he does not examine. Consider (44), which contrasts with (41).

(44) Was hat, meinen (*übrigens/*vermutlich/*auch) Sie (*übrigens/*vermutlich/*auch), am meisten zum Sturz des Kanzlers beigetragen?

We conclude that the unacceptability reported by Grewendorf may in fact be pragmatic infelicity arising from the clash between the question modality and the interpretation of these adverbials.

Grewendorf's most important argument is that relating to the strong cross-over ("Principle (C)") effect, since grammatical theory gives us a framework in which to interpret such data and to choose between the two phrase structures suggested in Figure 2.3. According to Principle (C) of Binding Theory (Chomsky, 1981), a pronoun cannot be co-referential with an R-expression (essentially, a referential NP) if it c-commands the R-expression. Principle (C) is illustrated in the following sentences. Here and in the following German examples, we use subscripts to indicate the intended co-referential readings.

(45) a. *He$_i$ likes Kevin$_i$

b. *He$_i$ thinks that most people aspire to liking Kevin$_i$

c. His$_i$ brother likes Kevin$_i$

d. When he$_i$ returns from Devon, we will visit Kevin$_i$

Grewendorf starts out by observing that Principle (C) effects are manifest in parentheticals ((46a) and (46b)), as they are in subordinate clauses ((46c) and (46d)) (adapted from Grewendorf's (7-58) and (7-59)):

(46) a. Peter$_i$ hat – glaubt er$_i$ – zwei Maß Bier getrunken
Peter has think he two pints beer drunk

Peter has, he believes, drunk two pints of beer

b. *Er$_i$ hat – glaubt Peter$_i$ – zwei Maß Bier getrunken

c. Peter$_i$ sagt, er hat zwei Maß Bier getrunken
Peter says he has two pints beer drunk

Peter says he has drunk two pints of beer

d. *Er$_i$ sagt, Peter$_i$ hat zwei Maß Bier getrunken

Grewendorf draws the conclusion that parentheticals behave like subordinate clauses, but his conclusion is of course only true with respect to the relation between the *upper* part of the "matrix" clause and the parenthetical. This result is unsurprising, given our two candidate analyses in Figure 2.3, and does not help us choose between them. What we are interested in is using Principle (C) effects to test for c-command relation between the material in the parenthetical or alleged matrix clause and material in the host (alleged subordinate) clause *below* the point of adjunction of the parenthetical. This test should distinguish between the two hypotheses: if we see Principle (C) effects, then the parenthetical analysis is implausible, while if we do not, the main clause analysis is implausible. Consider the following data:

(47) a. * Welches Konzert, glaubt er$_i$, hat Peter$_i$ gehört?
which concert believes he has Peter heard

Intended meaning: Which concert does Peter think that he heard?

b. Welches Konzert – er$_i$ kann sich nicht mehr daran erinnern – hat Peter $_i$ gehört?
which concert – he can himself not longer thereof remind – has Peter heard

Which concert – he cannot remember any more – has Peter heard?

c. * Welches Konzert hat, glaubt er$_i$, Peter$_i$ gehört?
which concert has believes he Peter heard

Intended meaning: Which concert does Peter think that he heard?

Grewendorf contrasts (47a) (his (7-62a)) to (47b) (his (7-62b)), and draws the conclusion that V2 extraction is structurally different from parentheticals. However, he fails to consider parentheticals that subcategorize for their host clause, as in (47c). Here, we see that the ungrammatical binding obtains as well, contrary to expectations. We speculate that the empty operator that is in SPEC(CP) of the parenthetical and that fulfills the clausal subcategorization requirement of the parenthetical verb contributes to a reconstruction effect, whereby elements in the parenthetical, which c-command the trace of the embedded clause, can appear to c-command elements in the host clause. But this means that binding facts can never distinguish between the two possible structures, and the facts are inconclusive.

The picture is further muddled by a very clear contrast, not considered by Grewendorf, in which V2 extractions pattern with parentheticals and not with *daß*-extraction. This concerns cases in which the (alleged) matrix clause contains a negative marker. Consider the following sentences:

(48) a. Ikonen glaubt {Hans nicht/keiner} daß Elke in Moskau kauft
icons believes Hans not/no-one that Elke in Moscow buys

Hans doesn't think/No-one thinks that Elke buys icons in Moscow

b. * Ikonen glaubt {Hans nicht/keiner} kauft Elke in Moskau
icons believes Hans not/no-one buys Elke in Moscow

Intended meaning: Hans doesn't think/No-one thinks Elke buys icons in Moscow

c. * Ikonen kauft, glaubt {Hans nicht/keiner}, Elke in Moskau
icons buys believes Hans not/no-one Elke in Moscow

Intended meaning: Hans doesn't think/No-one thinks Elke buys icons in Moscow

d. Ikonen glaubt Hans kauft Elke in Moskau
icons believes Hans buys Elke in Moscow

Icons Hans doesn't think that Elke buys in Moscow

As (48a) shows, long-distance topicalization past *nicht* is fine out of *daß* clauses, but, as shown in (48b), it is completely ruled out out of V2 clauses, just as it is ungrammatical to have negated parentheticals. (48d) is just the non-negated version of (48b), showing that the problem is indeed the negation in the alleged matrix clause. This contrast is surprising, if V2 extraction is brought

about by the same syntactic machinery as *daß*-clause extraction (say, successive cyclic movement out of the embedded SPEC(CP) position). Under a parenthetical analysis, we could suggest that the empty operator needed to satisfy the clausal subcategorization requirement interferes with an empty operator introduced by the negation.

We will draw the conclusion that *wh*-extraction from V2 clauses requires further study, and we will not include it in the set of phenomena that we wish to account for.

2.4 Differences Between *wh*-Movement and Scrambling

Topicalization and *wh*-movement are prototypical examples of what in the Government-and-Binding Theory (GB) of Chomsky (1981) is called "A′-movement". In A′-movement, a constituent is moved to a position in which it cannot receive either Case or θ–role, i.e., a non-argument position. Both must therefore be supplied through a chain anchored by a trace in an argument position. A certain cluster of properties has been identified which A′-movement displays. Chief among these is the behavior with respect to binding (anaphor, bound pronoun or "weak crossover", and Principle (C) or "strong crossover"). The behavior of A′-movement can descriptively be captured by the term "reconstruction": the moved element behaves for binding as if it had been returned to its original location. This contrasts with A-movement, a prototypical example of which is Raising: here, the element receives Case in its moved position (this being in fact the reason for movement), and no reconstruction effects can be seen.

Recently, there has been much debate in the GB literature over the proper analysis of scrambling in terms of the A/A′ distinction. This discussion is of interest to us to the extent that it actually bears on linguistic facts (as opposed to theory-internal arguments for either an A- or an A′-movement analysis). For example, Webelhuth (1989) and Mahajan (1989) identify properties of scrambling (in German and Hindi, respectively) that pattern after both A- and A′-movement. Webelhuth posits a third, mixed type of position, while Mahajan assumes that different landing sites in a very elaborate phrase structure have different properties. Frank et al. (1992) consider a set of data involving weak and strong crossover, systematically varying the grammatical function of the binder and the bindee. They argue that the observed A′-movement characteristics depend on the syntactic function of the binder: if the subject is the binder, then reconstruction effects are observed, and otherwise, they are not ("Subject Binding Condition"). These facts cannot be derived from the structural configuration. Previous arguments for or against specific movement types based on binding facts can be shown to arise from restricted data sets. This situation should be contrasted with that of true A′-movement: there, reconstruction effects are observed independently of the grammatical function of the binder (in particular, an A′-moved element can never bind an element it did not c-command from its base position, or perhaps an intermediate trace).

Given the analysis of Frank et al. (1992), we must draw the conclusion that scrambling and topicalization are different types of movement. Santorini (1990) and Lee (1993) have proposed that scrambling (in German and Korean, respectively) is in fact unitarily A-movement, and that the apparent reconstruction effects need to be explained by other means. Lee proposes a system in which scrambling is in fact case-driven, thus assimilating it to other cases of A-movement. We

will follow these authors in assuming a theoretical difference between the two movement types, but will not address the arcane issue of the A/A′-distinction, which hinges on definitions internal to a theory which is not adopted here (GB).

A second difference between scrambling and *wh*-movement relates to the freedom of the movement, and the way an instance of the movement types interacts with other movement. We have seen above in Section 2.2.1 that scrambling can be iterated. Furthermore, scrambling and *wh*-movement can occur in the same clause. In *wh*-movement, on the other hand, only one element can move. In scrambling, elements can move into the middle of a clause; *wh*-movement is always to the periphery of a clause. Finally, in German, scrambling cannot cross overt complementizers, while topicalization can. Müller and Sternefeld (1993) cite evidence that in Russian, on the other hand, scrambling is not clause-bound, while *wh*-movement is.

A third difference between scrambling and *wh*-movement relates to processing. Consider the following pair of sentences.

(49) a. Sentence with long-distance scrambling:

? Der Meister	hat	den Kühlschrank	niemandem	zu reparieren	versprochen
[the master]$_{\mathrm{NOM}}$	has	[the refrigerator]$_{\mathrm{ACC}}$	no-one$_{\mathrm{DAT}}$	to repair	promised
N_{11}	Aux_1	N_2	N_{12}	V_2	V_1

The master has promised no-one to repair the refrigerator

b. Sentence with long-distance topicalization:

Den Kühlschrank	hat	der Meister	niemandem	zu reparieren	versprochen
[the refrigerator]$_{\mathrm{ACC}}$	has	[the master]$_{\mathrm{NOM}}$	no-one$_{\mathrm{DAT}}$	to repair	promised
N_2	Aux_1	N_{11}	N_{12}	V_2	V_1

The master has promised no-one to repair the refrigerator

In both sentences, the embedded object *der Kühlschrank* has moved leftward beyond the matrix indirect object. But the case involving topicalization appears easier to process. If the two movements were of the same type, we would not expect such processing differences, especially in view of the fact that the distance of dislocation from the canonical position is greater in the more acceptable topicalization case.

From the binding facts, we conclude that there is no linguistic reason to implement *wh*-movement and scrambling by the same formal means, as would be the case if they showed uniform behavior with respect to binding. From the differences relating to the freedom of movement, we conclude that it would in fact be desirable to adopt different formal means to express the movement types, since otherwise we will necessarily have to resort to stipulations[14] in the linguistic system in order to derive these differences. Finally, using different formal means to achieve the two types of word order variations promises to enable us to account for the different processing facts within a principled theory of processing.

[14] We use this term in a non-pejorative sense; see Section 5.2.2.

2.5 Explanatory Goals for the Thesis

In this section, we summarize the data with respect to which we will judge the descriptive and explanatory adequacy of the theory that we will present.

- Scrambling is doubly unbounded.
- Scrambling of noun phrases, clausal scrambling, and extraposition may interact and a large number of sentences is possible.
- Some legal derivations involving scrambling and extraposition lead to degraded sentences. There is a "grey scale" of judgments, not a clear cut-off.
- Long-distance topicalization is possible out of both non-finite clauses and finite complementizer clauses.
- Long-distance topicalization is better than long-distance scrambling over a similar distance.

Chapter 3

Formal Analysis of the Data

In this chapter, we discuss options for the formal representation of German syntax, based on the data presented in Chapter 2. We start out by listing the properties that we want a formal system to have if we are to use it for describing syntactic competence (Section 3.1). Section 3.2 discusses the formal properties of scrambling and establishes a methodology for analyzing the adequacy of formal systems for the representation of syntax. Using this methodology, we show that context free grammars (CFG) and tree adjoining grammars (TAG), a tree rewriting system, are formally inadequate to derive the full range of scrambled sentences in German, and sketch two ways of extending the formal power of TAG in order to handle the linguistic phenomena in question. Section 3.3 pursues the first option, by investigating free-order TAG (FO-TAG), in which linear precedence constraints between nodes can be relaxed. Using data from extraposition and the third construction, we argue that this formalism cannot provide certain crucial constraints, and is not a good candidate. In Section 3.4, we investigate the other option, that of relaxing immediate dominance between nodes in a elementary tree. We show that "local" systems, in which derivations can be generated by a context-free grammar, are inadequate, and that we must allow less constrained derivations. Section 3.5 summarizes the results of this chapter, which are then used in the following chapter to formally define new systems which meet the desiderata argued for in this chapter.

3.1 Desiderata for the Formal Representation of Syntactic Competence

The question of what sort of mathematical formalism is best suited for the description of natural language syntax has long been a subject of debate. Generally speaking, the use of a restricted mathematical formalism in linguistic theory holds two promises:

- By constraining the set of sentences (viewed as strings or as structures) that can be derived within the mathematical formalism, the formalism limits the set of ungrammatical sentences that a linguistic theory expressed in the formalism must correctly rule out. Thus, much of

the work of predicting ungrammaticality can be shifted from the linguistic theory to the formalism in which it is expressed.

- Given a mathematical formalism, algorithms for the recognition of natural languages can be devised whose properties can be analyzed. For example, given humans' ease at parsing much natural language, parsing algorithms that take an amount of time exponential in the length of the input string are undesirable as models.

From these general goals, we can derive specific desiderata for mathematical formalisms that are to be used for the description of natural language syntax. Joshi (1987a) introduces the notion of *mild context-sensitivity* (also see Joshi et al. (1991)). A formalism $\mathcal{F}$ is mildly context-sensitive if it meets the following conditions:

- The set of languages generated by $\mathcal{F}$ is included in the set of context-sensitive languages.
- $\mathcal{F}$ generates only semi-linear languages.
- $\mathcal{F}$ generates only polynomially parsable languages.
- $\mathcal{F}$ generates the proper set of linguistic dependencies.

The second condition is motivated by the observation that sentences are formed from a finite set of initial elements (lexemes) in an incremental manner.[1] To these requirements we add two further, minor ones:

- $\mathcal{F}$ is closed under (positive) Kleene-star.
- $\mathcal{F}$ is closed under iterated substitution.

The requirement of closure under Kleene-star follows from the fact that an unlimited number of sentences in most natural languages (including German) can be conjoined to form a new sentence. The requirement of closure under iterated substitution follows from the observation that in many languages, including German, a nominal argument can be replaced by an operator that introduces a new clause (such as *the fact that*). This process can clearly be repeated on the nominal arguments of the newly introduced clause. We will take the liberty of extending the definition of mild context-sensitivity to include these two points, and will use the notion as a formal characterization of "suited for the description of natural language syntax".

In the early days of mathematical linguistics, context-free grammars (CFGs) seemed like a good candidate for linguistic use, since they are easily shown to be mildly context-sensitive. However, CFGs have turned out to be ill-suited for two reasons:

[1] Joshi et al. (1991) only require the weaker "constant-growth" property. However, we believe that the stronger requirement of semi-linearity is justified by the desire to write "lexicalized" grammars – grammars in which each elementary structure is associated with a single lexical item – in the formalism.

- In transformational syntactic theories, different syntactic structures (such as active and passive) are related derivationally. This has led to the addition of certain machinery to the basic CFGs, such as transformations. Transformations greatly increase the formal power of CFGs beyond mildly context-sensitive (Peters and Ritchie, 1973).
- It has long been argued that natural language is formally beyond the generative capacity of (transformation-free) CFGs. While many such arguments have been open to objections, we regard the argument of Shieber (1985), in which he uses the case-marked cross-serial dependencies observed in Swiss German, as settling the issue.

We conclude that we must go beyond CFGs in order to account for natural language, but we wish to stay within the bounds of mild context-sensitivity.

3.2 The Formal Analysis of Scrambling

This section discusses the formal properties of scrambling. We start out in Section 3.2.1 with a brief definition and discussion of tree adjoining grammars, a tree rewriting system. (For a more extensive introduction to TAG, see (Joshi, 1987a) and (Joshi et al., 1991).) Section 3.2.2 establishes a methodology for analyzing the adequacy of formal systems for the representation of syntax, called "derivational generative power". Using this methodology, we show in Section 3.2.3 that context free grammars (CFG) and tree adjoining grammars (TAG) are formally inadequate to derive the full range of scrambled sentences in German. In Section 3.2.4, we sketch two ways of extending the formal power of TAG in order to handle the linguistic phenomena in question.

3.2.1 Tree Adjoining Grammars: Review

We will first briefly review context-free grammars (CFG). While they have been all but abandoned as a basis for linguistic description in the linguistic and computational literature, they are quite familiar as a formalism and therefore useful as a starting point for the exposition of TAG. Recall that in a CFG, we have string rewriting rules that specify how a single symbol, called a nonterminal, can be rewritten as a sequence of other symbols. For example, in (1a) we rewrite the left-hand side string using the rewrite rule in (1b).

(1) a. $aAc \Longrightarrow axyc$

b. $A \longrightarrow xy$

In a complete derivation, we start out with a special nonterminal symbol, say S for "sentence", and successively apply string rewriting rules until we have no more nonterminal symbols left in the string, but only terminal symbols (such as *John likes Mary*). (In the process, we create a *derivation* tree which records how we rewrite each nonterminal symbol: the symbols we replaced it with appear as its daughters.)

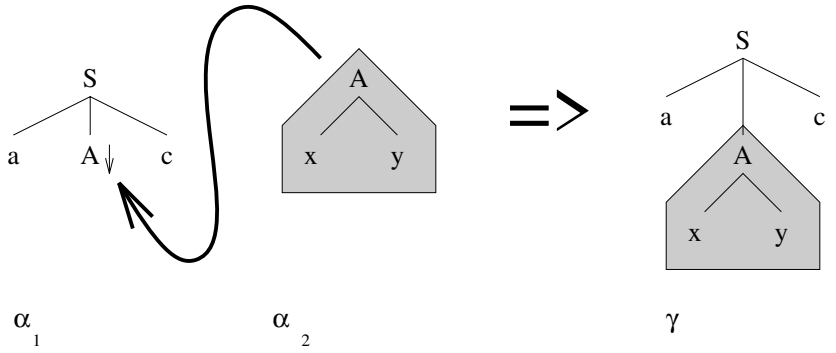


Figure 3.1: The Substitution Operation

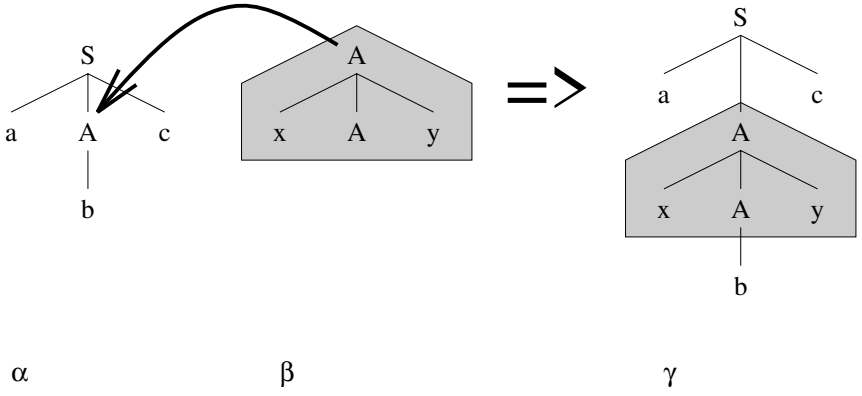


Figure 3.2: The Adjunction Operation

Just as CFG is a string rewriting system, TAG is a tree rewriting system: we start with elementary trees, and then can replace nonterminal nodes in the tree with entire trees. A TAG consists of a set of such elementary trees. Two tree combining operations are used to derive larger trees: substitution and adjunction. Substitution is shown in Figure 3.1: tree α_2 can be substituted into tree α_1 if the root node of α_2 has the same label as a non-terminal node on the frontier of α_1 which has been specially marked for substitution (a "substitution node"; substitution nodes are marked with down-arrows ($\downarrow$)). Adjunction is shown in Figure 3.2. Tree α contains a non-terminal node labeled A; the root node of tree β is also labeled A, as is exactly one non-terminal node on its frontier (the "foot node"). All other frontier nodes are terminal nodes or substitution nodes. We take tree α and remove the subtree rooted at its node A, insert in its stead tree β, and then add at the footnode of β the subtree of α that we removed earlier. The result is tree γ. We will call trees which have no footnode (i.e., which cannot be adjoined) "initial trees" and trees which do (i.e., which must be adjoined) "auxiliary trees". As we can see, sbstitution and adjunction are special cases of tree rewriting (i.e., replacing a node by a tree): substitution rewrites a node on the frontier of a tree, while adjunction rewrites an interior node, thus having the effect of inserting one tree into the center of another.[2] Formally, we define TAG as follows:

[2]In CFG, the difference between substitution and adjunction corresponds to the difference between rewriting a nonterminal symbol at the periphery of the sentential form and rewriting a nonterminal symbol in the middle of the sentential form. Note that if we restrict TAG to substitution, we obtain a system equivalent to CFG. Similarly,

Definition 1 *A* **Tree Adjoining Grammar** *(*TAG*) is a 5-tuple* (V_N, V_T, S, I, A),[3] *such that* V_N *and* V_T *are disjunct sets of nonterminal and terminal symbols, respectively,* $S \in V_N$ *is the start symbol,* I *is a set of initial trees, and* A *is a set of auxiliary trees.*

A derivation consists in choosing an initial tree, and then adjoining and substituting trees into it or each other. We do not give a formal definition for the derivation here, but refer to (Vijay-Shanker, 1987, p.16). The set of derived trees of a TAG G, $T(G)$ consists in all trees derived from I and A by substitution and adjunction such that the frontier, read off as a string in $(V_N \cup V_T)^*$, contains only terminal symbols. The string language derived by G, $L(G)$, is the set of all strings in V_T^* on the frontiers of the trees in $T(G)$.

In addition to the derived (phrase-structure) tree, a second structure is built up, the "derivation tree". In this structure, each of the elementary trees is represented by a single node. If the grammar is lexicalized, we can identify this node with the (base form of the) lexeme of the corresponding tree[4]. If a tree t_1 is substituted or adjoined into a tree t_2, then the node representing t_1 becomes a dependent of the node representing t_2 in the derivation tree. Furthermore, the arcs between nodes are annotated with the position in the "target tree" at which substitution or adjunction takes place. In the TAG literature, this annotation is in the form of the tree address of the node (using a formal notation to uniquely identify nodes in trees, without reference to linguistic concepts).

TAG is an appealing formalism for the representation of linguistic competence because it allows local dependencies (in particular the subcategorization frame and *wh*-dependencies) to be stated on the elementary structures of the grammar and to be factored apart from the expression of recursion and unbounded dependencies. These properties, which were first explored for linguistic use in (Kroch and Joshi, 1985; Kroch, 1987), are due to the *extended domain of locality* of TAG: the elementary structures are not simply context-free string rewriting rules, but trees. The extended domain of locality of TAG allows us to develop a lexicon-oriented theory of syntax: because the entire subcategorization frame of a lexical item can be represented in a single tree, we can "lexicalize" the grammar in the sense that every tree is associated with exactly one lexical item (be it a word or a multi-word idiom). Schabes (1990) investigates formal issues arising from lexicalization of TAG. It is this lexicalized version that has been used in the development of TAG grammars for English (Abeillé et al., 1990) and French (Abeillé, 1988).

3.2.2 The Derivational Generative Capacity of Formal Systems

The issue of formal complexity of natural language has been discussed in terms of two formal properties: weak and strong generative capacity.[5] Shieber (1985) and Miller (1991) argue that the set of sentences of Swiss German and Swedish, respectively, cannot be generated by a context-free grammar (CFG). The arguments rely on the weak generative capacity of the CFG. While

if we restrict CFG to peripheral rewriting, we obtain regular grammars.

[3] We assume that trees are uniquely identified, but do not include the labels in the definition. Furthermore, nodes in trees can be uniquely identified by stating the tree and their address within the tree.

[4] This is not exactly what is done in the TAG literature, but the difference is purely notational.

[5] This section is based on joint work with Tilman Becker and Michael Niv.

arguments based on weak generative capacity can show that a formalism is *not* adequate to generate a certain natural language, they cannot show adequacy. Consider, as an example, the case of Dutch. Like Swiss German, in embedded infinitivals Dutch displays cross-serial dependencies, meaning that a string of nouns is followed by a an equal number of verbs, where the first noun is the argument of the first verb, and so on (see Figure 3.3), as shown in (2):

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

...because Wim saw Jan help Marie teach the children to swim

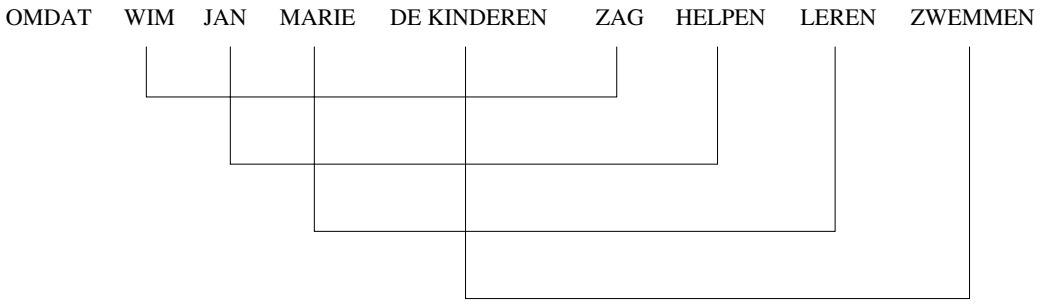


Figure 3.3: Cross serial dependencies in Dutch

The only relevant difference between Dutch and Swiss German is that in Swiss German, but not in Dutch, NPs carry case marking. It is the case marking that Shieber (1985) exploits in his argument to derive (a language closely related to) the copy language $\{ww \mid w \in \{a,b\}^*\}$, which can easily be shown not to be context-free. The lack of case marking in Dutch means that we can only map the set of relevant sentences on to the set $\{a^n b^n \mid n \in \mathbb{N}\}$, which clearly is context-free. Thus we do not have an argument that Dutch is not context-free. However, there is a strong intuition that if Swiss German is not context-free, then Dutch is not, either, since whatever linguistically appealing context-free analysis we could find for Dutch would also transfer to Swiss German. The case markings in Swiss German are only an indication of "deeper" properties, which Swiss German and Dutch share, and which are in fact what make both of these languages non-context free. Analyses based on weak generative power do not let us express these properties.

Bresnan et al. (1983) approach the question of Dutch from a different point of view. They argue that the set of syntactic trees of Dutch cannot be generated by a CFG. The arguments rely on the strong generative capacity of CFGs. They present the syntactic analysis for Dutch shown in Figure 3.4; the tree set implied by this analysis can be shown not to be derivable in a CFG. The problem with this analysis is that it depends on a particular linguistic analysis. If evidence for a different linguistic analysis is put forward, then any formal argument based on the previous analysis becomes irrelevant. For example, the Dutch case can also be analyzed as shown in Figure 3.5, as proposed by Zaenen (1979) and subsequently argued for by Kroch and Santorini (1991). However, whichever analysis we choose, the cross-serial dependencies are an irreducible property of this Dutch construction which remains constant across all existing and potential analyses in terms of trees. It is this constant property which we would like to exploit in a formal argument, not the particular shape that has been proposed for the trees. We conclude

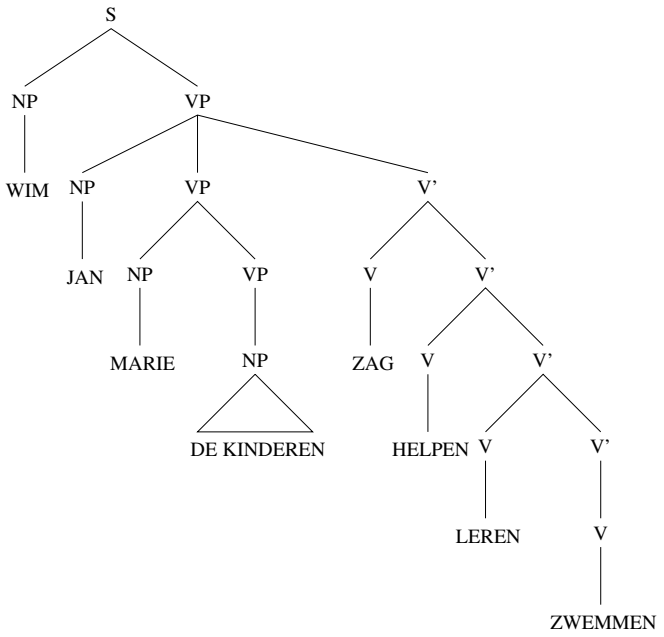


Figure 3.4: Tree structure for Dutch embedded infinitivals proposed by Bresnan et al. (1983)

that analyses based on strong generative power do not let us express the fundamental properties of the constructions we are interested in, either.

We therefore introduce a third criterion for the adequacy of formal systems, the "derivational generative capacity" of a system. The "deeper property" that we have alluded to above and that eludes formal analysis in terms of weak and strong generative capacity is that of the predicate-argument structure of verbs. This is exactly what the term "cross-serial dependencies" refers to. What characterizes the Dutch construction is the way in which the predicate-argument structures map onto the surface strings.[6] However, in the past such insight has not been exploited for the formal study of natural language, since most arguments discuss the adequacy of CFGs, and in a context-free framework the mapping between predicate-argument structure and syntactic structure cannot be expressed easily.

The picture is quite different in a TAG and related formalisms. Because of the extended domain of locality of a TAG, it is easy and in fact natural to localize predicate-argument relations within the elementary structures of the grammar. We will therefore assume that the elementary structures include exactly one lexical head and (positions for) its complements. We will refer to this restriction as the "co-occurrence constraint". It follows that dependent lexemes are contributed by two elementary structures in the grammar such that one is adjoined or substituted into the other. If we are interested in the relation between verbs and their nominal arguments (as we are in the Dutch case), we can assume that the nominal argument has already been substituted into the

[6]Observe that this is not a semantic notion *per se*, but rather an argument concerning the interface between semantics and syntax. The crucial notion is that of *syntactic dependency*, as used in dependency grammar (see, e.g., (Mel'čuk, 1988)). The derivational generative capacity of a system is really its ability to express certain dependencies; as argued in (Rambow and Joshi, 1994), a derivation in a lexicalized system such as a TAG defines syntactic dependencies. Thus, the notions are, for our purposes, equivalent.

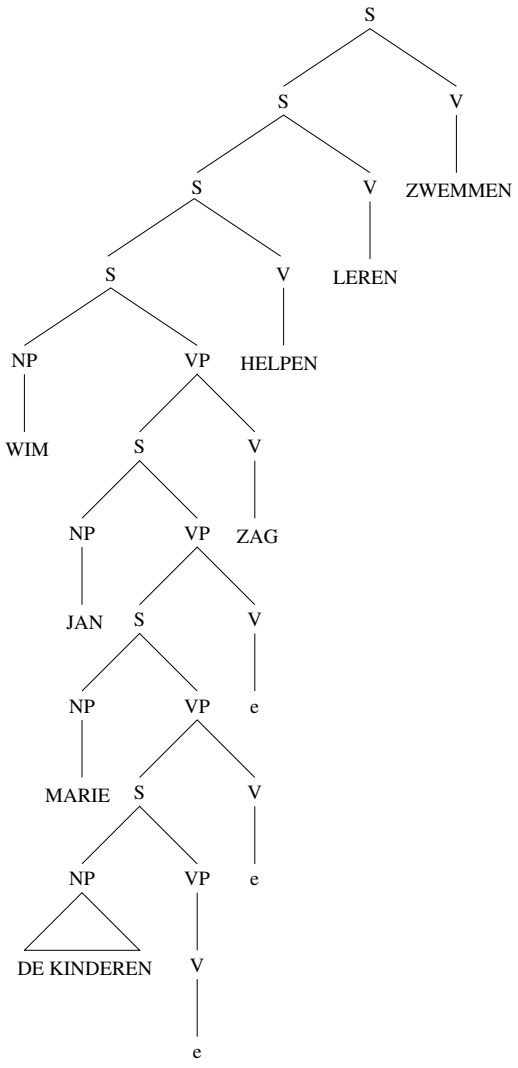


Figure 3.5: Tree structure for Dutch embedded infinitivals proposed by Zaenen (1979)

verbal tree, and then require that related noun-verb pairs be contributed by the same derivation step. Imposing predicate-argument relations among terminal symbols of a string therefore corresponds to imposing specific restrictions on the possible derivations of the string. If we consider the predicate-argument structure to be an essential part of the description of a natural language, then we may impose restrictions on the derivation structures in a particular system such that only proper dependencies are derived.

We define a system's "derivational generative capacity" in terms of its ability to generate sets of derivation structures. Notice that derivational generative capacity is orthogonal to strong generative capacity: it is possible for a system to generate a given set of strings with some given constraints on the derivation (and thus on the derivation structure), but not with some constraints on the derived structures, and vice versa. However, as in the case of strong generative capacity, if a grammar can derive a set of strings with some constraints on their derivation, then

of course it can derive those strings without the constraints, and hence two formalisms with the same derivational generative capacity also have the same . We now present a formalization of constraints on derivations, which we will then use to represent predicate-argument relations in scrambled strings in German.

Definition 2 *An* **indexed string** *over an alphabet T is a finite string w over T, along with a mapping from the symbols (i.e., symbol tokens) of w to the natural numbers. A string w that is equipped with such a mapping is written as w^{ind}, and the mapping is represented by superscripts in brackets on the symbols of w.*

Let w^{ind} be an indexed string and G a grammar in grammar formalism $\mathcal{F}$. We will say that G derives w^{ind} if there is a derivation of w in G (as defined by $\mathcal{F}$) in which all symbols bearing the same index are contributed in the same derivation step, and any derivation step contributes only symbols bearing the same index.

The definition of the derivation of indexed strings extends in the obvious way to sets of indexed strings.

Here are some examples:

1. In a CFG, all symbols bearing the same index (and no others) must be generated by the same application of a single context-free rule.
2. In a TAG, all symbols bearing the same index (and no others) must be generated by adjunction (or substitution) of the same elementary tree.

We can now formally represent the kind of restriction we would like to place on derivations of Dutch by saying that we want to derive the set of indexed strings $\mathsf{Dutch} = \{n^{[0]} \cdots n^{[k]} v^{[0]} \cdots n^{[k]} \mid k \geq 0\}$. It can straightforwardly be seen that there can be no CFG that derives Dutch, while the TAG grammar that has been proposed for Dutch (Joshi, 1987a; Kroch and Santorini, 1991) can derive the indexed string. We conclude that TAG, but not CFG, is a formalism that is adequate for the representation of (the relevant subset of) Dutch syntax.

Similarly, we will want the formal system that we choose for representing the syntax of German not only to generate the right string set, we will also require of it that it generate the string sets using the right derivations. (We will make only minimal assumptions about the derived trees.) In the following section, we use the notion of derivational generative capacity to show that TAGs are not powerful enough to derive scrambling in German.

3.2.3 The Formal Power Needed for Scrambling

Let us consider sentence (35) from Chapter 2, repeated here (somewhat simplified) for convenience.

(3) [Dieses Buch]$_i$ hat [den Kindern]$_j$ niemand [PRO t$_j$ t$_i$ zu geben] versucht
[this book]$_{\text{ACC}}$ has [the children]$_{\text{DAT}}$ [no-one]$_{\text{NOM}}$ to give tried

No-one has tried to give this book to the children

If we map the overt lexical items in the German string into a formal alphabet representing the lexical categories (we map NPs to n and verbs and auxiliaries to v), and add indices representing the desired dependencies, we obtain the following indexed string:

(4)
$$\begin{array}{lcccccccc} & \text{Dieses} & \text{Buch} & \text{hat} & \text{den Kindern} & \text{niemand} & \text{zu geben} & \text{versucht} \\ w^{ind} = & n^{[2]} & & v^{[1]} & n^{[2]} & n^{[1]} & v^{[2]} & v^{[1]} \end{array}$$

It can easily be seen that the indexed string w^{ind} above cannot be derived by any CFG. It cannot be derived by any TAG, either, since the adjunction of one tree into another creates at most five distinct segments (see Figure 3.2, page 36), while we would need six distinct segments to derive string $w^{ind} = n^{[2]}\ v^{[1]}\ n^{[2]}\ n^{[1]}\ v^{[2]}\ v^{[1]}$. A similar argument, based only on scrambling, is given in (Becker and Rambow, 1990). What is striking about these arguments is that a *single string* is sufficient to show that a given indexed language cannot be derived by a given formalism. If we are arguing about the weak or strong generative capacity of a formalism, any finite set of strings or structures can of course always be generated. This is not the case when talking about the derivational generative capacity, since now we are concerned with a particular derivation.

3.2.4 Extending TAG to Handle Scrambling

If TAG cannot derive certain German sentences, then we must go beyond TAG. At the same time, we would like to preserve those aspects of TAG that make it appealing as a formalism for the representation of natural language syntax, principally, its extended domain of locality.

Becker and Rambow (1990) and Becker et al. (1991) explore two options:

- Linear precedence between nodes can be relaxed, while maintaining immediate dominance. This leads to a type of ID/LP TAG. We discuss this option in Section 3.3.
- Immediate dominance between nodes can be relaxed, while maintaining linear precedence between nodes. This leads to a version of multi-component TAG with dominance constraints. We explore this option in Section 3.4, and in the remainder of the thesis.

It should be noted that in all versions of TAG, the dominance structure of an elementary tree can be altered by adjoining another tree. Therefore, the term "LD/LP TAG" is often used, where "LD" stands for "linear dominance". However, if we identify a node before adjunction with both the root node and the foot node of the adjoined tree after adjunction (for example, by splitting all nodes into pairs of nodes as done in (Vijay-Shanker, 1992)), then no *immediate* dominance relations are altered by adjunction. It is in this sense that we claim that immediate dominance is preserved in ID/LP TAG formalism we will introduce in the next section, as it is in TAG.

3.3 Relaxing Linear Precedence: FO-TAG

3.3.1 The FO-TAG Formalism

The FO-TAG formalism (Becker and Rambow, 1990; Becker et al., 1991) is closely based on the LD/LP-TAG framework presented by Joshi (1987b). A FO-TAG grammar consists of a set of elementary structures. Each elementary structure is a pair consisting of a linear dominance (LD) structure (i.e., an unordered tree) and corresponding linear precedence (LP) rules. The LD structure (which will, imprecisely, be referred to as a "tree" here) is either an initial or an auxiliary tree. The LP rules may relate any two nodes of the tree unless one linearly dominates the other. However, these precedence rules can only be stated with respect to the nodes of a single elementary structure; it is not possible to relate nodes in different structures.

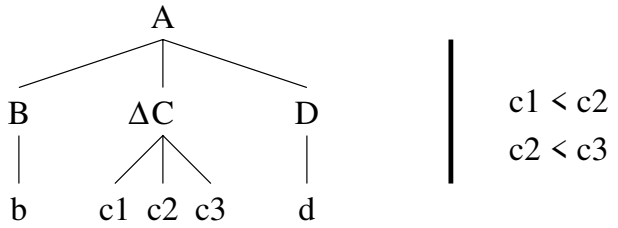


Figure 3.6: Sample FO-TAG Tree with Integrity Constraint

Adjunction and substitution are defined as in the case of regular TAG, except that the linear precedence rules of the derived tree must be stated separately. The set of LP rules of the derived tree is the union of the sets of LP rules of the two original trees. (In the case of adjunction, copies of those rules affecting the node at which adjunction takes place must be added to reflect the "splitting" of that node.) Furthermore, any LP rule that relates the node at which adjunction or substitution takes places in inherited by all nodes in the adjoined or substituted tree[7]. Note that if a node η_1 at which adjunction or substitution takes place is unordered with respect to some other node η_2 in its own elementary tree, then any two nodes in the tree adjoined or substituted at η_1 are individually unordered with respect to η_2, meaning that one could precede η_2 and the other follow η_2. This effect, "indeterminacy", is overridden by the "integrity constraint", written as Δ. If we have ΔX for some node X, then any node not in the subtree rooted at X must either precede or follow every node in the subtree rooted in X. For example, consider the tree in Figure 3.6. The bar separates the specification of the ID structure on the left from the LP rules on the right. Node C has an integrity constraint. Therefore, this tree represents the strings $bdc_1c_2c_3$, $dbc_1c_2c_3$, $bc_1c_2c_3d$, $dc_1c_2c_3b$, $c_1c_2c_3bd$, and $c_1c_2c_3db$. However, the tree does not represent, for example, the string $bc_1dc_2c_3$, since c_1, c_2, and c_3 are all dominated by C, and d must either precede all of them or follow all of them.

[7] In (Becker and Rambow, 1990), we do not postulate default inheritance of LP rules. Instead, we propose an additional "inheritance operator" to force inheritance in certain cases.

3.3.2 A Linguistic Example

In this section, we give trees in the spirit of a lexicalized grammar that show how scrambling would be handled by an FO-TAG. The trees are for sentence (5).

(5) Peter glaubt, [daß ich$_i$ [den Kühlschrank]$_j$ dem Kunden [PRO$_i$ t$_j$ zu reparieren]
Peter believes that I [the refrigerator]$_{\text{ACC}}$ [the client]$_{\text{DAT}}$ to repair
versprochen habe]
promised have

Peter believes that I have promised the client to repair the refrigerator

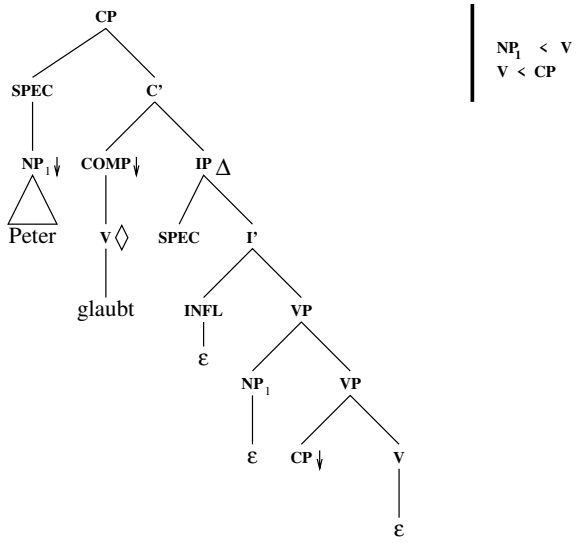


Figure 3.7: Top-level matrix clause

In order to derive the sentence, we substitute the tree for the immediately embedded clause (Figure 3.8) into the tree of the matrix clause (Figure 3.7)[8], and then substitute the tree for the most deeply embedded clause (Figure 3.9) into the intermediate derived tree. The result is shown in Figure 3.10. Remember that the depicted ordering of the leaves is but one of the admissible orderings. Several explanatory remarks are in order:

1. The trees shown in Figures 3.7, 3.8, and 3.9 are the initial clausal trees. The anchors are, as usual, marked by $\diamond$. To make the trees easier to read, the nominal arguments have already been substituted. The nodes at which they are substituted are still marked by $\downarrow$.

2. We assume that the trees shown in Figures 3.7, 3.8, and 3.9 are present in the grammar (without the substituted arguments, of course).

3. SPEC and PRO may be thought of as nonterminals that immediately dominate the empty string ε (not shown in the diagrams to avoid clutter).

[8]The finite verb has moved (within the elementary tree) to Comp, following the standard analysis.

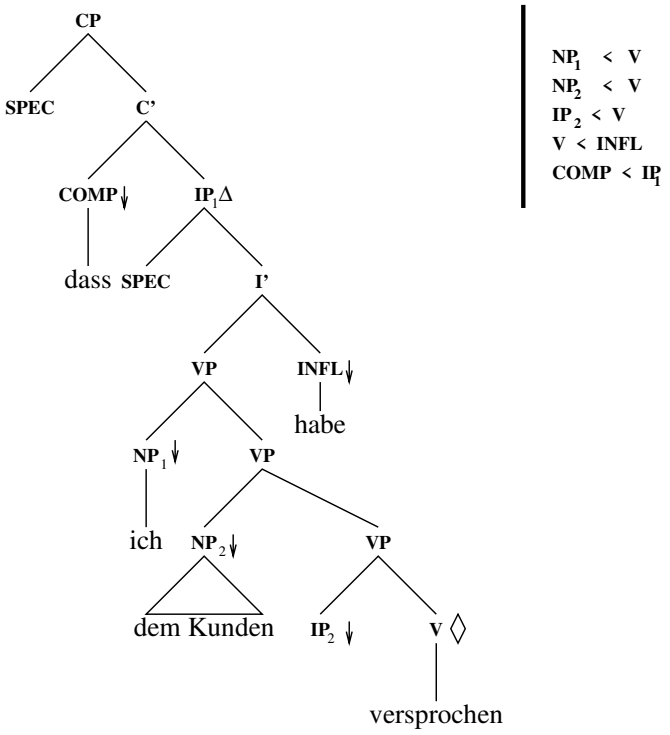


Figure 3.8: First embedded clause

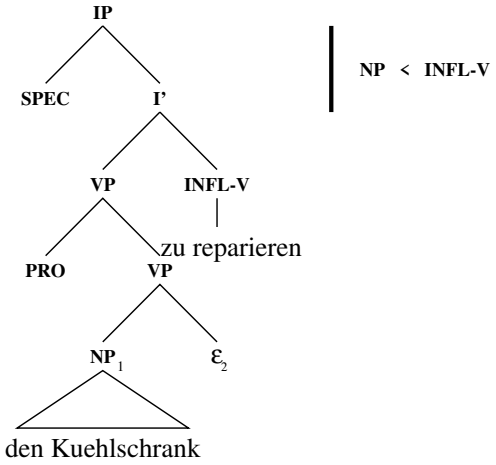


Figure 3.9: Second embedded clause

4. In clauses such as that represented by Figure 3.8 which require an auxiliary verb, this auxiliary is substituted into INFL. Features make sure that the right auxiliary is substituted, and, as always, handle agreement.

5. IPs that are sisters of overt COMPs are marked with the integrity operator. This reflects the fact that CPs (with overt COMPs) and relative clauses are islands for scrambling in German, as well as the verb-second constraint in the matrix clause.

It can easily be seen that all six orderings of the three overt arguments of *reparieren* and *versprochen* are possible, while all three need to follow the complementizer *daß* and precede the most deeply embedded verb *zu reparieren*.

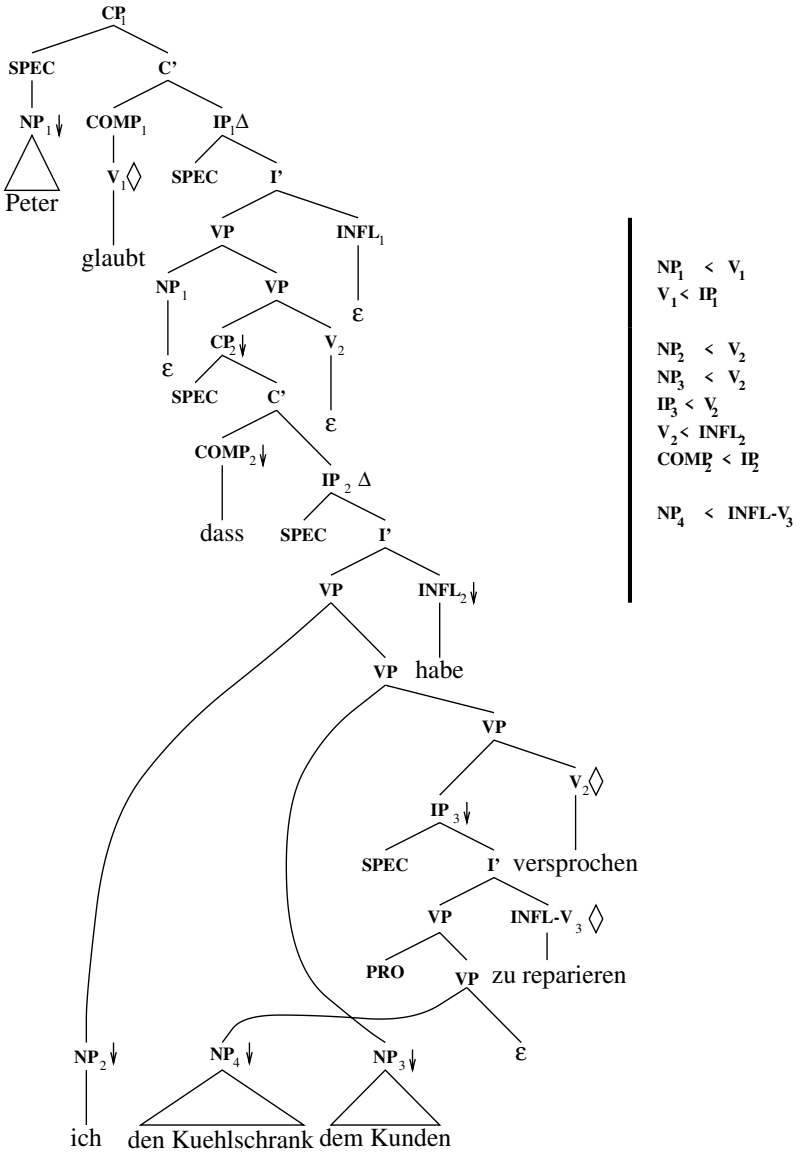


Figure 3.10: Derived tree

3.3.3 Linguistic Interpretation

A tree in a FO-TAG grammar does not have a "default ordering". The set of immediate dominance and linear precedence relations that hold within the tree specify a number of possible orderings of the terminal nodes of the tree. None of these are privileged: there is no base order or deep structure from which other orderings or structures are derived. As a corollary of this definition,

it follows that it is impossible to define a "trace" in the case of word orders that result from underspecified LP rules. This is in contrast to the MC-TAG formalism and its variants (MC-TAG-DL), in which the linguist may choose to include a trace or not. What is the linguistic implication of this impossibility of including traces for constituents "moved" by underspecified LP rules?

The main motivation for marking the original site of constituents (whichever type of movement they have undergone) is θ-theory: an argument of a verb must receive its θ-role from the verb under some structural conditions. If the argument moves outside the clause, or even within the clause, the structural conditions no longer hold. Therefore, a trace must be left in the position which is (structurally) assigned a θ-role, and the role is assigned to the chain formed by the NP and its trace. However, this argument is compelling only for those linguistic theories whose underlying formalism has a restricted domain of locality, such as Government and Binding Theory (GB), whose underlying formalism can be said to be a context-free grammar. In a formalism with an extended domain of locality, such as TAG, multi-component TAG (MC-TAG), or FO-TAG (which all have comparable domains of locality), θ-roles need not be assigned structurally under sisterhood. The only structural requirement is that θ-assignment can be stated with respect to that extended domain, i.e., the elementary tree. Thus, we need not worry about θ-assignment in an FO-TAG analysis, and can conclude that traces are not necessary for that purpose.[9]

However, in the case of A$'$-movement (such as *wh*-movement and topicalization), traces play an additional role: A$'$-movement is generally taken to license reconstruction, meaning that binding characteristics of the moved constituent are derived from those of the constituent in its underlying, unmoved position. Thus, in the case of A$'$-movement, we need the notion of an underlying structure, and we need the trace in order to relate moved elements to their underlying positions. We conclude that movement by relaxed LP rules in FO-TAG cannot be A$'$-movement, since FO-TAG cannot represent the underlying order of constituents "moved" by LP relaxation since there is no such notion. *wh*-movement and topicalization, on the other hand, can be handled differently, since no formal considerations force us to use the relaxation of LP rules. Thus we can choose to implement A$'$-movement as has been done in simple TAG for English, namely by tree-internal structural movement, leaving a trace in the base generated position. (This can be seen in Figure 3.7, where the subject in SPEC(CP) has left a VP-internal trace.)

To summarize: scrambling must be handled in a FO-TAG grammar by relaxation of LP rules, since other analyses, such as one similar to the analysis of *wh*-movement of Kroch (1987), are formally inadequate for long-distance scrambling. Furthermore, any movement by relaxed LP rules cannot be A$'$-movement. If LP rules are relaxed for long-distance scrambling, then necessarily clause-internal scrambling is also allowed by the same mechanism. We conclude that the choice of FO-TAG as an underlying formalism makes the linguistic prediction that scrambling is a unitary phenomenon and is not A$'$-movement, be it clause-internal or long-distance.

This prediction is at odds with some of the recent linguistic literature on scrambling. As discussed in Section 2.4, Webelhuth (1989) and Mahajan (1989) identify properties of scrambling that pattern with both A- and A$'$-movement. However, Frank et al. (1992) show that the observed A$'$-

[9] In the case of MC-TAG or MC-TAG-DL, the verb assigning a θ-role to a scrambled argument can only be determined from the derivation tree, while in the case of FO-TAG, the relationship between a verb and its arguments is immediately obvious in the derived tree as well.

movement characteristics (with respect to binding) depend on the syntactic function of the binder ("Subject Binding Condition"), and cannot be derived simply from the structural relation of antecedents and their traces at S-Structure (which FO-TAG does not express). Since the Subject Binding Condition can equally well be expressed in FO-TAG, the apparent A′-characteristics of scrambling do not constitute evidence against an FO-TAG analysis. We refer to (Rambow, 1992a) for a more detailed discussion.

We conclude from the very brief discussion in this section that, despite the fact that the structures of FO-TAG differ markedly from those of most contemporary linguistic theories, the unusual structures of FO-TAG do not automatically disqualify it from serving of the basis for the formulation of linguistically meaningful grammars. Furthermore, the formalism imposes constraints on linguistic analyses which are plausible and interesting.

3.3.4 Problems with a FO-TAG analysis

The representational adequacy of the FO-TAG formalism becomes questionable when confronted with the Third Construction. When considering the case of a bi-clausal sentence such as the one below, using the FO-TAG mechanism to account for the possible word order variations seems very attractive.

(6) ...daß niemand das Fahrrad zu reparieren verspricht
...that no-one [the bike]$_{\text{ACC}}$ to repair promises
...that no-one promises to repair the bike

The most natural way to represent extraposition in FO-TAG is to omit the LP rule that states that the embedded clause must precede its governing verb. This in a superficial way captures the idea of extraposition: if the subcategorized clause precedes the governing verb, we have center-embedding, and if it follows the governing verb, then we have extraposition. In fact, a grammar consisting of two elementary structures, one for each clause, in which only the NPs are specified to precede their respective verbs, would generate exactly all six licit permutations of sentence (6). However, this approach will not generalize. If we introduce an intermediate clause, then this approach will again generate all permutations in which the NPs precede their respective verbs, but, as we have seen in Chapter 2, six of those 30 sentences are not grammatical. The problem for FO-TAG arises from the fact that the leftward "movement" of NPs out of extraposed clauses is restricted. Recall that nominal arguments are specified to precede their governing verb, and otherwise are completely free. This works well for center-embedded structures, since it allows the completely free ordering of all nominal arguments, as we desire. However, if a clause is extraposed, ungrammaticality results if an NP is placed between two verbs from less deeply embedded clauses. For example, consider sentence (xxiv), reproduced here as (7):

(7) * Weil niemand zu versuchen das Fahrrad verspricht zu reparieren
Because no-one to try the bike promises to repair

Here, the nominal argument *das Fahrrad* of the most deeply embedded clause has placed itself between the matrix verb *verspricht* and the intermediate verb *zu versuchen*. The sentence is completely ungrammatical, despite the availability of the FO-TAG derivation given Figure 3.11.

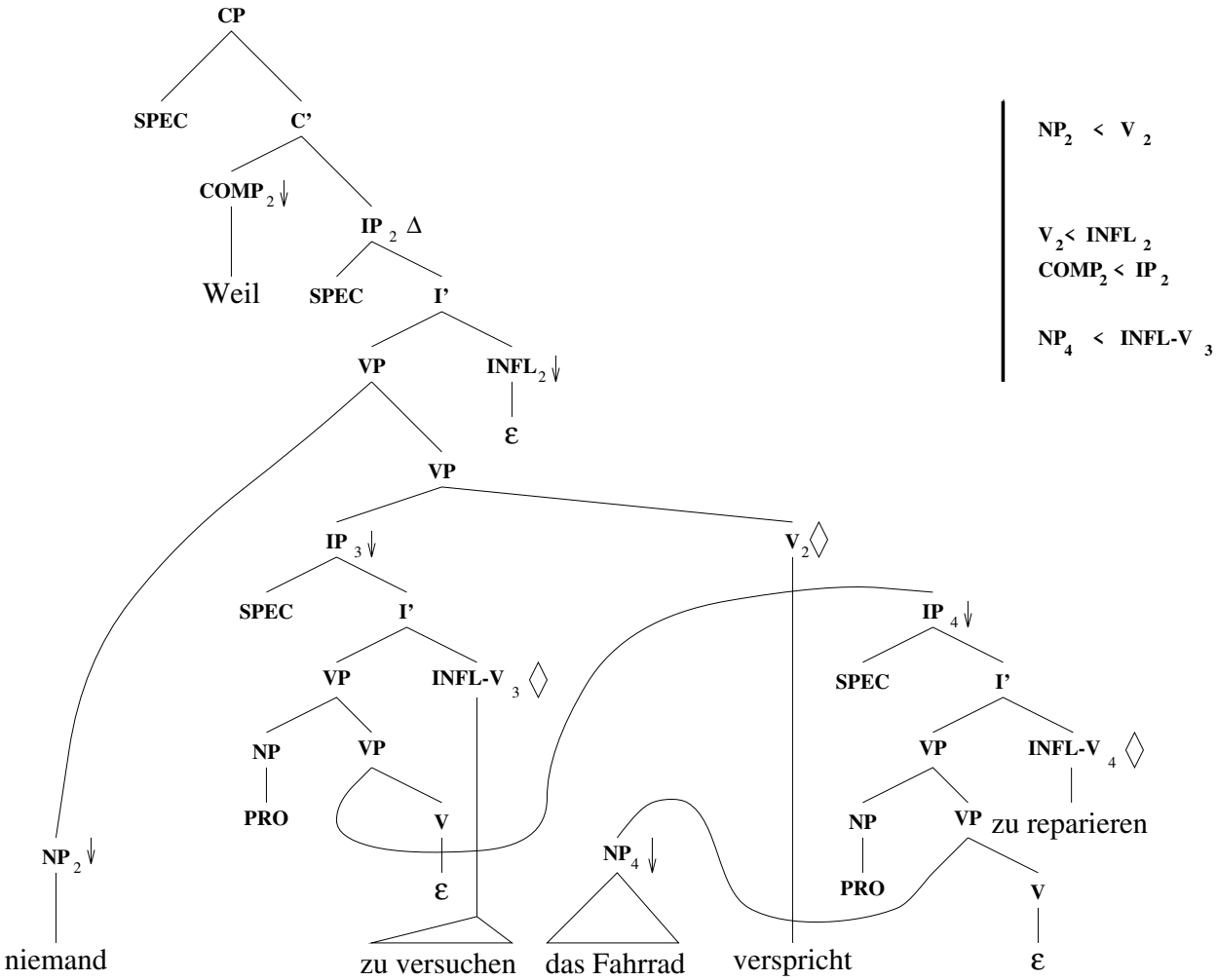


Figure 3.11: FO-TAG tree for ungrammatical sentence (xxiii), using relaxed LP for extraposition

Can we prevent this word order in FO-TAG? Since we want to maintain the integrity of the two-verb sequence *zu versuchen verspricht*, we might try to use the integrity constraint Δ. The smallest constituent containing both of these verbs is the VP immediately dominating IP_3 and V_2. However, this node also dominates the extraposed clause, and since placing an integrity constraint on this VP node does not affect what happens *within* the subtree it dominates, we will not be able to rule out the ungrammatical word order. Suppose we instead place the integrity constraint on the root node of the extraposed clause, IP_4. While this will in fact rule out sentence (7), it also rules out the "classical" Third Construction cases, i.e. any licit word order derived from "moving" an NP out of an extraposed clause, such as sentence (iv). Finally, if we place the integrity constraint on the root node of the intermediate clause, IP_3, we falsely rule out sentence (xxiii), in which the *reparieren* clause is extraposed behind the matrix verb. We conclude that handling extraposition by relaxing linear precedence cannot be successful.

We may instead wish to explore an option under which extraposition is not the result of underspecified LP rules, but of different choices in the derivation of the complex sentence. Let us

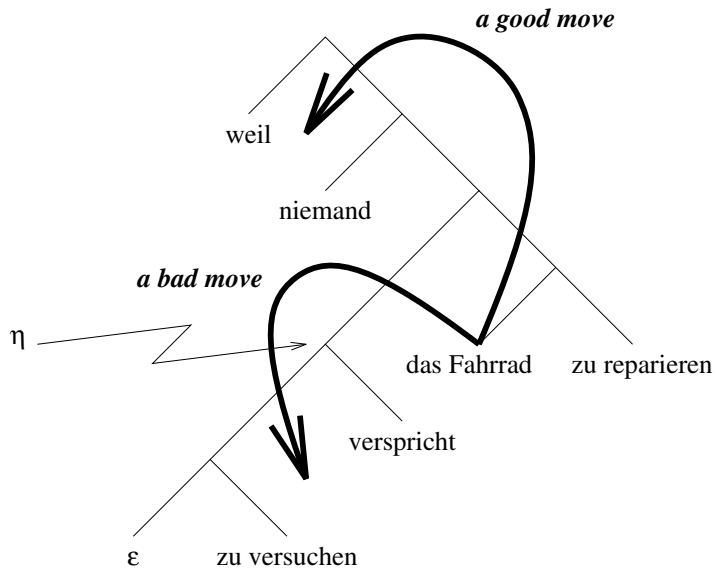


Figure 3.12: FO-TAG tree (schematic) for ungrammatical sentence (xxiii), using a different structural derivation for extraposition

assume that the extraposed clause is right-adjoined above the verb behind which it is extraposed (in our example, *verspricht*). We leave aside any details (in particular as they relate to the interpretation of this clause as the argument of the intermediate verb); the resulting structure is shown schematically in Figure 3.12. Can we now rule out the ungrammatical sentence using the integrity constraint? We now have a node, marked η in the diagram, which dominates the verb sequence whose integrity we need to protect (*zu versuchen verspricht*), and which does not dominate the extraposed clause. Clearly, if we mark this node with the integrity constraint, we rule out sentence (7), while allowing the Third Construction cases, i.e., long-distance scrambling of the NP *das Fahrrad* from the extraposed clause into the matrix clause to obtain word orders such as sentence (iv). However, this approach is too restrictive as well. Consider a case in which the intermediate clause also has a nominal argument:

(8) a. Weil niemand dem Günther zu versprechen beabsichtigt, das Farhrrad
because no-one [the Günther]$_{\text{DAT}}$ to promise intends the bike
zu reparieren
to repair

Because no-one intends to promise Günther to repair the bike

b. Weil dem Günther niemand zu versprechen beabsichtigt, das Farhrrad zu reparieren

If we place an integrity constraint on the node dominating the matrix verb and the intermediate clause, we not only keep NPs from the extraposed clause form placing themselves between intermediate and matrix verb, we also keep intermediate NPs from scrambling out of their clause. However, this is possible as shown in (8b) above. What is needed is a "permeable" integrity constraint: the sentence elements can leave the constituent, but no others can enter. The defini-

tion of such a constraint turns out to be overly complex. The definition of the simple integrity constraint is straightforward, since we can specify a string (the leaves of the subtree dominated by the node marked with the integrity constraint) which must appear unbroken (though perhaps permuted in some way) in any string derived from the structure. However, the "permeable" integrity constraint would have to allow certain ways of breaking up a string, but not others. For example, consider a string $a_1b_1a_2b_2$, where a_1a_2 form a constituent, and b_1b_2 form a constituent. It cannot be determined from inspecting this string whether b_1 has placed itself between a_1 and a_2, or whether a_2 has placed itself between b_1 and b_2. But such a distinction is crucial for the "permeable" integrity constraint. Clearly, it can only be made on the basis of the immediate dominance structure.

In a traditional phrase structure representation, there is no problem in representing the relevant restriction: the displaced NP must c-command its governing verb (or its own trace, if traces are represented). The "permeable" integrity constraint would essentially amount to encoding enough structure in order to capture the notion of c-command. We may conclude that traditional phrase-structure representations are better equipped to represent the syntactic constraints relevant to the third construction, and that multi-component TAG should be preferred over FO-TAG. While we will not pursue the linguistic application of FO-TAG any further, we note that several notions, including the (simple) integrity constraint, will be adapted to the formal systems we explore in Chapter 4 and will be used in the linguistic analysis given in Chapter 5.

3.3.5 Incremental Generation

We will now briefly address the issue of processing, specifically that of incremental generation. It has been argued that incrementality in generation implies that linear precedence of constituents must be stated independently from immediate dominance (de Smedt, 1990, p.171). In this spirit, Harbusch et al. (1991) use a version of TAG in which the order of sister nodes is determined by a separate component. Thus, at first it may seem that FO-TAG is better equipped to represent syntactic knowledge for incremental generation than MC-TAG. However, what is really at issue in incremental generation is not the linear precedence of (sister) *nodes*, but of *constituents* (arguments and adjuncts). In context-free based systems, these two questions coincide. In TAG and its variants, the extended domain of locality allows us to express grammatical function separately from phrase-structure (namely in the derivation tree), so that for a TAG-related formalism the requirement for incremental generation can be expressed as the requirement that the order of constituents be determined independently of the choice of an elementary tree or tree set. Both FO-TAG and MC-TAG meet this criterion.

3.3.6 Reape's Word Order Domains

The FO-TAG system bears an interesting relation to a theory of word order variation proposed by Reape (1990). Reape's system is embedded in a unification-based framework, but we will briefly summarize it in general terms. Reape assumes that word order is not expressed by a surface-syntactic structure: syntactic constituency, but not word order, is represented by a tree. Word order is derived from the constituency tree compositionally by determining order in so-called *word*

order domains, which roughly correspond to the elements subsumed by phrasal domains. The elements within a domain are ordered by LP rules. In recursive settings, an element of a domain may itself be a domain. Crucially, Reape introduces the operation of *domain union*, which merges a daughter domain into its mother domain. This formal operation is inspired by the linguistic "clause union" analysis of Evers (1975). LP relations that hold between elements in the daughter domain are preserved after domain union. We see that the notion of domain union resembles substitution or adjunction in an FO-TAG: in both cases, elements from the embedded structure are freely positioned among elements of the embedding structure, as long as LP rules locally valid in each structure are valid in the merged structure. Absence of domain union, the default in Reape's system, corresponds to presence of the integrity constraint in FO-TAG. Thus, Reape's analysis of German is in some ways quite similar to that presented in this section.

However, there is an important difference. In Reape's system, it is possible to state LP rules that relate elements of (originally) different domains. Such LP rules are stated as LP schemas of the form [DOM NP $\prec$ V], meaning that in a domain, all NPs must precede all verbs, whether or not the NP and verb originate in the same word order domain. Recall that in FO-TAG, only nodes originating in the same structure can be related by LP rules. In this way, Reape can also order the verbs in the Dutch order using LP rules, while FO-TAG achieves this task by using the power of tree adjunction. We conjecture that FO-TAG can be simulated using word order domains, while the inverse does not hold.

How does Reape handle the case which we suggested is difficult for FO-TAG, such as sentence (xxiv) ((7) above)? Reape avoids the problem by assuming that extraposed clauses do not form a domain union. Therefore, their elements must all follow their governing verb, and entanglements of the kind shown in Figure 3.11 are avoided. In order to derive instances of the Third Construction, Reape admits an operation of "raising", which lifts a member of a domain into the domain of the governing verb.[10] This is exactly the kind of "permeable" integrity constraint that we suggested might save FO-TAG: rather than encode word order variation purely by LP rules, it allows a structural notion (moving up but not down) to be used in the definition of allowable variations.

Reape embeds word order domains in the unification-based framework of HPSG. In such a formally powerful framework, all necessary operations can be encoded adequately and elegantly. It is not clear how FO-TAG can be extended with a raise-type operation without destroying the essential character of the formalism.

3.4 Relaxing Immediate Dominance: Multi-component TAG

3.4.1 Types of Multi-component TAG

Weir (1988) proposes several multi-component systems as extensions to simple TAGs. These systems have in common that a grammar consists of a set of sets of trees. A set can contain either a single initial tree, or one or more auxiliary trees. During a derivation, all trees from an auxiliary set must be adjoined simultaneously. The different multi-component systems differ in

[10] Reape actually motivates this operation by an even more complex problem involving partial VP topicalization, which however is related to the one we have discussed in this section.

terms of the restrictions that they place on the loci of adjunction of the members of one set:

- In **tree-local MC-TAG**[11], all trees from one set must be adjoined into a single elementary tree. It can easily be shown that this system is equivalent, both weakly and strongly (though not derivationally), to simple TAG.
- In **set-local MC-TAG**, all trees from one set must be adjoined into a single elementary tree set, though not necessarily into the same tree. This system is weakly equivalent to linear context-free rewriting systems (LCFRS; Weir (1988)) and multiple context-free grammars (MCFG; (Seki et al., 1991)). LCFRS have been studied in detail; they are known to generate only semi-linear languages and to be polynomially parsable.
- In **non-local MC-TAG**, trees from a tree set are adjoined into the derived tree. There is no restriction on the locus of adjunction for each individual tree. While non-local MC-TAGs have not been studied much, it is easy to show, using other results from the literature, that non-local MC-TAGs generate non-semi-linear languages and that the word recognition problem for non-local MC-TAGs is NP-complete (Rambow and Satta, 1992). It is known that non-local MC-TAG can generate languages that set-local MC-TAG cannot; Rambow and Satta (1992) conjecture that the inverse holds as well.

To all three of these systems, we can add an additional constraint system called "dominance links", giving rise to MC-TAG-DL. A dominance link (an ordered pair) may be specified between any two nodes of different trees in the same tree set. In the derived tree, the first node must dominate (not necessarily immediately) the other. Like locality, dominance links restrict derivations, but the two types of restrictions are orthogonal. While any linguistic application of any of the MC-TAG systems will use dominance links, they have not been studied formally; they appear not to decrease the weak generative power.

Which, if any, of these systems can handle scrambling? While tree-local MC-TAG (and tree-local MC-TAG-DL) is weakly and strongly equivalent to simple TAG, it does not follow that its derivational generative power is that of simple TAG. In fact, it is greater: clearly, our counter-example for TAG (sentence (4), page 42) can be derived by a tree-local MC-TAG-DL, as can all structures involving only two clauses. In the following section, we will show that the two types of local MC-TAGs cannot, however, derive the full range of scrambled German sentences.

3.4.2 Local Systems Cannot Derive Scrambling

In this section, we show that local systems cannot derive scrambling.[12]

We start out by discussing the class of linear context-free rewriting systems (LCFRS) (Vijay-Shanker et al., 1987; Weir, 1988), which represent a generalization of the locality constraint embodied by tree-local MC-TAG. LCFRS is not a formalism, but a class of formalisms which share the property that all derivations can be characterized by a CFG. The definition of an

[11] The terminology used here has been proposed by David Weir.

[12] This section is based on joint work with Tilman Becker and Michael Niv. See (Becker et al., 1992).

LCFRS comprises two parts: First, a generalized context-free grammar (GCFG) constructs a set of (context-free) derivation trees. Each context-free production rule is associated with a function symbol. Second, a unique yield function is associated with each production rule of the GCFG. The yield of a node is a tuple of strings. The yield function constructs the yield of the node from the yields of the nodes of the right-hand side along with a bounded collection of new terminals. LCFRS includes tree-local MC-TAGs, a tree-rewriting system (Weir, 1988); context-free hypergraph grammars, a hypergraph rewriting system defined by Engelfriet and Heyker (1991) and whose inclusion in LCFRS is shown in (Weir, 1992); and multiple context-free grammars (MCFG), defined by (Seki et al., 1991). MCFG, which was developed contemporaneously with and independently of LCFRS, is a notational variant of the "canonical" string-rewriting version of LCFRS, which is often used in proofs regarding LCFRS. We will therefore freely use results about MCFG when discussing LCFRS.

We now formally define an LCFRS.

Definition 3 *An LCFRS G is an ordered pair (C, E), where C is a GCFG and E a set of equations defining yield functions. The GCFG is a quadruple, $C = (V, S, F, P)$, where V is a set of variables (nonterminals), S a distinguished variable, P a set of productions of the form $A \rightarrow f(A_1, \ldots, A_n)$, and F a set of such function symbols. If k is the maximum number of components in the yield of any nonterminal of G (its "fanout"), we will say that G is an LCFRS(k) grammar.*

We now define a formal representation for a subset of scrambled German sentences as a set of indexed strings. We will assume that each verb has exactly one overt nominal argument. Recall that the indices are not part of the terminal alphabet, which in this case is simply $\{n, v\}$.

$$\mathsf{SCR}^{ind} = \{\sigma(n^{[0]}, \ldots, n^{[m]})v^{[0]} \ldots v^{[m]} | m \geq 0 \text{ and } \sigma \text{ a permutation}\}$$

The unindexed string language SCR is the language $\{n^m v^m | m > 0\}$. Each index in SCR pairs exactly one n and one v, reflecting the fact that the n is an argument of the v. This means that the elementary structures of the grammar should be clause-sized, i.e., contain a verb and (a position for) its argument, and nothing else.

Before we can prove anything about the derivational generative capacity of LCFRS, we need to explain what is meant by a single derivation step in MC-TAG and LCFRS.

- In an Multi Component-TAG (Weir, 1988), all symbols bearing the same index (and no others) must be generated by multi-component adjunction of the same elementary tree set.
- In an LCFRS, G derives the indexed string w^{ind} if symbols bearing the same index (and no others) are contributed during the application of the same production. Put differently, the yield function associated with a production rule p is a composition of the strings contributed by the symbols of the right-hand side of p and symbols of w^{ind} that bear the same index. Furthermore, each application of a yield function introduces different indices.

We can now state and prove our main result about local systems.

Theorem 1 *No LCFRS can derive* $\mathsf{SCR}^{\mathrm{ind}}$.

Proof. We will show by contradiction that there can be no LCFRS that can derive SCR^{ind}. The idea of the proof is as follows: We assume that there is some LCFRS(k) grammar G that derives SCR^{ind}. Then we construct a new grammar G' that introduces k' subscripts into the terminal alphabet of G'. This allows us to interpret the derivational generative capacity in terms of weak generative capacity and from this point on, we only deal with q(unindexed) string languages. From G' we derive by intersecting with a regular language and by applying a homomorphism an LCFRS(k) grammar G''' that derives COUNT-k' $= \{a_1^n, \ldots, a_{k'}^n | n > 0\}$. Since we choose k' to be sufficiently larger than k, it is known that COUNT-k' cannot be derived by any LCFRS(k) grammar. Thus we have a contradiction.

In detail:

1. Suppose there is an LCFRS(k) grammar $G = (C, E)$ with $C = (V, S, F, P)$ that derives SCR^{ind}. The alphabet T of G is $\{n, v\}$. Let $k' = 2k + 1$. We construct a new LCFRS(k) grammar $G' = (C', E')$ over the alphabet $T' = \{n_0, \ldots, n_{k'-1}, v_0, \ldots, v_{k'-1}\}$. $C' = (V, S, F', P')$, where $F' = \{f_i^{(j_1,\ldots,j_l)} | f \in F \text{ and } 0 \leq i, j_1, \ldots, j_l < k'\}$ and $P' = \{A^j \longrightarrow f_i^{(j_1,\ldots,j_l)}(A_1^{j_1}, \ldots, A_l^{j_l}) | (A \longrightarrow f(A_1, \ldots, A_l)) \in P \text{ and } 0 \leq i, j, j_1, \ldots, j_l < k'\}$. For every equation $e \in E$ associated with $f \in F$, there are k'^{l+1} equations $e_0^{(0,\ldots,0)}, \ldots, e_{k'-1}^{(k'-1,\ldots,k'-1)}$ in E', associated with $f_i^{(j_1,\ldots,j_l)}$, respectively, where $e_i^{(j_1,\ldots,j_l)}$ is the same as e (for $0 \leq i, j_1, \ldots, j_l < k'$), except that all non-terminal symbols introduced bear the subscript i. We have:

$$L(G') = \{\sigma(n_{i_0}, \ldots, n_{i_m}) v_{i_0} \cdots v_{i_m} | \quad m \geq 0, i_0, \ldots, i_m \in \{0, \ldots, k' - 1\}, \text{ and } \sigma \text{ a permutation}\}$$

It is clear that in every string in $L(G')$, for any i, $0 \leq i \leq k' - 1$, the number of instances of n_i is equal to the number of instances of v_i.

3. Let LCFRL(k) be the class of languages derived by LCFRS(k) grammars. Since LCFRL(k) is closed under intersection with regular languages (Seki et al., 1991), there is an LCFRS(k) grammar G'' that derives the language that results from the intersection of $L(G')$ with the regular language R:

$$R = n_0^* \ldots n_{k'-1}^* (v_0 \ldots v_{k'-1})^*$$

We obtain:

$$L(G'') = \{n_0^j \cdots n_{k'-1}^j (v_0 \cdots v_{k'-1})^j | j > 0\}$$

4. Since the LCFRL(k) are also closed under application of homomorphisms (Seki et al., 1991), there is an LCFRS(k) grammar G''' that derives the language that results from the application of the following homomorphism to L(G'') (ϵ denotes the empty string):

$$h(a) = \begin{cases} \epsilon & \text{if } a = v_i, 0 \leq i < k' \\ n_i & \text{if } a = n_i, 0 \leq i < k' \end{cases}$$

The resulting language is

$$L(G''') = \{n_0^j \cdots n_{k'-1}^j | j > 0\}$$

which is COUNT-k'. The pumping lemma for LCFRS (Seki et al., 1991) can then be used to show that COUNT-k' is not in LCFRL(k). So we have derived a contradiction.

Therefore no LCFRS can derive the Scrambling Language SCR^{ind}. ■

Since both tree-local and set-local MC-TAGs are LCFRSs, we conclude that neither is suited for the representation of German syntax, and we are currently left with only one system, namely non-local MC-TAG. Unfortunately, as we have mentioned previously, this system is not mildly context-sensitive: it can generate non-semi-linear and NP-complete languages. But as we will argue in the next section, non-local MC-TAG is not an adequate formalism, either.

3.4.3 Topicalization in German

German is a verb-final language and exhibits scrambling. But as we have seen in Section 2.3, in addition it is verb-second (V2), which means that in a root clause, the finite verb (main verb or auxiliary) moves into the second position in the clause (standardly assumed to be the COMP position). Non-local MC-TAG-DL would provide an adequate analysis for languages like Korean and Japanese (Lee, 1991; Rambow and Lee, 1994), but in German the approach proves problematic. In Korean, there is little evidence that there is a topicalization movement into a functional projection that is distinct from scrambling. German, however, is a V-2 language, and therefore topicalization is obligatory in every declarative root clause.

As mentioned above, topicalization, as a classical instance of *wh*-movement, exhibits the full range of island effects; thus, it would be appealing to express long-distance topicalization by the means proposed in (Kroch, 1987) and refined in (Kroch, 1989; Frank, 1992), namely the adjunction of a matrix clause into the subordinate clause, one of whose elements has been moved. Under this analysis, the island constraints follow from independently motivated constraints on the shape of the elementary trees; successive cyclic movement is modeled by successive adjunction of clausal trees. Subjacency need no longer be stated as a separate condition, it is a corollary of the particular analysis. However, this analysis is unavailable in a system in which clausal subcategorization is handled by substitution.

Instead, let us follow the approach laid out in (Kroch, 1989), and modify it to account for scrambling as well. For verbs that subcategorize for clausal elements we use auxiliary trees instead of initial trees, with the clausal subcategorization indicated by the footnode. With this approach, we run into a perplexing problem: we predict that scrambling into matrix clauses from embedded clauses is impossible. But this prediction is false; in fact, we can simultaneously (long-distance) scramble and (long-distance) topicalize NPs out of a subordinate clause, as we have seen in sentence (35), page 24. The problem is as follows: if we want to implement long-distance topicalization by adjoining the matrix clause into the embedded clause, then we cannot subsequently adjoin the long-distance scrambled argument of the lower clause into the matrix clause, for this will violate the definition of any of the multi-component systems that have been proposed to date,

be it tree-local, set-local, or non-local. These formalisms require *simultaneous* adjunction of all trees of a set (Weir, 1988, p.32), but in order to scramble the embedded argument into the matrix clause, we must first adjoin the matrix clause, and only then can we adjoin the embedded NP auxiliary tree into the matrix clause. Note that the option of set-internal adjunction introduced by (Lee, 1991) does not solve the problem, since we wish to adjoin the embedded argument into a tree from a *different* set. We conclude that if we want to maintain the "traditional" analysis of topicalization, and at the same time implement scrambling by multi-component adjunction, then we need to redefine the formalism by relaxing the simultaneity requirement.[13] We do not give examples here, but refer to Section 4.6.

3.5 Conclusion

In this chapter, we have analyzed the data given in the previous chapter and have discussed in broad strokes the kind of formal system that we would like to have as the basis for a formal representation of German syntax. We have arrived at the following conclusions:

- We need a system that relaxes dominance links rather than linear precedence, i.e., a version of MC-TAG-DL rather than FO-TAG.
- We need to allow for non-local derivations in order to derive the full range of scrambled sentences.
- We need to allow for non-simultaneous adjunction in order to derive sentences with both long-distance topicalization and long-distance scrambling

The only currently defined system that meets the first two criteria is non-local MC-TAG-DL, which does not have the formal and computational properties we desire. No currently defined system meets all three criteria. In the following chapter, we will take a closer formal look at a variety of systems, and define V-TAG, which exactly meets all three criteria above. As we will see, it also meets the formal and computational desiderata outlined in Section 3.1.

[13] Alternatively, we could keep the notion of simultaneous adjunction, and allow adjunction from one set into different elementary sets *and/or* the derived tree. That definition we conjecture to be equivalent.

Chapter 4

Formal Issues

In this chapter, we develop formal systems for the natural language phenomena that we are interested in. We start out by summarizing the notation we will use in Section 4.1. In Section 4.2, we provide an overview over so-called regulated rewriting systems, some of which have been previously defined, and some of which we define in the remainder of this chapter. Section 4.3 defines multiset-valued linear index grammar ({}-LIG) and discusses some of its formal properties. This formalism will not be used for linguistic purposes in this thesis, but it is useful for some of the proofs we present in this chapter. Unordered vector grammars with dominance links are defined in Section 4.4, and are shown to be equivalent to {}-LIG. In Section 4.5, we review the definition of a formal automaton for tree adjoining grammar, which we will need for subsequent exposition. Section 4.6 introduces V-TAG, the tree rewriting version of UVG-DL. Finally, in Section 4.7, we discuss the relation of the formalisms introduced in this chapter to several other systems, including categorial systems, "quasi-trees", and unification-based formalisms.

4.1 Notational Conventions

This section presents a summary of standard notation used in the remainder of this chapter. Notation that is particular to this chapter will be introduced when appropriate; all conventions discussed here are standard notational conventions used widely in the literature.

A set of symbols V is called an *alphabet*. A *string* or *word* is a sequence of symbols. V^* is the set of all strings over alphabet V. If a is a string, $|a|$ denotes its length. ε denotes the empty string. The concatenation operation will be denoted by juxtaposition: if w_1 and w_2 are strings, then $w_1 w_2$ is the string obtained by concatenating w_1 and w_2. Concatenation of a sequence of strings is denoted by three dots in the center of the line: $w_1 \cdots w_n$ is the concatenation of the n strings $w_1, \ldots, w_n$. If w is a string, w^k denotes the string obtained by concatenating k copies of w; if $k = 0$, $a^k = \varepsilon$. We denote by $\mathsf{Perm}(V^*)$ the set of all words obtained by any permutation of the symbols in some word in V^*. If T is a subset of an alphabet V, and w a string over V, then $\#_T(w)$ denotes the number of occurrences of members of T in w.

We will be using the following conventions for choosing variables:

- Lower case letters at front of alphabet: terminal or input (automaton) symbols.
- Lower case letters towards front of alphabet (f and onward): index symbols.
- Lower case letters towards end of alphabet (s and onward): strings of index symbols.
- Lower case letters at end of alphabet (v and onward): strings of terminal or input (automaton) symbols.
- Capital letters at front of alphabet: nonterminal symbols.
- Capital letters at back of alphabet: stack symbols (Z, Y, ...) or variables for symbols that can be either terminal or nonterminal symbols (X, Y).
- Lower-case Greek letters at front of alphabet: sentential forms, i.e. strings of terminal and non-terminal (and perhaps index) symbols.
- Lower-case Greek letters towards front of alphabet (γ and onward): stack configurations, i.e., strings of stack (and perhaps index) symbols.
- Capital Greek letters: sets of symbols.

In this chapter, we will be introducing various formal systems. The designation for a type of formal systems always also denotes the set of all such formal systems. If $\mathcal{F}$ is a formal system, we define $\mathcal{L}(\mathcal{F}) = \{L(G) \mid G \in \mathcal{F}\}$. For example, UVG is a formalism, but UVG also denotes the class of all grammars in that formalism, and the class of all languages derived by grammars in UVG is denoted by $\mathcal{L}(\text{UVG})$. If $\mathcal{F}$ is a string rewriting system, we will designate the class of rewriting systems without ε-productions by $\mathcal{F}$–ε.

A *rewrite rule* or *production* is an ordered pair. For a production $p \in P$, $\mathsf{rhs}(p)$ denotes the right-hand side of p, and $\mathsf{lhs}(p)$ denotes the left-hand side of p.

For string rewriting systems, we will use $\Longrightarrow_G$ to denote the derivation relation for grammar G. When clear from the context, we will omit the subscript G and denote the rewrite relation simply as $\Longrightarrow$. As usual, we represent the reflexive and transitive closure of $\Longrightarrow_G$ by $\overset{*}{\Rightarrow}_G$. We also write $\gamma \overset{\pi}{\Rightarrow}_G \delta$ if $\pi \in P^*$ is a sequence of productions that rewrites γ into δ, and we write $\overset{n}{\Rightarrow}_G$ for a natural number n to indicate that δ can be derived from γ in a sequence of n rewrite steps.

Similarly, for automata, we will use $\vdash_M$ to denote the move relation for automaton M. When clear from the context, we will omit the subscript M and denote the move relation simply as $\vdash$. As usual, we represent the reflexive and transitive closure of $\vdash_M$ by $\vdash_M^*$.

For a grammar G (in some string rewriting system $\mathcal{F}$) with terminal alphabet V_T, start symbol S, and derive relation $\Longrightarrow_G$, the *language generated by* G or $L(G)$ will usually be the set $\{w \mid S \overset{*}{\Longrightarrow}_G w \text{ and } w \in V_\mathrm{T}^*\}$. (We will refer to this as "the usual definition".)

If v is a vector, $|v|$ denotes its length.

We will use the same notation for multisets as we use for sets. In particular, multisets will be enclosed by curly brackets ($\{\}$), and we will use the operations $\cup, \cap, \setminus$ and the symbol $\emptyset$. The meaning of these symbols (set or multiset) will always be clear from the context. If A is a set,

$\mathcal{M}(A)$ denotes the set of all multisets whose members are also members of A. Note that if A is non-empty, then $\mathcal{M}(A)$ is an infinite set.

$\mathbb{N} = \{0, 1, 2, \ldots\}$ is the set of non-negative integers.

We will number theorems, lemmas, corollaries, and conjectures consecutively. Definitions and examples will be numbered independently. Proofs will be closed by a filled box (■), examples by an empty box (□).

4.2 Multi-Component Rewriting Systems: a Classificatory Overview

In this section, we briefly describe several related formal systems. They have in common that the elementary structures in these systems consist of a set of rewrite rules. These can be either string rewrite rules or tree rewrite rules. A derivation consists in the application of a collection of such sets of rewrite rules. The systems differ in the restrictions which they place on the application of rewrite rules during a derivation. We will refer to a formal system in which the elementary structures are sets of rewrite rules as *multi-component rewriting systems*. These are a subclass of what has been called *regulated rewriting systems* (Dassow and Păun, 1989), in which context-free derivations are restricted in some manner in order to increase the generative power of the system beyond that of CFG.

4.2.1 Definitional Parameters for Multi-Component Rewriting Systems

In defining the derivations in multi-component rewriting systems, there are several parameters that can be varied.

1. **Type of rewriting.** The sets can consist of context-free rules that rewrite sentential forms (string rewriting), or of trees that are adjoined or substituted (tree rewriting).
2. **Locality of derivation.** Members of a set can be required to rewrite members of a single, other set. Equivalently, we can require the derivation structure to be a tree. This is the locality constraint introduced by Weir (1988); it is a constraint on derivations.
3. **Dominance links.** Dominance links can be stated between members of an elementary set. These act as constraints on derivations: they must hold when the derivation has terminated. In the case of string rewriting, this means that the dominance relation expressed by the link must hold on the derivation tree; in the case of tree rewriting systems, this means that the dominance relation expressed by the link must hold on the derived tree.
4. **Ordering of rules.** The application of members of a set of rewrite rules can be ordered. In the case of string rewriting, ordering is not independent of dominance: if a rewrite rule dominates another, then it must be applied first. However, dominance constraints need not link all members of a set, while the ordering must be total.
5. **Timing of application.** The application of rewrite rules from a single set can occur in one of three possible temporal sequences:

- All rules from a set must be applied *simultaneously*.
- All rules from a set must be applied *contiguously*, meaning one right after the other. No rule from a different set can intervene.
- The rules may be applied *freely*, meaning that rules from different sets can be interleaved.

Observe that linguistically, dominance links are used to enforce c-command relationships, such as between "moved" elements and their traces. It is not clear whether there is any linguistic application where dominance links are *not* needed; they therefore are a natural constraint to impose on formal systems.

We thus have a large number of potential systems, some of which have in fact been defined. We summarize some systems in the table in Figure 4.1. Their formal properties are summarized in Figure 4.2. In both tables, systems that are not separated by horizontal lines are weakly equivalent; a single horizontal line separates string rewriting systems from the corresponding tree rewriting system. In table Figure 4.2, entries in parentheses indicate conjectures. We will subsequently comment briefly on these systems; for the non-local string rewriting systems, Dassow and Păun (1989) provide an invaluable and detailed overview, which this summary has drawn on heavily.

System	defined	type	locality	dom	ord	time
lMC-TAG	Weir 1988	trees	local	no	no	sim
LUSCG	R&S 1993	strings	local	no	no	sim
USCG	M&R 1971	strings	non-local	no	no	sim
VG	C&M 1973	strings	non-local	no	yes	free
MG	Abraham 1965	strings	non-local	no	yes	cont
UMG	C&M 1973	strings	non-local	no	no	cont
nlMC-TAG	Weir 1988	trees	non-local	no	no	sim
nlMC-TAG-DL	BJR 1991	trees	non-local	yes	no	sim
UVG	C&M 1973	strings	non-local	no	no	free
VMC-TAG	Section 4.6.1	trees	non-local	no	no	free
UVG-DL	Section 4.4.1	strings	non-local	yes	no	free
V-TAG	Section 4.6.1	trees	non-local	yes	no	free

Figure 4.1: Table of multi-component rewriting systems (definitions)

4.2.2 Local MC-TAG and LUCSG

Local multi-component TAG (lMC-TAG) is defined by Weir (1988, p.31ff), where it is called "Type 2 MC-TAG". Weir (1988) shows that $\mathcal{L}$(lMC-TAG) = $\mathcal{L}$(LCFRS), where LCFRS is a generalization of CFGs (see Section 3.4.2, page 53, for a definition of LCFRS), and that all languages

System	type	locality	recog	semi-lin	in CSL
lMC-TAG	trees	local	poly	yes	yes
LUSCG	strings	local	poly	yes	yes
MG	strings	non-local	NP-compl	no	?
MG–ε	strings	non-local	NP-compl	no	yes
nlMC-TAG	trees	non-local	NP-compl	no	?
nlMC-TAG-DL	trees	non-local	(NP-compl)	?	?
UVG	strings	non-local	poly	yes	yes
VMC-TAG	trees	non-local	(poly)	(yes)	(yes)
UVG-DL	strings	non-local	?	?	?
UVG-DL$_{\text{Lex}}$	strings	non-local	poly	(yes)	yes
V-TAG	trees	non-local	?	?	?
V-TAG-Δ_{Lex}	trees	non-local	poly	(yes)	yes

Figure 4.2: Table of multi-component rewriting systems (properties)

in $\mathcal{L}$(lMC-TAG) are semilinear and polynomially parsable. Seki et al. (1991) independently define multiple context-free grammar (MCFG), which is the string-rewriting version of LCFRS. They also show polynomial parsability and semi-linearity, in addition to inclusion in $\mathcal{L}$(CSG). However, MCFG is not a true string-rewriting system, since a MCFG first generates a derivation tree, from which the derived string is then computed "bottom-up".

Local unordered scattered context grammar (LUSCG) is defined by Rambow and Satta (1994b) as the local counterpart of USCG (see Section 4.2.3) and is there shown to be equivalent to MCFG. Unlike MCFG, LUCSG is a "standard" top-down string rewriting system. In LUSCG, locality of derivation is enforced by equivalence relations over non-terminals in the sentential forms of a derivation, similar to the system used in SDTS (Aho and Ullman, 1972). Interestingly, in local systems, the string vs. tree rewriting parameter has no formal effects.

4.2.3 USCG, VG, MG, UMG, and non-local MC-TAG

In an Unordered Scattered Context Grammar (USCG; originally defined by Milgram and Rosenfeld (1971)), unordered sets of context-free rules are applied simultaneously to a sentential form. There is no notion of locality in derivation. USCG is weakly equivalent to several other well-studied formalisms, in particular Matrix Grammar (MG, defined by Abraham (1965) as an attempt to capture the linguistic expressiveness of transformations in a restricted formal system), unordered Matrix Grammar (UMG), and Vector Grammar (VG) (UMG and VG are both defined by Cremers and Mayer (1973)). In MG, as in USCG, we have sets of context-free productions, but they must be applied contiguously (one right after the other), not simultaneously. In UMG, the sequences are not ordered, while in MG, they are. A VG is defined like an MG, except that the context-free productions in a set need not be applied contiguously during a derivation. Instead, productions from different sets can be interleaved, but productions from one set must be applied

in a given order, and during the derivation, all productions from an instance of a set (from which at least one production has been used) must be used.

From previous results, we know that these four formalisms are weakly equivalent[1], that the ε-free versions of these systems are strictly contained in $\mathcal{L}$(CSG) (Dassow and Păun, 1989, p.57),[2] that they generate non-semi linear languages (Dassow and Păun, 1989, p.26), and that they generate languages that are NP-complete (Dahlhaus and Warmuth, 1986). These formalisms are therefore not mildly context-sensitive.

Non-local multi-component TAG (nlMC-TAG) is also defined by Weir (1988, p.31ff), where it is called "Type 3 MC-TAG". It can easily be shown that $\mathcal{L}$(USCG) $\subseteq$ $\mathcal{L}$(nlMC-TAG); the relationship is exactly the same as that between CFGs and TAGs. The (negative) results about recognition complexity and semi-linearity therefore transfer immediately, and it is highly likely that we also have $\mathcal{L}$(nlMC-TAG) $\subset$ $\mathcal{L}$(CSG), since the move from string to tree rewriting can easily be simulated by a CSG.

4.2.4 nlMC-TAG-DL

Non-local multi-component TAGs with dominance links were first described by Becker et al. (1991), where they were used to handle German scrambling facts. Dominance links can be specified between nodes in trees from the same set. They must hold when the derivation is completed.

4.2.5 UVG

Unordered vector grammars (UVG) were first defined in (Cremers and Mayer, 1973). In a UVG, context-free productions are grouped into sets. If one production from a set is used in a derivation, then all other members of that set must also be used during the derivation, but not necessarily immediately. Since UVG is closely related to several other formalisms we define in this thesis, we give a full formal definition and an example, and discuss its formal properties in some detail.

Definition 4 *An* **unordered vector grammar** *(*UVG*) is a quadruple $G = (V_{\mathrm{N}}, V_{\mathrm{T}}, V, S)$ where V_{N} and V_{T} are finite, disjoint sets of nonterminal and terminal symbols respectively, $S \in V_{\mathrm{N}}$ is the start symbol and V is a finite set of vectors having the form $(A_1 \rightarrow \alpha_1, \ldots, A_n \rightarrow \alpha_n)$, $n \geq 1$, where $A_i \in V_{\mathrm{N}}$ and $\alpha_i \in (V_{\mathrm{N}} \cup V_{\mathrm{T}})^*$ for $1 \leq i \leq n$.*

Let P be the set of all productions participating in some vector of G. We associate with G a rewrite relation, written $\overset{*}{\Longrightarrow}_G$, defined on $(V_{\mathrm{N}} \cup V_{\mathrm{T}})^*$ in such a way that $\gamma A \delta \Longrightarrow_G \gamma \alpha \delta$ holds if and only if $A \rightarrow \alpha$ is in P.

The language derived by G, written $L(G)$, is the set of all strings $w \in V_{\mathrm{T}}^*$ such that $S \overset{\pi}{\Rightarrow}_G w$ for some $\pi \in \mathsf{Perm}(V^*)$.

[1] Another weakly equivalent system is State Grammar (SG). In SGs, a finite-state control regulates the rewriting of a sentential form with a single context-free production at a time.

[2] In the case of VG, it is not known whether VG–ε(the ε-free version of VG) is weakly equivalent to the ε-free versions of the other systems, or whether the inclusion of $\mathcal{L}$(VG–ε) in $\mathcal{L}$(CSG) is proper.

Note that a sequence $\pi \in P^*$ belongs to $\mathsf{Perm}(V^*)$ just in case every production p from some vector $v \in V$ is found in π as many times as any other production in v.

If $G = (V_\mathrm{N}, V_\mathrm{T}, \{v_1, \ldots, v_n\}, S)$ is a UVG, then the *underlying context-free grammar* is the CFG $G' = (V_\mathrm{N}, V_\mathrm{T}, \cup_{i=1}^{n} v_i, S)$. The *context-free derivation tree* (or simply *derivation tree*) for a derivation ϱ in a UVG G is just the derivation tree we obtain if we consider ϱ to be a derivation in the underlying context-free grammar, except that the nonterminal symbols that label the nodes are annotated with the name of the production used to rewrite them. (Labels of nodes on the frontier, whether terminals or nonterminals, are not annotated.) The annotation is necessary since there can be different instances of the same production in different vectors. We will often talk about a derivation as if it were a derivation tree; this is legitimate since a unique derivation tree corresponds to each derivation (though the converse does not necessarily hold). A sample derivation tree is shown in Figure 4.3.

We illustrate UVG by giving a sample grammar that derives German embedded subordinate clauses with an overt complementizer, and which allows long-distance scrambling. (Note that the grammar is given only to illustrate the formalism; it is not supposed to represent a serious linguistic analysis.)

EXAMPLE 1

$G = (V_\mathrm{N}, V_\mathrm{T}, V, \mathrm{S}')$ with

$$
\begin{array}{lcl}
V_\mathrm{N} & = & \{\mathrm{S}', \mathrm{VP}, \mathrm{NP}_\mathrm{nom}, \mathrm{NP}_\mathrm{acc}\} \\
V_\mathrm{T} & = & \{\textit{daß}, \textit{verspricht}, \textit{zu versuchen}, \textit{zu verschrotten}, \textit{der Meister}, \textit{den Kühlschrank}\}^3 \\
V & = & \{v_1, v_2, v_3, v_4, v_5, v_6, v_7\}
\end{array}
$$

where

$$
\begin{array}{ll}
v_1\colon \{ & (\mathrm{S}' \rightarrow \textit{daß}\ \mathrm{VP}) \ \} \\
v_2\colon \{ & (\mathrm{VP} \rightarrow \mathrm{NP}_\mathrm{nom}\ \mathrm{VP}), (\mathrm{VP} \rightarrow \mathrm{VP}\ \textit{verspricht}) \ \} \\
v_3\colon \{ & (\mathrm{VP} \rightarrow \mathrm{VP}\ \textit{zu versuchen}) \ \} \\
v_4\colon \{ & (\mathrm{VP} \rightarrow \textit{zu versuchen}\ \mathrm{VP}) \ \} \\
v_5\colon \{ & (\mathrm{VP} \rightarrow \mathrm{NP}_\mathrm{acc}\ \mathrm{VP}), (\mathrm{VP} \rightarrow \textit{zu verschrotten}) \ \} \\
v_6\colon \{ & (\mathrm{NP}_\mathrm{nom} \rightarrow \textit{der Meister}) \ \} \\
v_7\colon \{ & (\mathrm{NP}_\mathrm{acc} \rightarrow \textit{den Kühlschrank}) \ \}
\end{array}
$$

Recursion on vector v_3 allows for arbitrary depths of embedding. Vector v_4 is the extraposed counterpart of vector v_3. Here is a sample derivation. For clarity, we have annotated the sentential forms with multisets of productions that still need to be applied. (p_{ij} refers to the j-th production from vector v_i.) Clearly, a derivation terminates successfully only if this set is empty.

[3] Glosses (in order): that, promises, to try, to scrap, the master, the refrigerator.

$$
\begin{aligned}
\mathrm{S}'\ \{\} &\overset{p_{11}}{\Longrightarrow} && \textit{daß}\ \mathrm{VP},\ \{\} \\
&\overset{p_{51}}{\Longrightarrow} && \textit{daß}\ \mathrm{NP}_{\mathrm{acc}}\ \mathrm{VP},\ \{p_{52}\} \\
&\overset{p_{71}}{\Longrightarrow} && \textit{daß den Kühlschrank}\ \mathrm{VP},\ \{p_{52}\} \\
&\overset{p_{21}}{\Longrightarrow} && \textit{daß den Kühlschrank}\ \mathrm{NP}_{\mathrm{nom}}\ \mathrm{VP},\ \{p_{22}, p_{52}\} \\
&\overset{p_{61}p_{22}}{\Longrightarrow} && \textit{daß den Kühlschrank der Meister}\ \mathrm{VP}\ \textit{verspricht},\ \{p_{52}\} \\
&\overset{p_{31}}{\Longrightarrow} && \textit{daß den Kühlschrank der Meister}\ \mathrm{VP}\ \textit{verspricht},\ \{p_{52}\} \\
&\overset{p_{52}}{\Longrightarrow} && \textit{daß den Kühlschrank der Meister zu verschrotten zu versuchen verspricht},\ \{\}.
\end{aligned}
$$

□

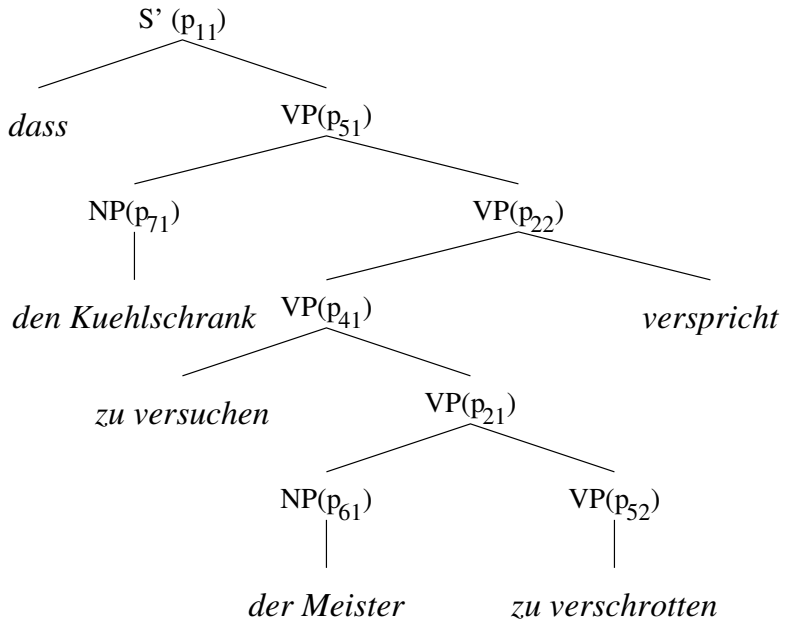


Figure 4.3: Derivation tree for an ungrammatical sentence

While this grammar allows for unbounded scrambling, it does not adequately rule out illicit word order variations, in particular movement of NPs to the right (or, in the phrase-structure tree, downward). For example, G_1 derives the sentence *daß den Kühlschrank zu versuchen der Meister zu verschrotten verspricht*, which is completely out. A derivation tree for this string is shown in Figure 4.3. The problem is that, whichever analysis we choose, there is always at least one argument (nominal or clausal) that no longer c-commands its governing verb. In Section 4.4 we will see how the addition of dominance links can remedy this problem.

Languages generated by UVG are known to be context-sensitive and semi-linear (Cremers and Mayer, 1974). Sudborough (1977) shows that the subclass of ε- and chain-production-free UVG ("strict UVG"; see Definition 17, page 89) generates languages that are contained in LOG(CFL), a complexity class that is included in the class of all languages that can be recognized in polynomial deterministic time. Satta (1993) shows that all UVGs are parsable in polynomial deterministic time.[4] Thus we can conclude that UVG is mildly context-sensitive. However

[4]See Rambow and Satta (1994a) for a sketch of this proof, and for a broader discussion of the use of UVG for the description of natural language syntax.

4.2.6 VMC-TAG

VMC-TAG, defined in Definition 20, page 102, is the tree-rewriting version of UVG. We conjecture that the relevant properties of UVG transfer to VMC-TAG, since the difference between string rewriting and tree rewriting does not affect polynomial parsability, semilinearity, or inclusion in CSL.

4.2.7 UVG-DL and V-TAG

Unordered vector grammar with dominance links (UVG-DL) is defined like UVG, except that dominance links can be specified within vectors. It is defined in Definition 14, page 85, and discussed in Section 4.4. Vector MC-TAG with dominance links (VMC-TAG-DL or V-TAG for short) is defined in Definition 21, page 103 and discussed at length in Section 4.6. V-TAG is the tree rewriting version of UVG-DL.

4.3 Multiset-Valued Linear Index Grammars ({}-LIG)

In this section, we will define multiset-valued linear index grammar ({}-LIG) and explore some of the properties of {}-LIG. A {}-LIG is like a Linear Index Grammar (LIG) in that it is defined as a context free grammar with indices which control the derivation, and that these indices are not copied during a derivation (linearity). However, contrary to the case of the stack-valued LIG, any symbol in the index set may be accessed during any step of the derivation.

Many of the techniques used to derive formal properties for formalisms such as []-LIG, UVG, and LCFRS/MCFG are not usable in the case of {}-LIG. As we will see, the results obtained are conditional on certain additional assumptions, which will, in Section 4.4, turn out to be reasonable when an equivalent formalism is used for the description of natural language syntax.

This section is structured as follows. Section 4.3.1 gives a formal definition and also defines some variants. Section 4.3.2 presents some normal forms. Section 4.3.3 discusses the relation between {}-LIG and context-sensitive grammars (CSG), showing inclusion for certain key cases. In Section 4.3.4 closure properties are discussed. Section 4.3.5 presents the formal automaton, {}-PDA, that corresponds to {}-LIG. Section 4.3.6 discusses the recognition complexity for parsing {}-LIGs, with polynomial parsing shown for certain key cases.

4.3.1 Definitions

In index grammars (Aho, 1968), a CFG is altered in such a way that nonterminal symbols are equipped with stacks of index symbols. These stacks are copied from a node of the derivation tree to all nonterminal daughters. The top stack element controls the possible steps of the derivation. If, as suggested by Gazdar (1988), the copying is restricted in such a way that only one daughter

node gets a copy of the mother node's stack, then we have linear index grammars (LIG).[5] The variation we propose here, called multiset-valued LIG or {}-LIG, assigns a multiset of indices to nonterminal symbols, rather than a stack of indices. The system is still linear, since the members of this multiset are never copied. However, the whole set is not passed on to a single daughter; instead, its contents is distributed among the daughters. We also define a variant which allows us to state a restriction such that only certain daughter nodes can inherit index symbols.

We now give two definitions of {}-LIG which are notational variants of one another. The first defines a pure string-rewriting system, in which the multiset is encoded as a string of index symbols. The second definition defines the multiset as just that: a multiset. While the first definition is "purer" as it is defined in terms of a single data structure (the string), the second is much easier to grasp and to use in proofs.

Definition 5 *(String-Rewriting System) A* **multiset-valued Linear Index Grammar** *({}-LIG) is a 5-tuple $(V_{\mathrm{N}}, V_{\mathrm{T}}, V_{\mathrm{I}}, P, S)$, where V_{N}, V_{T}, and V_{I} are disjoint sets of terminals, nonterminals, and indices, respectively; $S \in V_{\mathrm{N}}$ is the start symbol; and P is a set of productions of the following form:*

$$p : As \longrightarrow v_0 B_1 s_1 v_1 \ldots v_{n-1} B_n s_n v_n$$

for some integer n, $A, B_1, \ldots, B_n \in V_{\mathrm{N}}$, $s, s_1, \ldots, s_n \in V_{\mathrm{I}}^*$, and $v_0, \ldots, v_n \in V_{\mathrm{T}}^*$. (If $n = 0$, there are no nonterminal symbols.)

The derivation relation $\Longrightarrow$ for a {}-LIG is defined as follows. Let $\beta, \gamma \in (V_{\mathrm{N}} V_{\mathrm{I}}^* \cup V_{\mathrm{T}})^*$, $t, t_1, \ldots, t_n \in V_{\mathrm{I}}^*$, and $p \in P$ of the form given above. Then we have

$$\beta A t \gamma \Longrightarrow \beta v_0 B_1 t_1 v_1 \ldots v_{n-1} B_n t_n v_n \gamma$$

where there are $t', t'' \in V_{\mathrm{I}}^*$ with $t' \in \mathsf{Perm}(t)$ such that $t' = st''$, and there are $t_1'', \ldots, t_n'' \in V_{\mathrm{I}}^*$ such that $t_1'' \cdots t_n'' \in \mathsf{Perm}(t'')$, and $t_i = s_i t_i''$, $1 \leq i \leq n$. $L(G)$ for a {}-LIG is defined as usual.

The following "multiset" definition differs from the preceding "string-rewriting" definition in that all indices are represented using multiset notation, not as strings. This allows us to avoid the use of permutations.

Definition 5′ *(Multiset)A* **multiset-valued Linear Index Grammar** *({}-LIG) is a 5-tuple $(V_{\mathrm{N}}, V_{\mathrm{T}}, V_{\mathrm{I}}, P, S)$, where V_{N}, V_{T}, and V_{I} are disjoint sets of terminals, non-terminals, and indices, respectively; $S \in V_{\mathrm{N}}$ is the start symbol; and P is a set of productions of the following form:*

$$p : As \longrightarrow v_0 B_1 s_1 v_1 \ldots v_{n-1} B_n s_n v_n$$

for some integer n, $A, B_1, \ldots, B_n \in V_{\mathrm{N}}$, $s, s_1, \ldots, s_n$ multisets of members of V_{I}, and $v_0, \ldots, v_n \in V_{\mathrm{T}}^*$.

[5] Note that a LIG is not an IG that is linear (i.e., whose productions have at most one nonterminal on the right-hand side), but rather, it is a context-free grammar with linear indices (i.e., the indices are never copied).

The derivation relation $\Longrightarrow$ *for a* $\{\}$*-LIG is defined as follows. Let* $\beta, \gamma \in (V_{\mathrm{N}}V_{\mathrm{I}}^* \cup V_{\mathrm{T}})^*$, $t, t_1, \ldots, t_n$ *multisets of members of* V_{I}*, and* $p \in P$ *of the form given above. Then we have*

$$\beta A t \gamma \Longrightarrow \beta v_0 B_1 t_1 v_1 \ldots v_{n-1} B_n t_n v_n \gamma$$

such that $t = \cup_{i=1}^{n}(t_i \setminus s_i) \cup s$*. L(G) for a* $\{\}$*-LIG is defined as usual.*

It is clear that the two definitions are notational variants. The string-rewriting definition has the advantage of being a pure string-rewriting system and not requiring the definition of additional data structures for the notion of "derivation". The multiset definition has the advantage of corresponding more directly to the intuition underlying $\{\}$-LIG. In proofs, we will usually use the multiset definition unless stated otherwise. Examples will always be presented in terms of the multiset definition.

In both definitions, we see that the indices on the left-hand nonterminal symbol of a rule correspond to indices that are removed from the set when the rule is applied, and that indices associated with nonterminal symbols on the right-hand side of the rule are added to the sets for those nonterminals. We will refer to the former as *removed indices* of the rule, and the latter as the *added indices* of the rule. Furthermore, in the derivation step such as the one given above we will say that the indices t are *distributed among nonterminals* $B_1, \ldots, B_n$ *in accordance with rule* p. In a sentential form, the indices immediately following a nonterminal will be referred to as its *associated indices*. We now give two examples. Recall that examples are given in terms of the multiset definition.

EXAMPLE 2

The following grammar derives the language (COUNT-3)*, where COUNT-3 = $\{a^n b^n c^n \mid n \in \mathbb{N}\}$.

Let $G_2 = (V_{\mathrm{N}}, V_{\mathrm{T}}, V_{\mathrm{I}}, P, S)$ with:

$$\begin{array}{lcl}
V_{\mathrm{N}} & = & \{S, T, A, B, C\} \\
V_{\mathrm{T}} & = & \{\mathrm{a,b,c}\} \\
V_{\mathrm{I}} & = & \{s_b, s_c\} \\
P & = & \{p_1 : S \longrightarrow TS \\
 & & \ p_2 : S \longrightarrow T \\
 & & \ p_3 : T \longrightarrow A \\
 & & \ p_4 : A \longrightarrow aA\{s_b\} \\
 & & \ p_5 : A \longrightarrow B \\
 & & \ p_6 : B\{s_b\} \longrightarrow bB\{s_c\} \\
 & & \ p_7 : B \longrightarrow C \\
 & & \ p_8 : C\{s_c\} \longrightarrow cC \\
 & & \ p_9 : C \longrightarrow \varepsilon \ \}
\end{array}$$

Here is a sample derivation:

$$
\begin{array}{rcl}
S & \stackrel{p_1}{\Longrightarrow} & TS \\
& \stackrel{p_1}{\Longrightarrow} & TTS \\
& \stackrel{p_2}{\Longrightarrow} & TTT \\
& \stackrel{p_3}{\Longrightarrow} & ATT \\
& \stackrel{p_4}{\Longrightarrow} & aA\{s_b\}TT \\
& \stackrel{p_4p_4p_5}{\Longrightarrow} & aaaB\{s_b,s_b,s_b\}TT \\
& \stackrel{*}{\Longrightarrow} & aaaB\{s_b,s_b,s_b\}aaaaB\{s_b,s_b,s_b,s_b\}T \\
& \stackrel{p_6}{\Longrightarrow} & aaabB\{s_b,s_b,s_c\}aaaaB\{s_b,s_b,s_b,s_b\}T \\
& \stackrel{p_6p_6p_7}{\Longrightarrow} & aaabbbC\{s_c,s_c,s_c\}aaaaB\{s_b,s_b,s_b,s_b\}T \\
& \stackrel{p_8}{\Longrightarrow} & aaabbbcC\{s_c,s_c\}aaaaB\{s_b,s_b,s_b,s_b\}T \\
& \stackrel{p_8p_8}{\Longrightarrow} & aaabbbcccC\{\}aaaaB\{s_b,s_b,s_b,s_b\}T \\
& \stackrel{p_9}{\Longrightarrow} & aaabbbcccaaaaB\{s_b,s_b,s_b,s_b\}T \\
& \stackrel{*}{\Longrightarrow} & aaabbbcccaaaabbbbcccc T \\
& \stackrel{*}{\Longrightarrow} & aaabbbcccaaaabbbbccccabc
\end{array}
$$

□

EXAMPLE 3

The following grammar derives the language COUNT-5, where COUNT-5 = $\{a^n b^n c^n d^n e^n \mid n \in \mathbb{N}\}$.

Let $G_3 = (V_N, V_T, V_I, P, S)$ with:

$$
\begin{array}{rcl}
V_N & = & \{S, A, B, C, D, E\} \\
V_T & = & \{\text{a,b,c,d,e}\} \\
V_I & = & \{s_a, s_b, s_c, s_d, s_e\} \\
P & = & \{p_1 : S \longrightarrow S\{s_a, s_b, s_c, s_d, s_e\} \\
& & \ p_2 : S \longrightarrow ABCDE \\
& & \ p_3 : A\{s_a\} \longrightarrow Aa, \quad p_4 : A \longrightarrow \varepsilon \\
& & \ p_5 : B\{s_b\} \longrightarrow Bb, \quad p_6 : B \longrightarrow \varepsilon \\
& & \ p_7 : C\{s_c\} \longrightarrow Cc, \quad p_8 : C \longrightarrow \varepsilon \\
& & \ p_9 : D\{s_d\} \longrightarrow Dd, \quad p_{10} : D \longrightarrow \varepsilon \\
& & \ p_{11} : E\{s_e\} \longrightarrow Ee, \quad p_{12} : E \longrightarrow \varepsilon \ \}
\end{array}
$$

Here is a sample derivation:

$$
\begin{array}{rcl}
S & \stackrel{p_1}{\Longrightarrow} & S\{s_a, s_b, s_c, s_d, s_e\} \\
& \stackrel{p_1p_1}{\Longrightarrow} & S\{s_a, s_b, s_c, s_d, s_e, s_a, s_b, s_c, s_d, s_e, s_a, s_b, s_c, s_d, s_e\} \\
& \stackrel{p_2}{\Longrightarrow} & A\{s_a, s_a, s_a\}B\{s_b, s_b, s_b\}C\{s_c, s_c, s_c\}D\{s_d, s_d, s_d\}E\{s_e, s_e, s_e\} \\
& \stackrel{p_7}{\Longrightarrow} & A\{s_a, s_a, s_a\}B\{s_b, s_b, s_b\}C\{s_c, s_c\}cD\{s_d, s_d, s_d\}E\{s_e, s_e, s_e\} \\
& \stackrel{p_3p_3p_4}{\Longrightarrow} & aaaB\{s_b, s_b, s_b\}C\{s_c, s_c\}cD\{s_d, s_d, s_d\}E\{s_e, s_e, s_e\} \\
& \stackrel{*}{\Longrightarrow} & aaabbbcccdddeee
\end{array}
$$

□

The derivation shows how indices are distributed among right-hand side nonterminals of a production in accordance with its definition. This example shows that $\mathcal{L}(\{\}\text{-LIG})$ is not contained in $\mathcal{L}([]\text{-LIG})$, since the latter cannot derive COUNT-5.

Here is a linguistic example. (Recall that the "linguistic" examples in this chapter are not supposed to represent serious linguistic analyses. Rather, they are meant to motivate certain features of the formal system being discussed.)

EXAMPLE 4

This grammar derives German sentences that may contain long-distance scrambling. Unlike the grammar given in Example 1, this {}-LIG avoids the problem of rightward movement. Note that we could also write a grammar that allows extraposition which enforces a c-command relation between moved element and its governing verb, thus avoiding the problem that FO-TAG (Section 3.3) and UVG (Section 4.2.5) encounter.

Let $G_4 = (V_N, V_T, V_I, P, S')$ with:

$$
\begin{array}{lcl}
V_N & = & \{S', \text{VP}, \text{NP}_{\text{nom}}, \text{NP}_{\text{dat}}, \text{NP}_{\text{acc}}\} \\
V_T & = & \{\textit{daß}, \textit{verspricht}, \textit{zu versuchen}, \textit{zu verschrotten}, \\
 & & \qquad \textit{der Meister}, \textit{niemandem}, \textit{den Kühlschrank}\}^6 \\
V_I & = & \{\text{vp}, \text{np}_{\text{nom}}, \text{np}_{\text{dat}}, \text{np}_{\text{acc}}\} \\
P & = & \{p_1\text{: } S' \rightarrow \textit{daß VP} \\
 & & \quad p_2\text{: VP} \rightarrow \text{NP}_{\text{nom}}\ \text{VP}\ \{\text{np}_{\text{nom}}\} \\
 & & \quad p_3\text{: VP} \rightarrow \text{NP}_{\text{acc}}\ \text{VP}\ \{\text{np}_{\text{acc}}\} \\
 & & \quad p_4\text{: VP} \rightarrow \text{NP}_{\text{dat}}\ \text{VP}\ \{\text{np}_{\text{dat}}\} \\
 & & \quad p_5\text{: VP} \rightarrow \text{VP VP}\ \{\text{vp}\} \\
 & & \quad p_6\text{: VP} \rightarrow \text{VP}\ \{\text{vp}\}\ \text{VP} \\
 & & \quad p_7\text{: VP}\ \{\text{np}_{\text{nom}}\ \text{np}_{\text{dat}}\ \text{vp}\} \rightarrow \textit{verspricht} \\
 & & \quad p_8\text{: VP}\ \{\text{vp}\} \rightarrow \textit{zu versuchen} \\
 & & \quad p_9\text{: VP}\ \{\text{np}_{\text{acc}}\} \rightarrow \textit{zu verschrotten} \\
 & & \quad p_{10}\text{: NP}_{\text{nom}} \rightarrow \textit{der Meister} \\
 & & \quad p_{11}\text{: NP}_{\text{dat}} \rightarrow \textit{niemandem} \\
 & & \quad p_{12}\text{: NP}_{\text{acc}} \rightarrow \textit{den Kühlschrank}\ \ \}
\end{array}
$$

Note that production p_6 leads to extraposition, while production p_5 produces a center-embedded structure. (Since this grammar does not represent the difference between finite and non-finite clauses explicitly, it does not enforce extraposition of finite clauses.) Note also that the removed indices on the productions that generate the verbs (p_7, p_8, p_9) serve as a compact representation of the verb's subcategorization frame.

Here is a sample derivation involving extraposition and long-distance scrambling :

[6] Glosses (in order): that, promises, to try, to scrap, the master, no-one, the refrigerator.

$$
\begin{aligned}
\mathrm{S}' &\overset{p_1}{\Longrightarrow} \textit{daß}\ \mathrm{VP} \\
&\overset{p_2}{\Longrightarrow} \textit{daß}\ \mathrm{NP}_{\mathrm{nom}}\ \mathrm{VP}\ \{\mathrm{np}_{\mathrm{nom}}\} \\
&\overset{p_{10}}{\Longrightarrow} \textit{daß der Meister}\ \mathrm{VP}\ \{\mathrm{np}_{\mathrm{nom}}\} \\
&\overset{p_3}{\Longrightarrow} \textit{daß der Meister}\ \mathrm{NP}_{\mathrm{acc}}\ \mathrm{VP}\ \{\mathrm{np}_{\mathrm{nom}}, \mathrm{np}_{\mathrm{acc}}\} \\
&\overset{p_6}{\Longrightarrow} \textit{daß der Meister}\ \mathrm{NP}_{\mathrm{acc}}\ \mathrm{VP}\ \{\mathrm{vp}, \mathrm{np}_{\mathrm{nom}}\}\ \mathrm{VP}\ \{\mathrm{np}_{\mathrm{acc}}\} \\
&\overset{p_4}{\Longrightarrow} \textit{daß der Meister}\ \mathrm{NP}_{\mathrm{acc}}\ \mathrm{NP}_{\mathrm{dat}}\ \mathrm{VP}\ \{\mathrm{vp}, \mathrm{np}_{\mathrm{nom}}, \mathrm{np}_{\mathrm{dat}}\}\ \mathrm{VP}\ \{\mathrm{np}_{\mathrm{acc}}\} \\
&\overset{p_5}{\Longrightarrow} \textit{daß der Meister}\ \mathrm{NP}_{\mathrm{acc}}\ \mathrm{NP}_{\mathrm{dat}}\ \mathrm{VP}\ \{\mathrm{vp}\}\ \mathrm{VP}\ \{\mathrm{vp}, \mathrm{np}_{\mathrm{nom}}, \mathrm{np}_{\mathrm{dat}}\}\ \mathrm{VP}\ \{\mathrm{np}_{\mathrm{acc}}\} \\
&\overset{p_7}{\Longrightarrow} \textit{daß der Meister}\ \mathrm{NP}_{\mathrm{acc}}\ \mathrm{NP}_{\mathrm{dat}}\ \mathrm{VP}\ \{\mathrm{vp}\}\ \textit{verspricht}\ \mathrm{VP}\ \{\mathrm{np}_{\mathrm{acc}}\} \\
&\overset{p_8 p_9}{\Longrightarrow} \textit{daß der Meister}\ \mathrm{NP}_{\mathrm{acc}}\ \mathrm{NP}_{\mathrm{dat}}\ \textit{zu versuchen verspricht zu verschrotten} \\
&\overset{p_{12} p_{11}}{\Longrightarrow} \textit{daß der Meister den Kühlschrank niemandem} \\
&\qquad\qquad \textit{zu versuchen verspricht zu verschrotten}
\end{aligned}
$$

□

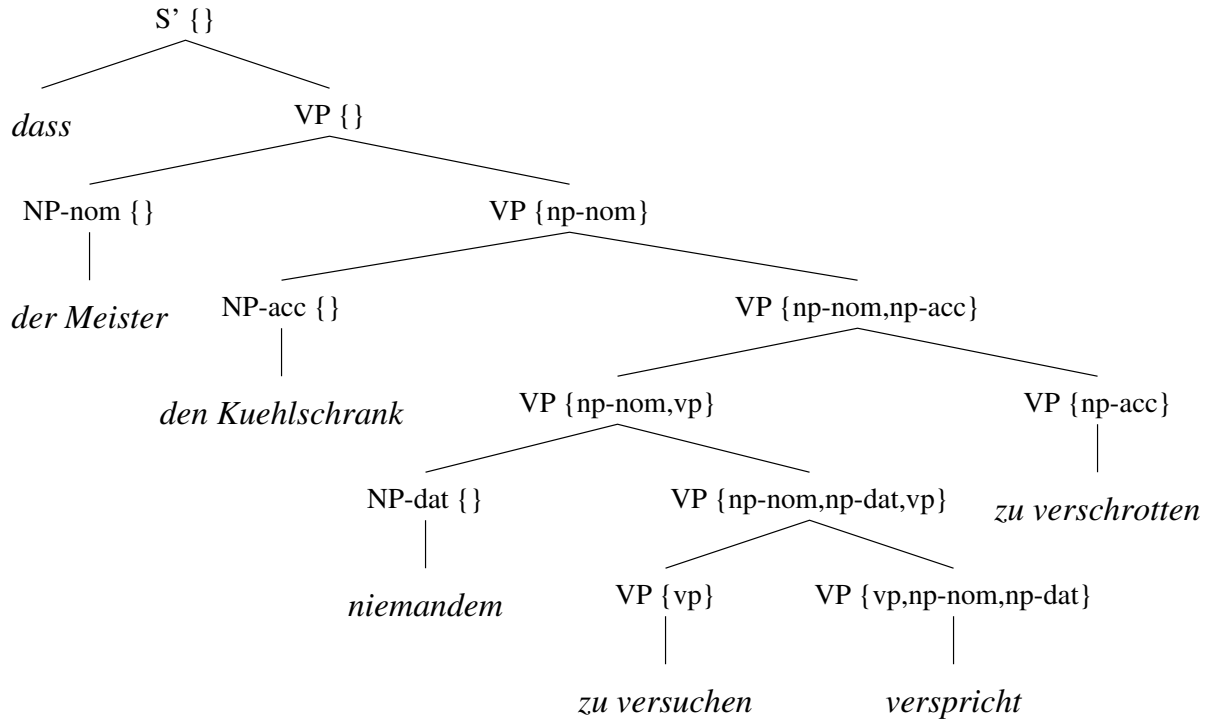


Figure 4.4: Sample derivation for grammar G_4

The derivation tree associated with this derivation is shown in Figure 4.4. (We have not formally defined the notion of derivation tree for {}-LIG, but it is easy to see that such a tree can be defined in the same way as for CFG, except that nodes are annotated with the multiset of indices.) Observe that the index symbol *vp* on the VP node dominating *zu versuchen verspricht* may appear to be copied to both of the daughters of the VP node. This is not the case; it is passed to only one of the daughter nodes. The other instance of index symbol *vp* is generated there as a result of applying production p_5 or p_6.[7]

[7]The fact that the derivation is ambiguous reflects the fact that we have not encoded finiteness features in the

One linguistic constraint we still cannot express is that scrambling is not possible out of finite clauses. Therefore, we introduce an extension of {}-LIG in which we can "close off" certain subtrees of the derivation tree. The nonterminal nodes in which these subtrees are rooted cannot pass on index symbols from their parent node to their daughter nodes.[8]

Definition 6 *A* **set-valued linear index grammar with integrity** *({}-LIG-Δ) is defined like a {}-LIG, except that a production may be marked with the* **integrity constraint** *(Δ).*

The derivation relation $\Longrightarrow$ for a {}-LIG-Δ is defined as follows. If a production p is not closed, it is used as in a {}-LIG. Suppose production $p : \Delta Au \longrightarrow v_0 B_1 s_1 v_1 \ldots v_{n-1} B_n s_n v_n$ is closed, and let $\beta, \gamma \in (V_N V_I^* \cup V_T)^*$. Then we have

$$\beta Au\gamma \Longrightarrow \beta v_0 B_1 s_1 v_1 \ldots v_{n-1} B_n s_n v_n \gamma$$

$L(G)$ for a {}-LIG-Δ is defined as usual.

A production marked with the integrity constraint is called a *closed production*. We will use the convention of saying that a subset of V_N is closed, meaning that all productions whose left-hand side nonterminal is a member of the closed set are closed. Members of a closed set will be referred to as *barriers*. The notion of closed set clearly does not affect the formal definition of the system, but allows us to capture certain generalizations, as Example 6 shows.

EXAMPLE 5

The following grammar derives the iterated a-substitution closure of COUNT-3, where an a-substitution of a grammar G is the substitution of a word of $L(G)$ for each occurrence of a in a word of $L(G)$, and the iterated a-substitution closure of G is the set of words derived from a word of $L(G)$ by successively substituting a word of $L(G)$ for each occurrence of a, until there are no occurrences of a left (see (Salomaa, 1973, p.217) for a formal definition).

Let $G_5 = (V_N, V_T, V_I, P, S)$ with:

$$\begin{array}{lcl} V_N & = & \{S, A, B\} \\ V_T & = & \{\text{b,c}\} \\ V_I & = & \{s_b, s_c\} \\ P & = & \{p_1 : \Delta S \longrightarrow A \\ & & \ p_2 : \Delta S \longrightarrow \varepsilon \\ & & \ p_3 : A \longrightarrow SA\{s_b\} \\ & & \ p_4 : A \longrightarrow B \\ & & \ p_5 : B\{s_b\} \longrightarrow bBc \\ & & \ p_6 : B\{s_b\} \longrightarrow bc \end{array}$$

Here is a sample derivation:

sample grammar, which is what disambiguates the linguistic string.

[8] This notion was first introduced for FO-TAG (Becker and Rambow, 1990; Becker et al., 1991); see Section 3.3, page 43.

$$
\begin{array}{lll}
S & \overset{p_1 p_3}{\Longrightarrow} & SA\{s_b\} \\
 & \overset{p_3 p_3}{\Longrightarrow} & SSSA\{s_b s_b s_b\} \\
 & \overset{p_4}{\Longrightarrow} & SSSB\{s_b s_b s_b\} \\
 & \overset{p_5 p_5 p_6}{\Longrightarrow} & SSSbbbccc \\
 & \overset{p_3}{\Longrightarrow} & SA\{s_b\}SSbbbccc \\
 & \overset{p_3 p_4 p_5 p_6}{\Longrightarrow} & SbbccSSbbbccc \\
 & \overset{p_2}{\Longrightarrow} & bbccSSbbbccc \\
 & \overset{p_1 p_3 p_3}{\Longrightarrow} & bbccSSA\{s_b s_b\}ccbbbccc \\
 & \overset{p_4 p_5 p_6}{\Longrightarrow} & bbccSSbbccbbbccc \\
 & \overset{p_1 p_3 p_3 p_4 p_5 p_6 p_2}{\Longrightarrow} & bbccbbccSbbccbbbccc \\
 & \overset{p_1 p_3 p_4 p_6 p_2}{\Longrightarrow} & bbccbbccbcbbccbbbccc
\end{array}
$$

We can see that the integrity constraint on p_2 is crucial, since in the third rewrite step it keeps the index s_b from being passed on from A to the newly generated S, which would lead to the generation of a b in the wrong place. □

EXAMPLE 6

We will extend the grammar G_4 given in Example 4 for German embedded clauses to handle recursively embedded finite clauses. For convenience, we omit certain features of G_4 (namely, extraposition of infinitival clauses, the verb *versuchen*, and the dative objects of *versprechen*), but it is clear how they could be included. This grammar does not generate the required commas.

Let $G_6 = (V_{\mathrm{N}}, V_{\mathrm{T}}, V_{\mathrm{I}}, P, \mathrm{S}')$ with:

V_{N} = {S′, VP, $\mathrm{NP}_{\mathrm{nom}}$, $\mathrm{NP}_{\mathrm{acc}}$}

V_{T} = {*daß*, *verspricht*, *zu versuchen*, *zu verschrotten*, *der Meister*, *der Lehrling*, *den Kühlschrank*}[9]

V_{I} = {s, vp, $\mathrm{np}_{\mathrm{nom}}$, $\mathrm{np}_{\mathrm{dat}}$, $\mathrm{np}_{\mathrm{acc}}$}

P = {p_1: Δ S′ → *daß* VP
p_2: VP → $\mathrm{NP}_{\mathrm{nom}}$ VP {$\mathrm{np}_{\mathrm{nom}}$}
p_3: VP → $\mathrm{NP}_{\mathrm{acc}}$ VP {$\mathrm{np}_{\mathrm{acc}}$}
p_4: VP → VP VP {vp}
p_5: VP → VP {s} S′
p_6: VP {$\mathrm{np}_{\mathrm{nom}}$ vp} → *verspricht*
p_7: VP {$\mathrm{np}_{\mathrm{nom}}$ s} → *verspricht*
p_8: VP {$\mathrm{np}_{\mathrm{acc}}$} → *zu verschrotten*
p_9: $\mathrm{NP}_{\mathrm{nom}}$ → *der Meister*
p_{10}: $\mathrm{NP}_{\mathrm{nom}}$ → *der Lehrling*
p_{11}: $\mathrm{NP}_{\mathrm{acc}}$ → *den Kühlschrank* }

Here is a sample derivation involving two nested embedded finite clauses:

[9] Gloss (in order): that, promises, to try, to scrap, the master, the apprentice, the refrigerator.

$$
\begin{array}{lll}
\mathrm{S}' & \stackrel{p_1}{\Longrightarrow} & \textit{daß}\ \mathrm{VP} \\
 & \stackrel{p_2}{\Longrightarrow} & \textit{daß}\ \mathrm{NP}_{\mathrm{nom}}\ \mathrm{VP}\ \{\mathrm{np}_{\mathrm{nom}}\} \\
 & \stackrel{p_5}{\Longrightarrow} & \textit{daß}\ \mathrm{NP}_{\mathrm{nom}}\ \mathrm{VP}\ \{\mathrm{s\ np}_{\mathrm{nom}}\}\ \mathrm{S}' \\
 & \stackrel{p_{10}p_7}{\Longrightarrow} & \textit{daß der Meister verspricht}\ \mathrm{S}' \\
 & \stackrel{p_1}{\Longrightarrow} & \textit{daß der Meister verspricht daß}\ \mathrm{VP} \\
 & \stackrel{p_3p_{11}p_2p_{10}}{\Longrightarrow} & \textit{daß der Meister verspricht daß den Kühlschrank der Lehrling} \\
 & & \quad \mathrm{VP}\ \{\mathrm{np}_{\mathrm{acc}}\}\ \mathrm{VP}\ \{\mathrm{np}_{\mathrm{nom}}, \mathrm{vp}\} \\
 & \stackrel{p_8p_6}{\Longrightarrow} & \textit{daß der Meister verspricht daß den Kühlschrank der Lehrling} \\
 & & \quad \textit{zu verschrotten verspricht}
\end{array}
$$

Note that the scrambled NP *den Kühlschrank* cannot scramble out of the finite clause because of the integrity constraint on S′ in production p_1. □

It is not clear whether a {}-LIG without integrity could derive the language in Example 5, which motivates the following conjecture:

Conjecture 2 *The inclusion* $\mathcal{L}(\{\}\text{-LIG}) \subset \mathcal{L}(\{\}\text{-LIG-}\Delta)$ *is proper.*

We return to the issue in Section 4.3.4.

We will now define several types of restricted {}-LIGs.

Definition 7 *A* **safe** {}-LIG *(or* {}-LIG-Δ*) is a* {}-LIG *(*{}-LIG-Δ*) which contains no* ε*-productions (productions of the form* $As \longrightarrow \varepsilon$*) and no chain productions (productions of the form* $As \longrightarrow Bt$*).*

Definition 8 *A* **linearly-restricted** *derivation in a* {}-LIG *or a* {}-LIG-Δ *is a derivation* ϱ : $S \stackrel{*}{\Longrightarrow} w$ *with* $w \in V_{\mathrm{T}}^*$ *such that:*

1. *The number of index symbols added (and hence removed) during the derivation is linearly bounded by* $|w|$.
2. *The number of* ϵ*-productions used during the derivation is linearly bounded by* $|w|$.

We let $L_R(G) = \{w \mid$ there is a derivation $\varrho : S \stackrel{*}{\Longrightarrow} w$ such that ϱ is linearly-restricted$\}$, and we let $\mathcal{L}_R(\mathcal{F}) = \{L_R(G) | G \in \mathcal{F}\}$. If G is a {}-LIG ({}-LIG-Δ) such that $L_R(G) = L(G)$, we say that G is linearly restricted. Observe that in a safe {}-LIG ({}-LIG-Δ), every derivation is linearly restricted, and hence a safe {}-LIG is linearly restricted.

Definition 9 *Let* q *be a function over the positive integers. A* q**-form-restricted** *derivation in a* {}-LIG *is a derivation* $\varrho : S \stackrel{*}{\Longrightarrow} w$ *with* $w \in V_{\mathrm{T}}^*$ *such that the total number of index symbols in any single sentential form of the derivation is bounded by* $q(|w|)$.

We let $L^q_{\mathrm{fR}}(G) = \{w \,|\, \text{there is a derivation } \varrho : S \stackrel{*}{\Longrightarrow} w \text{ such that } \varrho \text{ is } q\text{-form-restricted}\}$, and we let $\mathcal{L}^q_{\mathrm{fR}}(\mathcal{F}) = \{L^q_{\mathrm{fR}}(G) \,|\, G \in \mathcal{F}\}$. If G is a {}-LIG such that $L^q_{\mathrm{fR}}(G) = L(G)$, we say that G is q-form-restricted. Note that in a linearly-restricted derivation, the restriction applies to the *total* number of index symbols (and ε-productions) used during the derivation, while in a q-form-restricted derivation, the restriction applies to the number of index symbols in any one sentential form. Clearly, a linearly restricted derivation is also a q-form-restricted derivation for some $q \in O(n)$, but the converse is not true. Note that a c-form-restricted {}-LIG for $c \in O(1)$ is equivalent to a context-free grammar.

4.3.2 Normal Forms

This section presents three normal forms for both {}-LIG and {}-LIG-Δ. The first is based on the definition of IGs given in (Hopcroft and Ullman, 1979, p.389).

Definition 10 *A {}-LIG ({}-LIG-Δ) $G = (V_{\mathrm{N}}, V_{\mathrm{T}}, V_{\mathrm{I}}, P, S)$ is in* **restricted index normal form** *or RINF if all productions in P are of one of the following forms (where $A, B \in V_{\mathrm{N}}$, $f \in V_{\mathrm{I}}$ and $\alpha \in (V_{\mathrm{T}} \cup V_{\mathrm{N}})^*$):*

1. $A \longrightarrow \alpha$
2. $A \longrightarrow Bf$
3. $Af \longrightarrow B$

If G is a {}-LIG-Δ, only productions of type (i) are closed.

Theorem 3 *For any {}-LIG ({}-LIG-Δ), there is an equivalent {}-LIG ({}-LIG-Δ) in RINF.*

Proof. Let $G = (V_{\mathrm{N}}, V_{\mathrm{T}}, V_{\mathrm{I}}, P, S)$ be a {}-LIG ({}-LIG-Δ). We will construct a grammar $G' = (V'_{\mathrm{N}}, V_{\mathrm{T}}, V_{\mathrm{I}}, P', S)$ in RINF such that $L(G) = L(G')$. Let LHS $= \{(p, A, u) | p \in P, A$ its left-hand symbol, u the associated removed indices$\}$ and RHS $= \{(p, B, u) | p \in P, B$ a symbol in its right-hand side, u the associated added indices$\}$. Let $V'_{\mathrm{N}} = \{A^{(p,i)} \,|(p, A, u) \in$ LHS and $0 \leq i \leq |u|\} \cup \{B^{(p,i)} | (p, B, u) \in$ RHS and $0 \leq i \leq |u|\}$. We now define P'. Let $p \in P$ with $p : Au \longrightarrow v_0 B_1 s_1 v_1 \ldots v_{n-1} B_n s_n v_n$, $u = u^{(1)} \cdots u^{(k)}$ and $s_i = s_i^{(1)} \cdots s_i^{(k_i)}$, $1 \leq i \leq n$, where the $u^{(j)}$ and $s_i^{(j)}$ are index symbols. Then the following productions are in P':

1. $A^{(p,i-1)} u^{(i)} \longrightarrow A^{(p,i)}$, for $1 \leq i \leq k$
2. $B_i^{(p,j)} \longrightarrow B_i^{(p,j-1)} s_i^{(j)}$, for $1 \leq i \leq n$ and $1 \leq j \leq k_i$
3. $A^{(p,k)} \longrightarrow v_0 B_1^{(p,k_1)} v_1 \ldots v_{n-1} B_n^{(p,k_n)} v_n$

If in (i) $k = 0$ or if in (ii) $k_i = 0$, then no such productions are in P'. If G is a {}-LIG-Δ and p is closed, then the third production above is also closed. No other production are in P. Clearly, G' is in RINF. It is easy to show (by induction over the number of steps in a derivation) that $L(G) = L(G')$. ■

We have a weaker version of the Chomsky Normal Form for {}-LIG:

Definition 11 *A* {}-LIG *(*{}-LIG-Δ*)* $G = (V_N, V_T, V_I, P, S)$ *is in* **Extended Two Form** *(ETF) if every production in* P *has the form* $As \rightarrow B_1 s_1 B_2 s_2$, $As \rightarrow Bs'$, *or* $A \rightarrow a$, *where* $A, B_1, B_2 \in V_N$, $s, s_1, s_2, s' \in V_I^*$, *and* $a \in V_T \cup \{\varepsilon\}$.

Theorem 4 *For any* {}-LIG *(*{}-LIG-Δ*), there is an equivalent* {}-LIG *(*{}-LIG-Δ*) in ETF.*

Proof. For any {}-LIG ({}-LIG-Δ) $G = (V_N, V_T, V_I, P, S)$ we construct an equivalent grammar $G' = (V'_N, V_T, V_I, P', S)$ in ETF. Let $V'_N = V_N \cup \{[p,i] | p \in P \text{ and } 1 \leq i \leq \#_{V_N \cup V_T}(\mathsf{rhs}(p))\} \cup \{[a] | a \in V_T\}$. The construction of P' consists of two steps:

- For every production p in P, replace every occurrence of a terminal symbol a in $\mathsf{rhs}(p)$ by nonterminal $[a]$ and add production $([a] \longrightarrow a)$ to P'.
- For every production p in P such that $\#_{V_N \cup V_T}(\mathsf{rhs}(p)) > 2$, cover p by means of productions $p_1, \ldots, p_q$, $q \geq 2$, which have right-hand side length not greater than 2. This is done using standard techniques for the construction of Chomsky normal form grammars out of context-free grammars, where the indices are "spread out" in the obvious manner. More precisely, if p is $As \longrightarrow B_1 s_1 \ldots B_n s_n$, then replace p in P' by the following sequence of productions:

$$\begin{aligned} p_1 &: As \longrightarrow B_1 s_1 [p,1] \\ p_2 &: [p,1] \longrightarrow B_2 s_2 [p,2] \\ &\vdots \\ p_{n-1} &: [p,n-2] \longrightarrow B_{n-1} s_{n-1} B_n s_n \end{aligned}$$

If G is a {}-LIG-Δ and p is a closed production, then p_1 is a closed production.

It is not difficult to verify that G' obtained as above has the claimed properties. ■

Observe that the construction of G' preserves safety, linear restrictedness, and q form-restrictedness. In the case of q form-restrictedness, the greatest number of indices in any sentential form of a derivation of a given string w may actually increase, but the increase is by a constant amount determined by G, so that G' is still q form-restricted.

Because we have a set of index symbols, we can encode the information normally encoded in the nonterminal symbol in the associated index set, so that we need only one nonterminal.

Definition 12 *A* {}-LIG $G = (V_T, V_N, V_I, S, P)$ *is in* **Single Nonterminal Form** *(SNF) if there is only one nonterminal symbol (i.e.,* $|V_N| = 1$*).*

Theorem 5 *For every* {}-LIG G *there is an equivalent* {}-LIG G' *in SNF.*

Proof. Let $G = (V_\mathrm{T}, V_\mathrm{N}, V_\mathrm{I}, S, P)$ be a {}-LIG or {}-LIG-Δ. We will construct a $G' = (V_\mathrm{T}, \{X\}, V'_\mathrm{I}, X, P')$ in SNF. Let the set of "nonterminal indices" NI be $\{[A] \mid A \in V_\mathrm{N}\}$. Let $V'_\mathrm{I} = V_\mathrm{I} \cup \mathrm{NI}$. P' contains the following productions (and no others):

1. $X\{A\} \cup s \longrightarrow w_0 X\{B_1\} \cup s_1 w_1 \cdots X\{B_n\} \cup s_n w_n$ if $As \longrightarrow w_0 B_1 s_1 w_1 \cdots B_n s_n w_n$ is in P and $A \neq S$.
2. $X \longrightarrow w_0 X[B_1] s_1 w_1 \cdots X[B_n] s_n w_n$ if $S \longrightarrow w_0 B_1 s_1 w_1 \cdots B_n s_n w_n$ is in P.

(If G is a {}-LIG-Δ, then productions in G' derived from closed productions in G' are also closed.) It is clear that during a derivation, exactly one element of NI is associated with each nonterminal of each sentential form, except S. Therefore, the NI indices are always immediately removed in the first rewrite rule that rewrites their nonterminal. Furthermore, derivations can only start with productions derived from productions whose left-hand symbol is S, since all other productions require removing an index. It is thus easy to see that $L(G') = L(G)$. ■

Note that the existence of SNF means that we could define a formalism which is equivalent to {}-LIG, but in which there are no nonterminals at all, and in which the rewrite rules (and the sentential forms) are defined in terms of sets of index symbols. This would only be a slight notational variant of a {}-LIG in SNF; we omit the details.

4.3.3 Weak Generative Capacity of {}-LIG

This section discusses the weak generative capacity of {}-LIG. We first examine the relationship between {}-LIG (and {}-LIG-Δ) and CSG. While CSG is a powerful formalism, proving that the set of languages generated by a certain formalism is contained in $\mathcal{L}$(CSG) can be tricky, typically because of ε-productions. For example, while it is known that $\mathcal{L}$(MG$-\varepsilon$) (where X$-\varepsilon$ denotes the set of grammars of type X without ε-productions) and hence $\mathcal{L}$(USCG$-\varepsilon$) are a proper subset of $\mathcal{L}$(CSG), it is not known whether the inclusion $\mathcal{L}$(MG) $\subseteq$ $\mathcal{L}$(CSG) holds.[10] The problem of showing this inclusion appears to be a difficult one, given that it remains unsolved nearly 30 years after the definition of the formalisms involved. Similarly, in the case of {}-LIG, inclusion in $\mathcal{L}$(CSG) will be proven only with certain restrictions (related to, among other things, ε-productions). At first sight, these restrictions appear to be of a very unfortunate kind, namely restrictions on the derivations, not restrictions on the form of the grammars. However, we will see in Section 4.4 that we can give a more intuitive definition of the restrictions when we turn to the UVG-DL formalism.

Theorem 6 *1.* $\mathcal{L}_R(\{\}\text{-LIG}) \subseteq \mathcal{L}(\mathrm{CSG})$.

2. $\mathcal{L}_R(\{\}\text{-LIG-}\Delta) \subseteq \mathcal{L}(\mathrm{CSG})$.

[10] In fact, it is not even known whether the inclusion $\mathcal{L}$(MG) $\subseteq$ $\mathcal{L}$(RE) is proper.

Proof. (i) Let G be a {}-LIG. We assume without loss of generality that G is in RINF. We construct a linear bounded automaton M which accepts $\text{Ł}_R(G)$. The construction will only be given informally; the formal details can easily be worked out. M has as its workspace the input string w plus a number of blank tape squares equal to the sum of the linear bound on the number of index symbols used and the linear bound on the number of ε-productions used (plus one) times $|w|$. The tape symbols of M are $V_{\text{N}} \cup V_{\text{T}} \cup V_{\text{I}} \cup \{@\}$, where @ is a new symbol not in $V_{\text{N}} \cup V_{\text{T}} \cup V_{\text{I}}$ which is treated like a terminal symbol of G. M first writes S on the tape next to the input word. Then M nondeterministically performs a derivation in G of w, except that the following rule substitutions are made.

- If p is of the form $A \longrightarrow \varepsilon$, then production $A \longrightarrow @$ is used instead.
- If p is of the form $Af \longrightarrow B$, then $Af \longrightarrow @B$ is used instead.

Because all rewrite rules are now length-preserving, and because of the restriction on derivations we are using, M can perform the derivation in the space provided. Then, all occurrences of @ are deleted, and the resulting string is compared to the input string. Clearly, M accepts w iff $w \in L(G)$.

(ii) This case is shown analogously. (The integrity constraints are coded in the finite-state control of the automaton along with the rest of the grammar.) ■

What sort of languages could a {}-LIG possibly not generate? Let us discuss the copy language $L = \{ww \mid w \in \{a,b\}^*\}$, and let us suppose that it is generated by G, a {}-LIG. This language cannot be generated by a CFG. We therefore know that for any integer M, there are infinitely many strings in L whose derivation in G is such that at some point, an index multiset in the sentential form contains more than M index symbols (since any finite use of index symbols can be simulated by a pure CFG). It must be the case that this unbounded multiset is crucial in restricting the second half of the generated string in such a way that it copies the first half (again, since a pure CFG cannot derive such strings). However, it is impossible for a data structure like a (multi-)set (over a finite index alphabet) to record the required sequential information. Therefore, the second half of the string cannot be adequately constrained, and g cannot exist.

This argument motivates, but does not quite prove, the following conjecture.

Conjecture 7 *$\{ww \mid w \in \{a,b\}^*\}$ is not in $\mathcal{L}(\{\}\text{-LIG})$.*

If this conjecture is true, we have the interesting fact that {}-LIG and []-LIG are incomparable: the former can count to any integer, while the latter can only count to four, but the latter can derive the copy language, which the former cannot. The contribution of the two data structures, the stack and the multiset, would turn out to be different and complementary. We define a LIG-variant equipped with both data structures in Section 4.6.3.

As discussed in Section 3.2.2, page 37, it has been argued that the copy language is an important test case for formalisms that are used for natural language syntax, since Swiss German sentences with embedded infinitivals can be mapped onto it by a homomorphism (Shieber, 1985). If the

conjecture is true, {}-LIG and the formalisms that we will show are weakly equivalent to it are not adequate formalisms for the representation of natural language, and we need to extend the power of these formalisms by the power that TAG provides; put differently, that we must move from string rewriting to tree rewriting. This is exactly the move made in Section 4.6.

4.3.4 Closure Properties

In this section, we discuss the closure properties of $\mathcal{L}(\{\}\text{-LIG})$. We show that $\mathcal{L}(\{\}\text{-LIG})$ and $\mathcal{L}(\{\}\text{-LIG-}\Delta)$ are substitution-closed full AFLs.

Theorem 8 *$\mathcal{L}(\{\}\text{-LIG})$ ($\mathcal{L}(\{\}\text{-LIG-}\Delta)$) is a substitution-closed full AFL.*

Proof. Since $\mathcal{L}(\{\}\text{-LIG})$ ($\mathcal{L}(\{\}\text{-LIG-}\Delta)$) contains all context-free languages, it contains all regular languages, and therefore it is sufficient to show that $\mathcal{L}(\{\}\text{-LIG})$ ($\mathcal{L}(\{\}\text{-LIG-}\Delta)$) is closed under intersection with regular languages and substitution (Salomaa, 1973, Theorem 1.6, p.129).

1. Closure under intersection with regular languages is shown using the same technique as for context-free languages (see, e.g., Salomaa (1973, Theorem 6.7, p. 59)). In brief, if G is a $\mathcal{L}(\{\}\text{-LIG})$ ($\{\}\text{-LIG-}\Delta$) and M a finite-state automaton, a new grammar G' is defined whose nonterminals are of the form $[A, q_1, q_2]$, where A is a nonterminal from G and q_1, q_2 are states of M. A derivation of the form $[A, q_1, q_2] \stackrel{*}{\Longrightarrow}_{G'} w$, w a terminal string, means that $A \stackrel{*}{\Longrightarrow}_G w$ and also M accepts w while moving from state q_1 to q_2. The presence of the indices (and the integrity constraints) in the grammar and of the indices in the derivations does not affect the method of construction of G', and the proof that $L(G') = L(G) \cap L(M)$ is straightforward.

2. To show closure under substitution, the same technique as for context-free languages (see, e.g., Salomaa (1973, Theorem 3.5, p. 23)) can be used, and we just sketch the proof here. Let G_i be the {}-LIG ({}-LIG-Δ) that generates the substitution for terminal symbol a_i ($1 \leq i \leq |V_T|$), and let the nonterminal and index sets of the G_i and of G be pairwise disjoint. Then a new grammar is constructed whose nonterminal and index alphabets are just the unions of those of G and the G_i, and whose set of productions is again the union of the productions of G and the G_i, except that every occurrence in a production of G of nonterminal a_i is replaced by S_i, where S_i is the start symbol of G_i. It can easily be seen that the grammar generates the proper language. ■

In the proof of closure under substitution, it is the disjointedness of the sets of terminal, nonterminal, and, crucially, index symbols of G and the corresponding sets of G_i that allows us to simulate the substitutions by separating the derivations in G from those in the G_i (and those in the G_i from one another). This technique does not extend to iterated substitution, since iterated substitution may include an unbounded number of nested substitutions. However, using the integrity constraint, we can show that {}-LIG-Δ is also closed under iterated substitution and iterated a-substitution (see Section 4.3.4). This does not appear to be the case for {}-LIG. (For example, it is doubtful whether the language in Example 6 can be generated by a {}-LIG.) This justifies the conjecture that integrity constraints increase the weak generative capacity of {}-LIG (Conjecture 2, page 74).

4.3.5 A Formal Automaton: {}-PDA

In this section, we introduce a formal automaton that accepts exactly the languages generated by {}-LIGs. The automaton is derived from the push-down automaton, PDA, which accepts a context-free language. The PDA is modified by equipping each stack symbol with a multiset of indices. The formal definition is as follows:

Definition 13 *A* **multi-set valued index pushdown automaton** *or* {}-PDA *is a system* $(Q, \Sigma, \Gamma, \Upsilon, \delta, q_0, Z_0)$ *where*

1. Q *is a finite set of states;*
2. $q_0 \in Q$ *is the start state;*
3. Σ *is a finite set of input symbols;*
4. Γ *is a finite set of stack symbols;*
5. Υ *is a finite set of index symbols;*
6. $Z_0 \in \Gamma \cup \{\epsilon\}$ *is the symbol on the stack initially;*
7. δ *is a mapping from* $Q \times (\Sigma \cup \{\varepsilon\}) \times \Upsilon^* \times \Gamma$ *to finite subsets of* $Q \times (\Gamma\Upsilon^*)^*$.

An **instantaneous description** *(ID) is a "snapshot" of the automaton. It is a member of* $Q \times (\Gamma\Upsilon^*)^* \times \Sigma^* \times \Sigma^*$*; its four components represent the current state, the current configuration of the stack (exceptionally, with the top of the stack to the left), the part of the input word that has been read, and the part of the input word that has not yet been read, respectively.*

The **move relation** *of a* {}-PDA M *is a (reflexive, transitive) relation* $\vdash_M$ *over the set of IDs. Let* $I_1 = (q, Zt\gamma, w_1, aw_2)$ *be an ID, with* $\gamma \in (\Gamma\Upsilon^*)^*, Z \in \Gamma$, *and* $t \in \Upsilon^*$, *and* I_2 *also an ID. Suppose* $(q', Z_1 s_1 \cdots Z_n s_n)$ *is in* $\delta(q, a, s, Z)$. *Then we have* $I_1 \vdash (q', Z_1 t_1 \cdots Z_n t_n \gamma, w_1 a, w_2)$ *if there are* $t', t'' \in \Upsilon^*$ *with* $t' \in \mathsf{Perm}(t)$ *such that* $t' = st''$, *and there are* $t''_1, \ldots, t''_n \in \Upsilon^*$ *such that* $t''_1 \cdots t''_n \in \mathsf{Perm}(t'')$, *and* $t_i = s_i t''_i$, $1 \leq i \leq n$.

The language accepted (by null stack) by a {}-PDA $M = (Q, \Sigma, \Gamma, \Upsilon, \delta, q_0, Z_0)$ *is defined to be*

$$N(M) = \{w \mid (q_0, Z_0, \varepsilon, w) \vdash^* (q, \varepsilon, w, \varepsilon), q \in Q\}.$$

We will say that a move of the automaton distributes the indices t among $t_1, \ldots t_n$ in accordance with δ. A set of final states can be included in the formal definition of the {}-PDA, and acceptance by final state defined in the usual way.

We now prove in two steps that the set of languages accepted by {}-PDAs is equal to the set of languages generated by {}-LIGs. The proofs are based on those for simple CFGs and PDAs given by Hopcroft and Ullman (1979, p.115ff).

Lemma 9 *Let* M *be a* {}-PDA. *Then there is a* {}-LIG G *such that* $N(M) = L(G)$.

Proof. Let $M = (Q, \Sigma, \Gamma, \Upsilon, \delta, q_0, Z_0)$. We construct a $\{\}$-LIG $G = (V_N, \Sigma, \Upsilon, P, S)$. Let $V_N = \{[q_1, Z, q_2] \mid q_1, q_2 \in Q \text{ and } Z \in \Gamma\} \cup \{S\}$. The set P contains the following productions (and no others):

1. $S \longrightarrow [q_0, Z_0, q]$ for all q in Q;
2. $[q, Z, q_{m+1}]s \longrightarrow a[q_1, Y_1, q_2]s_1[q_2, Y_2, q_3]s_2 \cdots [q_m, Y_m, q_{m+1}]s_m$ for all $q, q_1, \ldots, q_{m+1}$ in Q, a in $\Sigma \cup \{\varepsilon\}$, $Z, Y_1, \ldots, Y_m$ in Γ, and $s, s_1, \ldots, s_m$ in Υ^* such that $\delta(q, a, s, A)$ contains $(q_1, Y_1 s_1 \cdots Y_m s_m)$.

We will show by induction (on the number of moves of M or the length of the derivation in G) that $[q, Z, p]t \stackrel{*}{\Longrightarrow} w$ iff $(q, Zt, \varepsilon, w) \stackrel{*}{\vdash}_M p, \varepsilon, w, \varepsilon)$. The lemma follows from this claim by application of a production of type (i).

We first show the "only if" direction. Specifically, we show by induction on i that $[q, Z, p]t \stackrel{i}{\Rightarrow}_G w$ implies $(q, Zt, \varepsilon, w) \stackrel{*}{\vdash}_M (p, \varepsilon, w, \varepsilon)$. To show the base case $i = 1$, suppose $[q, Z, p]t \stackrel{1}{\Longrightarrow} w$. By construction of G, $w \in V_T$ and $\delta(q, w, t, Z)$ contains (p, ε), and the base case follows. For the induction step, suppose $[q, Z, p]t \stackrel{i}{\Longrightarrow} w$, and that the first production used in the derivation is of the form $[q, Z, q_{m+1}]s \longrightarrow a[q_1 Z_1 q_2]s_1 \cdots [q_m Z_m q_{m+1}]s_m$ where $q_m + 1 = p$. By construction of G, $\delta(q, a, s, Z)$ contains $(q_1, Y_1 s_1 \cdots Y_m s_m)$. Let $w' \in V_T^*$ such that $w = aw'$. Then $(q, Zt, \varepsilon, aw') \vdash 1(q_1, Y_1 t_1 \cdots Y_m t_m, a, w')$ with t distributed over $t_1, \ldots, t_m$ in accordance with $\delta(q, a, s, Z)$. There are $w_1, \ldots, w_m \in V_T^*$ such that $w_1 \cdots w_m = w'$ and $[q_j, Y_j, q_{j+1}]t_j \stackrel{i_j}{\Longrightarrow} w_j$ with $i_j < i$ for $1 \leq j \leq m$. By induction hypothesis, we have $(q_j, Y_j t_j, \varepsilon, w_j) \stackrel{*}{\vdash} (q_{j+1}, \varepsilon, w_j, \varepsilon)$. Clearly, we then have $(q_j, Y_j t_j \cdots Y_m t_m, \varepsilon, w_j \cdots w_m) \stackrel{*}{\vdash} (q_m, Y_{j+1} t_{j+1} \cdots Y_m t_m, w_j, w_{j+1} \cdots w_m)$ for $1 \leq j \leq m$, and the proposition follows.

We now show the "if" direction. Specifically, we show by induction on i that $(q, Zs, \varepsilon, w) \stackrel{i}{\vdash}_M (p, \varepsilon, w, \varepsilon)$ implies $[q, Z, p]s \stackrel{*}{\Longrightarrow}_G w$. First, to show the base case, suppose $i = 1$. Then (p, ε) is in $\delta(q, a, s, Z)$, and by construction of G, $[q, Z, p]s \longrightarrow a$ is a production of G. For the induction step, suppose $(q, Z, \varepsilon, w) \vdash^i (p, \varepsilon, w, \varepsilon)$. Let the first move of the automaton be $(q, Zt, \varepsilon, w) \vdash (q', Y_1 t_1 \cdots Y_m t_m, a, w')$, where t is distributed among $t_1, \ldots, t_m$ in accordance with $\delta(q, a, s, Z)$, which includes $(q', Y_1 s_1 \cdots Y_m s_m)$. By construction of G, there is a production $[q, Z, q_{m+1}]s \longrightarrow a[q_1 Y_1 q_2]s_1[q_2, Y_2, q_3]s_2 \cdots [q_m Y_m q_{m+1}]s_m$ in P. The processing of M can be broken up into the following steps, where $w_1 \cdots w_m = w'$:

$$\begin{aligned}(q', Y_1 t_1 \cdots Y_m t_m, \varepsilon, w_1 \cdots w_m) \quad &\stackrel{i_1}{\vdash}_M \quad (q', Y_2 t_2 \cdots Y_m t_m, w_1, w_2 \cdots w_m) \\ &\stackrel{i_2}{\vdash}_M \quad \cdots \\ &\stackrel{i_{m-1}}{\vdash}_M (p, \varepsilon, w_1 \cdots w_m, \varepsilon)\end{aligned}$$

Since $i_j < i$, $1 \leq j \leq m$, we have by induction hypothesis that $[q_j Z_j q_{j+1}]t_j \Longrightarrow_G w_j$, $1 \leq j \leq m$. The proposition follows. ∎

Lemma 10 *Let G be a $\{\}$-LIG. Then there is a $\{\}$-PDA M such that $L(G) = N(M)$.*

Proof. Let $G = (V_N, V_T, V_I, P, S)$. We construct a $\{\}$-PDA $M = (\{q\}, V_T, V_N \cup V_T, V_I, \delta, q, S)$ where δ is defined as follows:

1. $\delta(q, \varepsilon, s, A) = \{(q, \alpha) \mid As \longrightarrow \alpha \in P\}$
2. $\delta(q, a, \varepsilon, a) = \{(q, \varepsilon) \mid a \in V_{\mathrm{T}}\}$

δ maps other tuples to the empty set. Let $v, v' \in V_{\mathrm{T}}^*$ with $w = vv'$ and $\alpha \in ((V_{\mathrm{N}}V_{\mathrm{I}}^*) \cup V_{\mathrm{T}})^*$. We will show by induction that $S \stackrel{*}{\Longrightarrow} v\alpha$ by a leftmost derivation iff $(q, S, \varepsilon, w) \stackrel{*}{\vdash} (q, \alpha, v, v')$.

We first show the "only if" direction. Specifically, we show that $S \stackrel{i}{\Longrightarrow}_G v\alpha$ by a leftmost derivation implies $(q, S, \varepsilon, w) \stackrel{*}{\vdash}_M q, \alpha, v, v')$, where α is in $(V_{\mathrm{N}}V_{\mathrm{I}}^*V_{\mathrm{T}}^*)^*$. The base case ($i = 0$) is trivial. For the induction step, observe that if $\alpha = \varepsilon$, we are done. Thus, assume $S \stackrel{i}{\Longrightarrow} vAt\alpha'$ with $A \in V_{\mathrm{N}}$, $t \in V_{\mathrm{I}}^*$, and $\alpha' \in ((V_{\mathrm{N}} \cup V_{\mathrm{T}})((V_{\mathrm{N}}V_{\mathrm{I}}^*) \cup V_{\mathrm{T}})^*) \cup \{\varepsilon\}$. Let p be $As \longrightarrow w_0B_1s_1w_1 \cdots w_{n-1}B_ns_nw_n$ with $A, B_i \in V_{\mathrm{N}}$, $s, s_i \in V_{\mathrm{I}}^*$, $1 \leq i \leq n$, and $w_i \in V_{\mathrm{T}}^*$, $0 \leq i \leq n$. Then we have:

$$S \stackrel{i+1}{\Longrightarrow} vw_0B_1t_1w_1 \cdots w_{n-1}B_nt_nw_n\alpha'$$

where t is distributed among $t_1, \ldots, t_n$ in accordance with δ. By construction of M, $(q, w_0Y_1t_1w_1 \cdots w_{n-1}Y_nt_nw_n)$ is in $\delta(q, \varepsilon, s, A)$. Let $v' = w_0v''$. We have:

$$\begin{aligned}(q, At\alpha', v, v') &\vdash (q, w_0B_1t_1w_1 \cdots w_{n-1}B_nt_nw_n\alpha', v, v') \\ &\stackrel{*}{\vdash} (q, B_1t_1w_1 \cdots w_{n-1}B_nt_nw_n\alpha', vw_0, v'')\end{aligned}$$

where t is distributed among $t_1, \ldots, t_n$ in accordance with δ. The induction step follows from the induction hypothesis and the transitivity of the $\vdash$ relation.

We now show the "if" direction. Specifically, we show by induction on i that $(q, S, \varepsilon, w) \stackrel{i}{\vdash}_M (q, \alpha, v, v')$ implies $S \stackrel{*}{\Longrightarrow}_G v\alpha$ by a leftmost derivation, where α is in $(V_{\mathrm{N}}V_{\mathrm{I}}^*V_{\mathrm{T}}^*)^*$. The base case ($i = 0$) is trivial. For the induction step, suppose $(q, S, \varepsilon, w) \stackrel{i}{\vdash}_M (q, \alpha, v, v')$. If $\alpha = \varepsilon$, we are done. Assume $\alpha = At\alpha'$ with $A \in V_{\mathrm{N}}$, $s \in V_{\mathrm{I}}^*$, and $\alpha' \in ((V_{\mathrm{N}} \cup V_{\mathrm{T}})((V_{\mathrm{N}}V_{\mathrm{I}}^*) \cup V_{\mathrm{T}})^*) \cup \{\varepsilon\}$. There are two cases, depending on the type of move of the automaton.

1. Suppose the next move is of the form $\delta(q, \varepsilon, s, A) = \{(q, w_0B_1t_1w_1 \cdots w_{n-1}B_nt_nw_n) \mid p : As \longrightarrow w_0B_1t_1w_1 \cdots w_{n-1}B_nt_nw_n \in P\}$. Then we have

$$(q, \alpha, v, v') \vdash (q, w_0B_1t_1w_1 \cdots w_{n-1}B_nt_nw_n\alpha', v, v')$$

where t is distributed over $t_1, \ldots, t_n$ in accordance with δ. Using the induction hypothesis, we have $S \stackrel{*}{\Longrightarrow} vAt\alpha'$, and by the transitivity of the $\Longrightarrow$ relation, we get $S \stackrel{*}{\Longrightarrow} vw_0Y_1t_1w_1 \cdots w_{n-1}Y_nt_nw_n\alpha'$.

2. Now assume the next move is licensed by $\delta(q, a, \varepsilon, a) = (q, \varepsilon)$ for $a \in V_{\mathrm{T}}$. Let $v' = av''$ and $\alpha = a\alpha'$. Then we have

$$(q, \alpha, v, av'') \vdash (q, \alpha, va, v'')$$

Clearly, we still have $S \stackrel{*}{\Longrightarrow}_G v\alpha$ by a leftmost derivation. ■

The previous lemmas give us the following theorem.

Theorem 11 $\mathcal{L}(\{\}\text{-PDA}) = \mathcal{L}(\{\}\text{-LIG})$

4.3.6 Parsing Complexity

We address here the issue of recognition/parsing for languages in the class $\mathcal{L}(\{\}\text{-LIG})$. We will show that both problems are solvable in polynomial deterministic time for certain subclasses of {}-LIG and {}-LIG-Δ. The approach is based on that proposed by Satta (1993) for UVG.

In order to present the main result, we need to develop some additional notation. We define an *I-counter* to be a one-dimensional array whose elements are non-negative integers and are addressed by an index i ranging in V_{I}. In what follows, each non-terminal A from which starts a derivation π of the kind $At \overset{\pi}{\Rightarrow}_G w$, $A \in V_{\mathrm{N}}$ and $w \in V_{\mathrm{T}}^*$, will be associated with an I-counter γ such that $\gamma(i) = \#_i(t)$. For $s \in V_{\mathrm{I}}^*$, we define γ_s as follows:

$$\gamma_s(i) = \#_i(s)$$

The sum of I-counters is defined in the obvious way. The following norm will be used for I-counters:

$$\|\gamma\| \quad = \quad \sum_{i \in V_{\mathrm{I}}} \gamma(i).$$

We have $\gamma \leq \gamma'$ if $\gamma(i) \leq \gamma'(i)$ for all $i \in V_{\mathrm{I}}$.

To prove that polynomial form-restricted multi-set valued linear index languages (with or without integrity) can be recognized in polynomial deterministic time, we use a dynamic programming technique based on the CKY parser for CFG (independently proposed by Cocke, Kasami (1965), and Younger (1967)). The recognition complexity result that we can prove depends on the availability of a polynomial form-restriction on derivations. Polynomial recognition for the whole class of UVG is shown by Satta (1993) by using results from linear programming. Unfortunately, we have not been able to obtain a similar general result for {}-LIG.

Theorem 12 *Let q be a polynomial over the integers. Each language in $\mathcal{L}^q_{\mathrm{fR}}(\{\}\text{-LIG})$ can be recognized in polynomial deterministic time.*

Proof. Let $L \in \mathcal{L}^q_{\mathrm{fR}}(\{\}\text{-LIG})$, $L = L^q_{\mathrm{fR}}(G)$ for some $G \in \{\}\text{-LIG}$. We assume that G is in ETF (see Definition 11, page 76). Let $w = a_1 a_2 \cdots a_n$, $n \geq 0$, be an input string over V_{T}, where $w = \varepsilon$ for $n = 0$. We use sets $t_{i,j}$, $0 \leq i \leq j \leq n$ to represent successful recognition of substrings $a_i \cdots a_j$. These sets are initialized to the empty set. We apply to w the deterministic procedure specified in Figure 4.5.

First of all, we argue that the procedure always stops in a number of steps bounded by some polynomial in n. Steps 1 and 2 are executed $O(\max\{n, 1\})$ times. Because of the polynomial bound on the norm of I-counters, the main for-loop is executed at most polynomially many times. More precisely, the number of executions is bounded by $n^2|V_{\mathrm{N}}|(q(n))^{|V_{\mathrm{I}}|}$. The processing of each element (A, γ) requires a number of executions of step 3 bounded by a constant ($|P|$), and a number of executions of steps 4 and 5 bounded by a polynomial in n, namely $n|V_{\mathrm{N}}|(q(n))^{|V_{\mathrm{I}}|}$. Finally, each test in the for-cycles and in the final if-statement can be carried out in polynomial time.

Input. $w = a_1 a_2 \cdots a_n$, $n \geq 0$.
Output. accept/reject.
Method.

begin
    for every $i \in \{0..n\}$ **do**
        for every $p : As \longrightarrow \varepsilon$ in P **do**
1. $t_{i,i} := t_{i,i} \cup \{(A, \gamma_s)\}$
    for every $i \in \{1..n\}$ **do**
        for every $p : As \longrightarrow a_i$ in P **do**
2. $t_{i-1,i} := t_{i-1,i} \cup \{(A, \gamma_s)\}$
    for every (A, γ) added to some $t_{i,j}$, $\|\gamma\| < q(n)$, **do**
        for every $p : Bt \longrightarrow As$ in P such that $\gamma_s \leq \gamma$ **do**
3. $t_{i,j} := t_{i,j} \cup \{(B, \gamma - \gamma_s + \gamma_t)\}$
        for every $p : Bt \longrightarrow CuAs$ in P and (C, γ') in some $t_{i',i}$
                    such that $\gamma_s \leq \gamma$ and $\gamma_u \leq \gamma'$ **do**
4. $t_{i',j} := t_{i',j} \cup \{(B, \gamma - \gamma_s + \gamma' - \gamma_u + \gamma_t)\}$
        for every $p : Bt \longrightarrow AsCu$ in P and (C, γ') in some $t_{j,j'}$
                    such that $\gamma_s \leq \gamma$ and $\gamma_u \leq \gamma'$ **do**
5. $t_{i,j'} := t_{i,j'} \cup \{(B, \gamma - \gamma_s + \gamma' - \gamma_u + \gamma_t)\}$
    if $(S, \gamma_0) \in t_{0,n}$ **then** accept **else** reject
end.

Figure 4.5: Deterministic procedure for the recognition of $L(G)$; q is the polynomial associated with G .

Second, we claim that after the execution of the procedure, $(A, \gamma) \in t_{i,j}$ if and only if there exists a derivation π such that $At \stackrel{\pi}{\Longrightarrow}_G a_{i+1} \cdots a_j$ with $\#_i(t) = \gamma_i$ for $i \in V_{\mathrm{I}}$. The "only-if" part of the claim can be proved by induction on the order in which (A, γ) has been inserted in $t_{i,j}$; the "if" part of the claim can be proved by induction on the length of π. The correctness of the procedure in Figure 4.5 as a recognizer directly follows from our claim and the if test at the end of the procedure itself. ■

The simple procedure in Figure 4.5 can be enriched in such a way that each element (A, γ) in $t_{i,j}$ is associated with a list of pointers to the sets that caused (A, γ) itself to be inserted in $t_{i,j}$. This results in a parse forest representation for all derivations recognized by the procedure. It is not difficult to show that the resulting method does not affect the complexity analysis.

Observe that if $q \in O(1)$, then the algorithm runs in $O(n^3)$ time. This is to be expected, since in that case $L^q_{\mathrm{fR}}(G)$ can be generated by a context-free language.

The following two corollaries follow immediately from Theorem 12.

Corollary 13 *Each linearly restricted* {}-LIG *can be recognized in polynomial deterministic time.*

Corollary 14 *Each safe* {}-LIG *can be recognized in polynomial deterministic time.*

4.4 Unordered Vector Grammars with Dominance Links (UVG-DL)

4.4.1 Definition

Unordered vector grammars with dominance links (UVG-DL) are an extension of unordered vector grammars (UVG). As in UVG, several context-free string rewriting rules are grouped into vectors. In a derivation, all or no rules from a given instance of a vector must be used. Put differently, all productions from a given vector must be used the same number of times. They can be applied in any order and need not be applied simultaneously or one right after the other (contiguously). In addition, UVG-DL has "dominance links". A non-terminal in the right-hand side of a rule can be linked to the left-hand nonterminal of another rule in the same set. This link will act as a constraint on derivations: if the first rule is used in a derivation, then the second rule must be used subsequently in the subderivation that starts with the linked fnonterminal introduced by the first rule. Put differently, the link must become a relation of (not necessarily immediate) dominance in the derivation tree.

We now formally define UVG-DL. The definition differs from that of UVG (Definition 4, page 63) only in that vectors are equipped with dominance relations and that there is an additional condition on derivations. Note that the definition refers to the notion of derivation tree of a UVG, introduced in Section 4.2.5 (page 64). Recall that we have adopted the convention of identifying derivations with their unique derivation tree.[11]

Definition 14 *An* **Unordered Vector Grammar with Dominance Links** *(*UVG-DL*) is a 4-tuple (V_N, V_T, V, S), where V_N and V_T are sets of nonterminals and terminals, respectively, S is the start symbol, and V is a set of vectors of context-free productions equipped with dominance links. For a given vector $v \in V$, the dominance links form a binary relation dom_v over the set of occurrences of non-terminals in the productions of v such that if $\mathsf{dom}_v(A, B)$, then A (an instance of a symbol) occurs in the right-hand side of some production in v, and B is the left-hand symbol (instance) of some production in v.*

If G is a UVG-DL, $L(G)$ consists of all words $w \in V_T^*$ which have a derivation ϱ of the form

$$S \stackrel{p_1}{\Longrightarrow} w_1 \stackrel{p_2}{\Longrightarrow} w_2 \ldots w_{r-1} \stackrel{p_r}{\Longrightarrow} w_r = w,$$

such that ϱ meets the following two conditions:

1. $p_1 p_2 \ldots p_r \in \mathsf{Perm}(V^*)$
2. *The dominance relations of V, when interpreted as the standard dominance relation defined on trees, hold in the derivation tree of ϱ.*

[11] The definition of UVG-DL could be given without reference to the notion of derivation tree, but the definition we give is more intuitive.

More verbosely, the second condition can be formulated as follows: if v in V contributes the productions (instances) p_1 and p_2 (and perhaps others), and the nth nonterminal in the right-hand side of p_1 dominates the left-hand nonterminal of p_2, then in the context-free derivation tree associated with ϱ (the unique node associated with) the nth daughter node of p_1 dominates (the unique node associated with) p_2.

Derivation trees for UVG-DLs are defined as for UVGs (page 64).

Note that we can define vectors with circular dominance links, and vectors in which two different (instances of) nonterminals in the right-hand side of a production dominate the same left-hand side nonterminal. Neither vector can ever be used in a derivation, and therefore a grammar containing such vectors is equivalent to one in which these vectors have been removed. We will therefore assume from now on without further comment that vectors in UVG-DLs do not contain aberrant dominance links of the two types just discussed.

We can extend the definition of UVG-DL by allowing non-terminal nodes to be equipped with bounded feature structures. A node can only be rewritten if feature structures unify (under the usual definition of unification). Since the feature structures are bounded, they do not extend the formal power of the formalism. We omit the details of the definition here, since they are obvious.

We now present two examples, one formal and one linguistic, which correspond to Example 2, page 68 and Example 4, page 70, respectively.

EXAMPLE 7

We give a UVG-DL that derives the language (COUNT-3)*, where COUNT-3 = $\{a^n b^n c^n \mid n \in \mathsf{N}\}$.

Let $G_2 = (V_\mathrm{N}, V_\mathrm{T}, V, S)$ with:

$$\begin{array}{lcl} V_\mathrm{N} & = & \{S, T, A, B, C\} \\ V_\mathrm{T} & = & \{\mathrm{a,b,c}\} \\ V & = & \{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8\} \end{array}$$

where

$$\begin{array}{l} v_1\colon \{ \quad (S \longrightarrow TS) \quad \} \\ v_2\colon \{ \quad (S \longrightarrow T) \quad \} \\ v_3\colon \{ \quad (A^{(1)} \longrightarrow aA^{(2)}), (B^{(1)} \longrightarrow bB^{(2)}), (C^{(1)} \longrightarrow cC^{(2)}) \quad \} \\ \qquad \mathsf{dom}_{v_3} = \{(A^{(2)}, B^{(1)}), (B^{(2)}, C^{(1)})\} \\ v_4\colon \{ \quad (T \longrightarrow A) \quad \} \\ v_5\colon \{ \quad (A \longrightarrow B) \quad \} \\ v_6\colon \{ \quad (B \longrightarrow C) \quad \} \\ v_7\colon \{ \quad (C \longrightarrow \varepsilon) \quad \} \end{array}$$

and $\mathsf{dom}_{v_1} = \mathsf{dom}_{v_2} = \mathsf{dom}_{v_4} = \mathsf{dom}_{v_5} = \mathsf{dom}_{v_6} = \mathsf{dom}_{v_7} = \emptyset$.

Note that the superscripts on nonterminals only serve to differentiate instances of nonterminals; they are not part of nonterminals themselves. □

Without the dominance links, there would be no way of making sure that a terminal generated by vector v_3 appears in the same $a^n b^n c^n$ sequence as the other terminals generated by v_3. (Recall

that UVG is not closed under Kleene star.)

EXAMPLE 8

Like {}-LIG G_4 of Example 4, page 70, but unlike the UVG given in Example 1, page 64, the following UVG-DL avoids the problem of rightward movement.

Let $G_8 = (V_{\mathrm{N}}, V_{\mathrm{T}}, V, \mathrm{S}')$ with:

$$\begin{array}{lcl} V_{\mathrm{N}} & = & \{\mathrm{S}', \mathrm{VP}, \mathrm{NP}_{\mathrm{nom}}, \mathrm{NP}_{\mathrm{dat}}, \mathrm{NP}_{\mathrm{acc}}\} \\ V_{\mathrm{T}} & = & \{\textit{daß}, \textit{verspricht}, \textit{zu versuchen}, \textit{zu verschrotten}, \\ & & \quad \textit{der Meister}, \textit{niemandem}, \textit{den Kühlschrank}\}^{12} \\ V & = & \{v_1, v_2, v_3, v_4, v_5, v_6, v_7\} \end{array}$$

where

$$\begin{array}{ll} v_1\colon \{ & (\mathrm{S}' \longrightarrow \textit{daß}\ \mathrm{VP})\ \} \\ v_2\colon \{ & (\mathrm{VP}^{(1)} \longrightarrow \mathrm{NP}_{\mathrm{nom}}\ \mathrm{VP}^{(2)}), (\mathrm{VP}^{(3)} \longrightarrow \mathrm{NP}_{\mathrm{dat}}\ \mathrm{VP}^{(4)}), (\mathrm{VP}^{(5)} \longrightarrow \mathrm{VP}^{(6)}\ \mathrm{VP}^{(7)}), \\ & (\mathrm{VP}^{(8)} \longrightarrow \textit{verspricht})\ \} \\ & \mathsf{dom}_{v_2} = \{(\mathrm{VP}^{(2)}, \mathrm{VP}^{(8)}), (\mathrm{VP}^{(4)}, \mathrm{VP}^{(8)}), (\mathrm{VP}^{(7)}, \mathrm{VP}^{(8)})\} \\ v_3\colon \{ & (\mathrm{VP}^{(1)} \longrightarrow \mathrm{VP}^{(1)}\ \mathrm{VP}^{(2)}), (\mathrm{VP}^{(3)} \longrightarrow \textit{zu versuchen})\ \} \\ & \mathsf{dom}_{v_3} = \{(\mathrm{VP}^{(2)}, \mathrm{VP}^{(3)})\} \\ v_4\colon \{ & (\mathrm{VP}^{(1)} \longrightarrow \mathrm{NP}_{\mathrm{acc}}\ \mathrm{VP}^{(2)}), (\mathrm{VP}^{(3)} \longrightarrow \textit{zu verschrotten})\ \} \\ & \mathsf{dom}_{v_4} = \{(\mathrm{VP}^{(2)}, \mathrm{VP}^{(3)})\} \\ v_5\colon \{ & (\mathrm{NP}_{\mathrm{nom}} \longrightarrow \textit{der Meister})\ \} \\ v_6\colon \{ & (\mathrm{NP}_{\mathrm{dat}} \longrightarrow \textit{niemandem})\ \} \\ v_7\colon \{ & (\mathrm{NP}_{\mathrm{acc}} \longrightarrow \textit{den Kühlschrank})\ \} \end{array}$$

and $\mathsf{dom}_{v_1} = \mathsf{dom}_{v_5} = \mathsf{dom}_{v_6} = \mathsf{dom}_{v_7} = \emptyset$.

A derivation is shown in Figure 4.6; for clarity, we have also given the label of the left-hand nonterminal in the nodes of the derivation tree, and have added the terminal symbols. The dominance links are indicated as dotted lines. □

We will now introduce some more terminology with which to describe the derivations of UVG-DLs. If two productions $p_{v,1}$ and $p_{v,2}$ from vector v are linked by a dominance link from a right-hand side nonterminal of $p_{v,1}$ to the left-hand nonterminal $p_{v,2}$, then we will denote this link by $l_{v,1,2}$. (This notation is felicitous since we are assuming that there is only one dominance link from $p_{v,1}$ to $p_{v,2}$.) We will say that $p_{v,1}$ (or its kth right-hand side nonterminal) has a *passive dominance requirement* of $l_{v,1,2}$, and that $p_{v,2}$ has an *active dominance requirement* of $l_{v,1,2}$. If $p_{v,1}$ or $p_{v,2}$ is used in a partial derivation such that the other production is not used in the derivation, the dominance requirement (passive or active) will be called *unfulfilled*. Let ϱ be a (partial) derivation. We associate with ϱ a multiset which represent all the unfulfilled active dominance requirements of ϱ, written $\top(\varrho)$.

We can now define a variant of UVG-DL that includes integrity.

Definition 15 *An* **unordered vector grammar with dominance links and integrity** *or*

[12] Gloss (in order): that, promises, to try, to scrap, the master, no-one, the refrigerator.

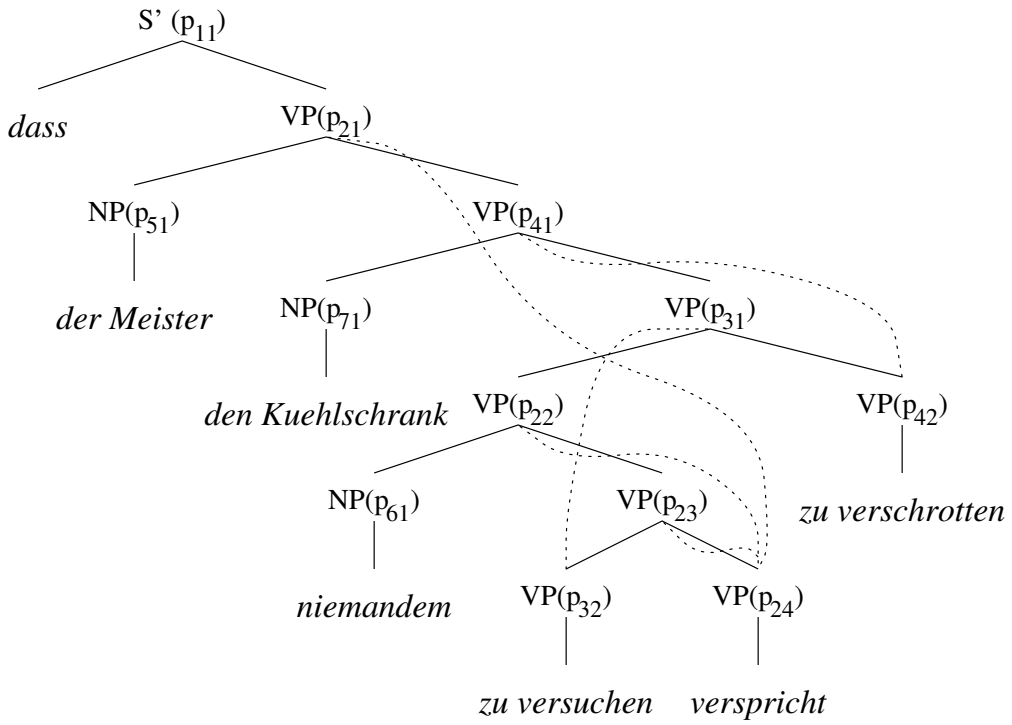


Figure 4.6: Sample derivation for grammar G_8

UVG-DL-Δ *is a* UVG-DL *in which the left-hand symbol in a production can be marked with the integrity constraint, written* Δ*. Let* ϱ *be a derivation in a* UVG-DL-Δ*, and* ϱ' *a subderivation of* ϱ *that starts with a production whose left-hand side is marked with the integrity constraint. Then there may be no unfulfilled active dominance requirements associated with* ϱ'*.*

Observe that the restriction on derivations in UVG-DL-Δ could also be formulated in terms of the linear order of the derivation: a production in which the left-hand symbol is marked with the integrity constraint may not be used in a derivation such that a nonterminal in the right-hand side of a production used before it in the derivation dominates the left-hand nonterminal in a production used after it.

As in the case of {}-LIG, we will call a production whose left-hand side symbol is marked Δ *closed*, and we will also call a set of nonterminal symbols a *set of barriers* if all productions whose left-hand side is a nonterminal from this set are closed.

We now formally introduce the notion of *lexicalization*, which will play an important role both in the mathematical exploration of these formalisms, and in the linguistic uses to which we put them.

Definition 16 *A* UVG-DL *or* UVG-DL-Δ *is* **lexicalized** *(*UVG-DL$_{\text{Lex}}$ *or* UVG-DL-Δ_{Lex}*) if every vector of productions in* V *generates at least one terminal symbol.*

4.4.2 Normal Form

The definition that we have given for UVG-DL does not put any requirements on the dominance relations in a vector. For example, there is no requirement that every production in a vector be connected to another by a dominance link, or that all productions of a vector be interconnected. In this section, we define a version of UVG-DL in which the productions of a vector must be interconnected in such a way that any production can be "reached" from any other production via the dominance links (in any direction). We show that this requirement does not increase the generative power. We also introduce an extended two-form which we will need later when we discuss parsing.

Definition 17 *A* **strict** UVG-DL (UVG-DL-Δ) *is a* UVG-DL *(*UVG-DL-Δ*) in which no vector can be partitioned into two non-empty vectors while preserving the dominance relation associated with the vector.*

Note that for a strict UVG-DL (UVG-DL-Δ) the first requirement from Definition 14 (that the same number of productions from a given vector be used in a derivation) follows from the second requirement (the constraint the dominance links impose on the derivation tree).

We now show that strictness does not alter the weak generative power of the formalism.

Theorem 15 *For every* UVG-DL *(*UVG-DL-Δ*) there is an equivalent strict* UVG-DL.

Proof. Let $G = (V_{\mathrm{N}}, V_{\mathrm{T}}, V, S)$ be a UVG-DL (UVG-DL-Δ). We will construct a strict UVG-DL $G' = (V'_{\mathrm{N}}, V_{\mathrm{T}}, V', S)$ where V' is constructed as follows. Let v_r be in V, $v_r = (p_{r,1} \ldots, p_{r,|v_r|})$ with dominance links dom_{v_r}. Let the left-hand nonterminal of $p_{r,i}$ be A_i, $1 \leq i \leq |v_r|$. Then V' contains the vector[13] $v'_r = (p_{r,1} \ldots, p_{r,|v_r|}, S^{(1)} \longrightarrow S^{(2)})$ with $\mathsf{dom}_{v'_r} = \mathsf{dom}_{v_r} \cup \{(S^{(2)}, A_i) \mid 1 \leq i \leq |v_r|\}$. V' contains no other vectors.

It is easy to prove by induction on i that $S \stackrel{i}{\Longrightarrow}_G \alpha$ iff $S \stackrel{*}{\Longrightarrow}_{G'} S \stackrel{i}{\Longrightarrow}_{G'} \alpha$, where $\alpha \in (V_{\mathrm{N}} \cup V_{\mathrm{T}})^*$. ■

Note that the construction in the proof above preserves lexicalization, since each vector in G' subsumes a vector in G.

We now define a version of Chomsky Normal Form for UVG-DL and UVG-DL-Δ.

Definition 18 *A* UVG-DL *(*UVG-DL-Δ*)* $G = (V_{\mathrm{N}}, V_{\mathrm{T}}, V, S)$ *is in* **Extended Two Form** *(ETF) if every production of every vector in* V *has the form* $A \rightarrow B_1 B_2$, $A \rightarrow B$, *or* $A \rightarrow a$, *where* $A, B_1, B_2 \in V_{\mathrm{N}}$, *and* $a \in V_{\mathrm{T}} \cup \{\varepsilon\}$.

Theorem 16 *For any* UVG-DL *(*UVG-DL-Δ*), there is an equivalent* UVG-DL *(*UVG-DL-Δ*) in ETF.*

[13] Recall that the superscripts on nonterminal symbols refer to occurrences of the symbol in rules; the superscripts are not part of the symbol itself.

Proof. For any UVG-DL (UVG-DL-Δ) $G = (V_{\mathrm{N}}, V_{\mathrm{T}}, V, S)$ we construct an equivalent grammar $G' = (V'_{\mathrm{N}}, V_{\mathrm{T}}, V', S)$ in ETF. Let $V'_{\mathrm{N}} = V_{\mathrm{N}} \cup \{[p,i] | p \in v, somev \in V \text{ and } 1 \leq i \leq \#_{V_{\mathrm{N}} \cup V_{\mathrm{T}}}(\mathsf{rhs}(p))\} \cup \{[a] \mid a \in V_{\mathrm{T}}\}$. The construction of V' consists of two steps:

- For every production p in a vector v of V, replace every occurrence of a terminal symbol a in $\mathsf{rhs}(p)$ by nonterminal $[a]$, add production $([a] \longrightarrow a)$ to v, and add a dominance link between the two new occurrences of nonterminal symbol $[a]$.
- For every production p in a vector v of V such that $\#_{V_{\mathrm{N}} \cup V_{\mathrm{T}}}(\mathsf{rhs}(p)) > 2$, cover p by means of productions $p_1, \ldots, p_q$, $q \geq 2$, which have right-hand side length not greater than 2. This is done using standard techniques for the construction of Chomsky normal form grammars out of context-free grammars, where the indices are "spread out" in the obvious manner. More precisely, if p is $A \longrightarrow B_1 \ldots B_n$, then replace p in v by the following sequence of productions:

$$\begin{aligned} p_1 &: A \longrightarrow B_1[p,1] \\ p_2 &: [p,1] \longrightarrow B_2[p,2] \\ &\vdots \\ p_{n-1} &: [p,n-2] \longrightarrow B_{n-1}B_n \end{aligned}$$

  such that the two occurrences of $[p,i]$, for each $1 \leq n-2$ are linked by dominance links, and the B_i, $1 \leq i \leq n$, retain their dominance links.

If G is a UVG-DL-Δ and p is a closed production, then p_1 is a closed production.

It is not difficult to verify that G' obtained as above has the claimed properties. ■

Observe that the construction of G' preserves lexicalization and strictness.

4.4.3 Formal Properties

Our main result is that UVG-DL is weakly equivalent to {}-LIG. The sets of a {}-LIG implement the dominance links and make sure that all members from one set of rules are used during a derivation. We will prove this result with the aid of two lemmas.

Lemma 17 $\mathcal{L}(\text{UVG-DL}) \subseteq \mathcal{L}(\{\}\text{-LIG})$

Proof. Let $G = (V_{\mathrm{N}}, V_{\mathrm{T}}, V, S)$ be a UVG-DL, where $V = \{v_1, \ldots, v_K\}$ with $v_i = (p_{i,1}, \ldots, p_{i,k_i})$, $k_i = |v_i|$, $1 \leq i \leq K$. We construct a {}-LIG $G' = (V_{\mathrm{N}}, V_{\mathrm{T}}, V_{\mathrm{I}}, P, S)$, which we will define using the multiset definition of {}-LIG (Definition 5′). Let $V_{\mathrm{I}} = \{l_{i,j,k} \mid 1 \leq i \leq K, 1 \leq j,k \leq k_i\}$. Define P as follows.

Let v in V, and let p in v be the production $A \longrightarrow w_0 B_1 w_1 \cdots B_n w_n$ be in v_r. In the following, we will denote by $\top(p)$ the multiset of active dominance requirements of p, and by $\bot_i(p)$ the

multiset of unfulfilled passive dominance requirements of B_i, $1 \leq i \leq n$. Add to P the following production:

$$A\,\top(p) \longrightarrow w_0 B_1 \bot_1(p) w_1 \cdots B_n \bot_n(p) w_n$$

P contains no other productions. We will show by induction that for A in V_{N}, and w in V_{T}^*, we have $A \stackrel{*}{\Longrightarrow}_G w$ iff $A \stackrel{*}{\Longrightarrow}_{G'} w$.

We first show the "only if" direction. Specifically, we show that for all integers k, $\varrho : A \stackrel{k}{\Longrightarrow}_G w$, $w \in V_{\mathrm{T}}^*$, with unfulfilled active dominance requirements $\top(\varrho)$, implies that there is a derivation $A\top(\varrho) \stackrel{*}{\Longrightarrow}_{G'} w$.

The base case ($k = 1$) is straightforward: if we have $\varrho : A \stackrel{1}{\Longrightarrow}_G w$ for $A \in V_{\mathrm{N}}$ and $w \in V_{\mathrm{T}}^*$, then there is a production $p : A \longrightarrow w$ in some vector v such that $\top(p) = \top(\varrho)$. By construction of G' we have a production $p' \in P'$, $p' : A\,\top(p) \longrightarrow w$, and hence we have a derivation $A\,\top(\varrho) \stackrel{1}{\Longrightarrow}_{G'} w$.

For the induction step, assume that we have a derivation $\varrho : A \stackrel{k}{\Longrightarrow}_G w$, and that we use production $p : A \longrightarrow w_0 B_1 w_1 \cdots B_n w_n$ for the first rewrite step. Consider the subderivations of ϱ that start from the nonterminals introduced by p, $\varrho_i : B_i \stackrel{k_i}{\Longrightarrow}_G v_i$, $1 \leq i \leq n$, where $w = w_0 v_1 w_1 \cdots v_n w_n$. Let the unfulfilled active dominance requirements of ϱ be $\top(\varrho)$, and let those of ϱ_i be $\top(\varrho_i)$, for $1 \leq i \leq n$. Then we have the relation $\top(\varrho) = \cup_{i=1}^{n}(\top(\varrho_i) \setminus \bot_i(p)) \cup \top(p)$.

By construction of G' there is a production p' in P of the form $A\top(p) \longrightarrow w_0 B_1 \bot_1(p) w_1 \cdots B_n \bot_n(p) w_n$. Since $k_i < k$ for $1 \leq i \leq n$, by induction hypothesis we have derivations $\varrho_i' : B_i \top(\varrho_i) \stackrel{*}{\Longrightarrow}_{G'} v_i$. By definition of the derivation relation in {}-LIG, we then have a derivation $At \stackrel{*}{\Longrightarrow}_{G'} w$ such that $t = \cup_{i=1}^{n}(\top(\varrho_i) \setminus \bot_i(p)) \cup \top(p)$, and therefore $t = \top(\varrho)$. This concludes the induction step.

We now show the "if" direction. Specifically, we show that for all integers k, $At \stackrel{k}{\Longrightarrow}_{G'} \alpha$, $A \in V_{\mathrm{N}}$, t a multiset of elements of V_{I}, and $\alpha \in V_{\mathrm{T}}^*$, implies that there is a derivation $\varrho : A \stackrel{*}{\Longrightarrow}_G \alpha$ such that $\top(\varrho) = t$.

The base case ($k = 1$) is very similar to the base case in the converse direction.

For the induction step, assume that $At \stackrel{k}{\Longrightarrow}_{G'} w$, and that we use production $p' : As \longrightarrow w_0 B_1 s_1 w_1 \cdots B_n s_n w_n$ for the first rewrite step. We then have subderivations $B_i t_i \stackrel{k_i}{\Longrightarrow}_{G'} v_i$ where t is distributed among $t_1, \ldots, t_n$ in accordance with p', and where $w = w_0 v_1 w_1 \cdots v_n w_n$. Since $k_i < k$ for $1 \leq i \leq n$, by induction hypothesis we have derivations $\varrho_i : B_i \stackrel{*}{\Longrightarrow}_G v_i$ with unfulfilled active dominance requirements of $\top(\varrho_i) = t_i$.

By construction of G' there is a vector in V with a production of the form $p : A \longrightarrow w_0 B_1 w_1 \cdots B_n w_n$ such that $\top(p) = s$ and $\bot_i(p) = s_i$ for $1 \leq i \leq n$. By definition of the derivation relation in UVG-DL, we have a derivation in G $\varrho : A \stackrel{p}{\Longrightarrow}_G w_0 B_1 w_1 \cdots B_n w_n \stackrel{*}{\Longrightarrow}_G w$ with $\top(\varrho) = \cup_{i=1}^{n}(\top(\varrho_i) \setminus \bot_i(p)) \cup \top(p)$, and therefore $t = \top(\varrho)$. This concludes the induction step.

We conclude that $L(G) = L(G')$.

Lemma 18 $\mathcal{L}(\{\}\text{-LIG}) \subseteq \mathcal{L}(\text{UVG-DL})$

Proof. Let $G = (V_{\mathrm{N}}, V_{\mathrm{T}}, V_{\mathrm{I}}, P, S)$ be a $\{\}$-LIG in RINF (Definition 10, page 75). We construct a UVG-DL $G^{'} = (V_{\mathrm{N}}, V_{\mathrm{T}}, V, S)$, where V is defined as follows:

1. If $p \in P$ is a $\{\}$-LIG production of type (i), then $((p), \emptyset) \in V$.
2. If $p \in P$ is a $\{\}$-LIG production of type (ii), with $p = A \longrightarrow Bf$ for $A, B \in V_{\mathrm{N}}$, $f \in V_{\mathrm{I}}$, then for all $q \in P$ such that $q = Cf \longrightarrow D$, $v = ((A \longrightarrow B, C \longrightarrow D), \mathsf{dom}_v(B, C))$ is in V. We refer to v as an *f-vector*.

We will show by induction that fro all w in V_{T}^*, $S \stackrel{*}{\Longrightarrow}_G w$ iff $S \stackrel{*}{\Longrightarrow}_{G'} w$.

We first show the "only if" direction. Specifically, we show that for all integers k, for all $\{\}$-LIGs G and for all UVG-DLs G' as constructed above, if there is a derivation $\varrho : S\,\{\} \stackrel{*}{\Longrightarrow}_G w$ with k instances of applications of rules of RINF type (ii), then there is a derivation $\varrho' : S \stackrel{*}{\Longrightarrow}_{G'} w$ such that ϱ and ϱ' are identical except for the index symbols in ϱ.

The base case ($k = 0$) is straightforward. Let G be a $\{\}$-LIG. Since all productions used in ϱ must be of type (i), ϱ is also a legal derivation in G'.

For the induction step, let G be a $\{\}$-LIG, and let G' be the UVG-DL constructed from G by the method given above. Furthermore, let ϱ be a derivation of the form $S \stackrel{*}{\Longrightarrow}_G \alpha_1 A t_1 \alpha_2 \stackrel{p}{\Longrightarrow}_G \alpha_1 B (t_1 \cup \{f\}) \alpha_2 \stackrel{*}{\Longrightarrow}_G \alpha_3 C (t_2 \cup \{f\}) \alpha_4 \stackrel{q}{\Longrightarrow}_G \alpha_3 D t_2 \alpha_4 \stackrel{*}{\Longrightarrow}_G w$, where p, q are in P with $p : A \longrightarrow Bf$ and $q : Cf \longrightarrow D$. Assume that ϱ includes k instances of applications of rules of type (ii) (at least one of which is an application of p). Now construct a new $\{\}$-LIG G_- which is identical to G except that P also includes rules $p_- : A \longrightarrow B$ and $q_- : C \longrightarrow D$. Clearly, we have a derivation $\varrho_- : S \stackrel{*}{\Longrightarrow}_{G_-} \alpha_1 A t_1 \alpha_2 \stackrel{p_-}{\Longrightarrow}_{G_-} \alpha_1 B t_1 \alpha_2 \stackrel{*}{\Longrightarrow}_{G_-} \alpha_3 C t_2 \alpha_4 \stackrel{q_-}{\Longrightarrow}_{G_-} \alpha_3 D t_2 \alpha_4 \stackrel{*}{\Longrightarrow}_{G_-} w$ which includes $k - 1$ instances of applications of rules of type (ii). By induction hypothesis, there is a UVG-DL G'_- derived from G_- using the construction given above such that there is a derivation $\varrho'_- : S \stackrel{*}{\Longrightarrow}_{G'_-} w$, and, furthermore, such that ϱ_- and ϱ'_- are identical except for the index symbols in ϱ_-. But ϱ'_- is also a legal derivation in G' (since rule q is used to rewrite a nonterminal in a sentential subform derived from the application of rule p). Since ϱ_- differs from ϱ only in terms of the indices, we have that ϱ'_- is identical to ϱ except for the indices in the latter. This concludes the induction step.

We now show the "if" direction. Specifically, we show that for all integers k, for all $\{\}$-LIGs G and for all UVG-DLs G' as constructed above, if there is a derivation $\varrho' : S\{\} \stackrel{k}{\Longrightarrow}_{G'} w$ with k instances of applications of rules from vectors with two elements, then there is a derivation $\varrho : S \stackrel{*}{\Longrightarrow}_G w$ such that ϱ and ϱ' are identical except for the index symbols in ϱ.

The base case ($k = 0$) is straightforward. Let G' be a UVG-DL. Since all productions used in ϱ' must be derived from productions of G of RINF type (i), ϱ' is also a legal derivation in G.

For the induction step, let G' be a UVG-DL, and let ϱ' be a derivation of the form $S \stackrel{*}{\Longrightarrow}_{G'} \alpha_1 A \alpha_2 \stackrel{p}{\Longrightarrow}_{G'} \alpha_1 B \alpha_2 \stackrel{*}{\Longrightarrow}_{G'} \alpha_3 C \alpha_4 \stackrel{q}{\Longrightarrow}_{G'} \alpha_3 D \alpha_4 \stackrel{*}{\Longrightarrow}_{G'} w$, where $v = ((p, q), \mathsf{dom}_v)$ is in V with $p : A \longrightarrow B$, $q : C \longrightarrow D$, and $\mathsf{dom}_v = (B, C)$. We will call these two applications of p and q "featured".

Assume that ϱ' includes k instances of applications of vectors of length two (one of which is the featured application of v). Now construct a new UVG-DL G'_- which is identical to G' except that V also includes vectors $v^p_- : ((p), \emptyset)$ and $v^q_- : ((q), \emptyset)$. Clearly, we have a derivation ϱ'_- in G'_- which is identical to ϱ' but which includes $k-1$ instances of applications of vectors of length two. Let G be the $\{\}$-LIG from which G' is constructed. By the construction of G', G must contain the productions $p : A \longrightarrow Bf$ and $q : Cf \longrightarrow D$ for some $f \in V_{\mathrm{I}}$. We now construct a new $\{\}$-LIG G_- which is identical to G except that P also includes rules $p_- : A \longrightarrow B$ and $q_- : C \longrightarrow D$. Clearly, the construction given above will yield G'_- when applied to G_-. By the induction hypothesis, there is a derivation $\varrho_- : S \stackrel{*}{\Longrightarrow}_{G_-} w$, and, furthermore, ϱ_- is identical to ϱ'_- (and hence to ϱ') except for the index symbols in ϱ_-. We can transform ϱ_- into a derivation in G by letting the featured application of p introduce an index symbol f and letting the featured application of q consume it again. Thus we have shown that there is a derivation in G of w, and furthermore, that it is identical to ϱ' except for the indices. This concludes the induction step.

We conclude that $L(G) = L(G^{'})$. ■

From the two lemmas above and the observation that we obtain the following theorems:

Theorem 19 $\mathcal{L}$(UVG-DL) = $\mathcal{L}$($\{\}$-LIG)

Theorem 20 $\mathcal{L}$(UVG-DL-Δ) = $\mathcal{L}$($\{\}$-LIG-Δ)

Proof. It can easily be seen that the proofs of Lemma 17 and 18 preserve integrity. ■

Theorem 21 $\mathcal{L}$(UVG) *is properly included in* $\mathcal{L}$(UVG-DL).

Proof. Inclusion follows trivially from the definitions of UVG and UVG-DL, while the properness of the inclusion follows from the fact that $\mathcal{L}$(UVG-DL) = $\mathcal{L}$($\{\}$-LIG) is closed under Kleene-star by Theorem 8, while UVG is not; specifically, $\{a^n b^n c^n \mid n \geq 1\}^*$ is not in $\mathcal{L}$(UVG) (Dassow and Păun, 1989, Exercise 2.1.4, p.113). ■

From the proof of Theorem 19 it can easily be seen that a lexicalized UVG-DL (UVG-DL-Δ) translates into a restricted $\{\}$-LIG ($\{\}$-LIG-Δ). We can therefore immediately apply Theorems 6 and 12 to obtain the following results:

Corollary 22 $\mathcal{L}(\text{UVG-DL}_{\text{Lex}}) \subseteq \mathcal{L}$(CSG) *and* $\mathcal{L}(\text{UVG-DL-}\Delta_{\text{Lex}}) \subseteq \mathcal{L}$(CSG)

Corollary 23 *Any language in* $\mathcal{L}(\text{UVG-DL}_{\text{Lex}})$ *or* $\mathcal{L}(\text{UVG-DL-}\Delta_{\text{Lex}})$ *is parsable in polynomial deterministic time.*

In the following section, we will present a parser specifically for UVG-DL.

4.4.4 Parsing UVG-DL

While we know that certain forms of UVG-DL are polynomially parsable from the equivalence with restricted forms of {}-LIG, we have not given a parsing algorithm for the formalism. In this section, we give a parsing algorithm for strict UVG-DL, which has properties that make it an appealing algorithm for linguistic applications. We define a $\top$-*counter* as a three-dimensional array whose elements are non-negative integers and are addressed by a primary index v ranging in V and secondary and tertiary indices ranging in the multiset $\{p_1, \ldots, p_{|v|}\}$, where $v = (p_1, \ldots, p_{|v|})$. In what follows, each derivation ϱ of the kind $A \stackrel{*}{\Longrightarrow}_G w$, $A \in V_{\mathrm{N}}$ and $w \in V_{\mathrm{T}}^*$, will be associated with a $\top$-counter γ which represents the multiset of unfulfilled active dominance requirements for ϱ. Specifically, $\gamma(v, p_1, p_2)$ is the number of unfulfilled active dominance requirements of the form l_{v,p_1,p_2}.[14] $\gamma_v^{(p_1,p_2)}$ denotes the $\top$-counter with value 1 at (v, p_1, p_2) and null everywhere else. Let $\gamma_\top = \sum_{l_{v,p_1,p_2} \in \top} \gamma_v^{(p_1,p_2)}$. $\gamma_\emptyset$ represents the $\top$-counter with value zero everywhere. The sum of $\top$-counters is defined in the obvious way. The following norm will be used for v-counters:

$$\|\gamma\| \quad = \quad \sum_{v \in V} \sum_{p_1 \in v} \sum_{p_2 \in v} \gamma(v, p_1, p_2). \tag{4.1}$$

The algorithm is shown in Figure 4.7. Grammar G is assumed to be in ETF (Definition 18, page 89).

The algorithm iterates CKY parses, allowing an increased number of unfulfilled dominance links on each pass. Like CKY, the algorithm uses a dynamic programming technique to represent partial parses, which are recorded in the two-dimensional table t. Since we must allow ε-productions, the indices of t represent positions between input symbols. If the input is $a_1 \cdots a_n$, the indices range from 0 to n. An entry of (A, γ) in $t_{i,j}$ represents the fact that there is a derivation $A \stackrel{*}{\Longrightarrow}_G a_{i+1} \cdots a_j$ with unfulfilled dominance relations γ. When a production is used (in CKY fashion), the associated $\top$-*counter* must be updated as well. The production may fulfill dominance link requirements of its daughter(s), and/or it may introduce new ones. (If the production is marked with the integrity constraint, then it must fulfill all unfulfilled dominance constraints of its daughters. For simplicity, this is not shown in the algorithm.) The algorithm then makes successive parses of the input string, using only those entries (A, γ) in t for which $\|\gamma\|$ is less than $k = 0, 1, 2, \ldots$. At least one valid parse has been found if $t_{0,n}$ contains the entry $(S, \gamma_\emptyset)$. The algorithm can be augmented by equipping the entries in t with backpointers, which will result in a compact representation of the parse forest (with $(S, \gamma_\emptyset)$ as its root).

The complexity of this algorithm remains unchanged over the parsing algorithm given for {}-LIG. Steps 1 and 2 are each performed $O(n)$ times. For steps 3 and 4, the additional complexity over the context-free CKY algorithm stems from the fact that the number of entries in each field of the table has increased to $O(|V_{\mathrm{T}}|q(n)^l)$, and the overall processing of the l-loop therefore takes time $O(q(n)^{L_G})$, where L_G is the total number of dominance links in V.

[14] Recall that we are assuming without loss of generality that grammars with two distinct dominance links between a single pair of productions of a vector are excluded.

Input. $w = a_1 a_2 \cdots a_n$, $n \geq 0$.
Parameter Function $q(n)$
Output. accept/reject.
Method.

begin
  for every $i \in \{0..n\}$ **do**
    for every $p : A \longrightarrow \varepsilon$ in some v with unfulfilled active dominance requirements $\top(p)$ **do**
1. $t_{i,i} := t_{i,i} \cup \{(A, \gamma_{\top(p)})\}$
  for every $i \in \{1..n\}$ **do**
    for every $p : A \longrightarrow a_i$ in some v with unfulfilled active dominance requirements $\top(p)$ **do**
2. $t_{i-1,i} := t_{i-1,i} \cup \{(A, \gamma_{\top(p)})\}$
  for $l := 0$ to $q(n)$ **do**
    for $i := 0$ to n **do**
      for $j := 0$ to $n - i$ **do**
        for every (B, γ) in $t_{i+j,j}$ such that $\|\gamma\| \leq l$ and such that there is a $p : A \longrightarrow B$ in a vector $v \in V$ with active dominance requirements $\top_A$ and passive dominance requirements $\bot_B$, with $\gamma_{\bot_B} \leq \gamma$ **do**
3. $t_{i+j,j} := t_{i+j,j} \cup \{(A, \gamma - \gamma_{\bot_B} + \gamma_{\top_A})\}$
        for $k := 0$ to i **do**
          for every (B, γ) in $t_{j,j+k}$ and every (C, γ') in $t_{j+k,i+j}$ such that $\|\gamma\|, \|\gamma'\| \leq l$ and such that there is a $p : A \longrightarrow BC$ in a vector $v \in V$ with active dominance requirements $\top_A$ and such that B has passive dominance requirements $\bot_B$, and C, $\bot_C$, with $\gamma_{\bot_B} \leq \gamma$ and $\gamma_{\bot_C} \leq \gamma'$ **do**
4. $t_{j,i+j} := t_{j,i+j} \cup \{(A, \gamma - \gamma_{\bot_B} + \gamma' - \gamma_{\bot_C} + \gamma_{\top_A})\}$
    if $(S, \gamma_0) \in t_{0,n}$ **then** accept
  if $(S, \gamma_0) \notin t_{0,n}$ **then** reject
end.

Figure 4.7: Deterministic procedure for the recognition of unordered vector languages with dominance links; q is the polynomial associated with G .

This algorithm has the advantage over the one given in Figure 4.5 for {}-LIG that it will first try to find a parse with the least "stretching" of dominance links. As such, it is presumably of interest for natural language applications, in which it can plausible be argued that the use of a powerful formalism should only be an option of last resort. For example, one could let the algorithm stop as soon as the first entry in $t_{0,n}$ of the form $(S, \gamma_\emptyset)$ is found. This approach would make sense if it could be convincingly argued that in cases of syntactic ambiguity arising from different assignment of dominance links, the parse involving more dominance links is always strongly dis-preferred. Standard examples of syntactic ambiguity, such as PP-attachment, would remain unaffected. The two parses found correspond to the two analyses:

(1) a. ...daß der Meister niemandem [PRO dem Lehrling zu helfen] verspricht
...that [the master]$_{\text{NOM}}$ no-one$_{\text{DAT}}$ [the apprentice]$_{\text{DAT}}$ to help promises

...that the master has promised no-one to help the apprentice

b. ...daß der Meister niemandem$_i$ dem Lehrling [PRO t$_i$ zu helfen]
...that [the master]$_{\mathrm{NOM}}$ no-one$_{\mathrm{DAT}}$ [the apprentice]$_{\mathrm{DAT}}$ to help
verspricht
promises

...that the master has promised the apprentice to help no-one

(1a) is greatly preferred, as would be predicted by the proposed "first-parse" algorithm.

4.5 Excursus: The BEPDA

In this section, we briefly depart from the study of formalisms that extend the formal power of tree adjoining grammar and discuss the bottom-up embedded push-down automaton (BEPDA), which is an automaton for simple TAG. There are two reasons to discuss it here: first, while it was fully specified by Schabes (1990), it was not actually formally defined there, and we will subsequently require a formal definition for proofs. Second, giving the definition of the simpler BEPDA first will motivate the definition of the {}-BEPDA we give subsequently.

4.5.1 BEPDA: Definition and Formal Properties

A formal automaton for a particular grammar formalism can simulate a derivation in that grammar formalism in two ways: top-down or bottom-up. In top-down mode, the rules of the automaton are such that it (non-deterministically) guesses the final derived tree from the root down. When a terminal symbol is predicted by the top-down automaton, it is matched against the input string. In a bottom-up mode, the input word is read onto the stack incrementally, and the automaton guesses (non-deterministically) the derived tree in a bottom-up manner. In the case of the push-down automaton (PDA), which is the formal automaton equivalent to context-free grammars, the difference between a top-down and a bottom-up PDA only lies in the way the particular automaton is specified (given a CFG), not in how the formal automaton is defined *qua* formal system. This is different in the case of the embedded pushdown automaton (EPDA), an automaton introduced by Vijay-Shanker (1987) that is equivalent to TAG. EPDA simulates a top-down derivation, and we cannot use this automaton to simulate a bottom-up derivation. (The problem is related to the simulation of adjunction.) We therefore need to define a new automaton, called the bottom-up EPDA, or BEPDA. The BEPDA was first proposed and specified by Schabes (1990) and Schabes and Vijay-Shanker (1990) as the "dual" of the EPDA , but it was not formally defined there. We add a formal definition here, using their specification and the definition of the EPDA in (Vijay-Shanker, 1987).

For each move of the EPDA there is a corresponding move of the BEPDA which does exactly the inverse. The moves of the EPDA as defined by Vijay-Shanker (1987) are the WRAP move and the POP move:

1. In a WRAP move, the top symbol of the top stack is popped and replaced by a sequence of (possibly empty) stack symbols. Furthermore, finite (and possibly empty) sequences of

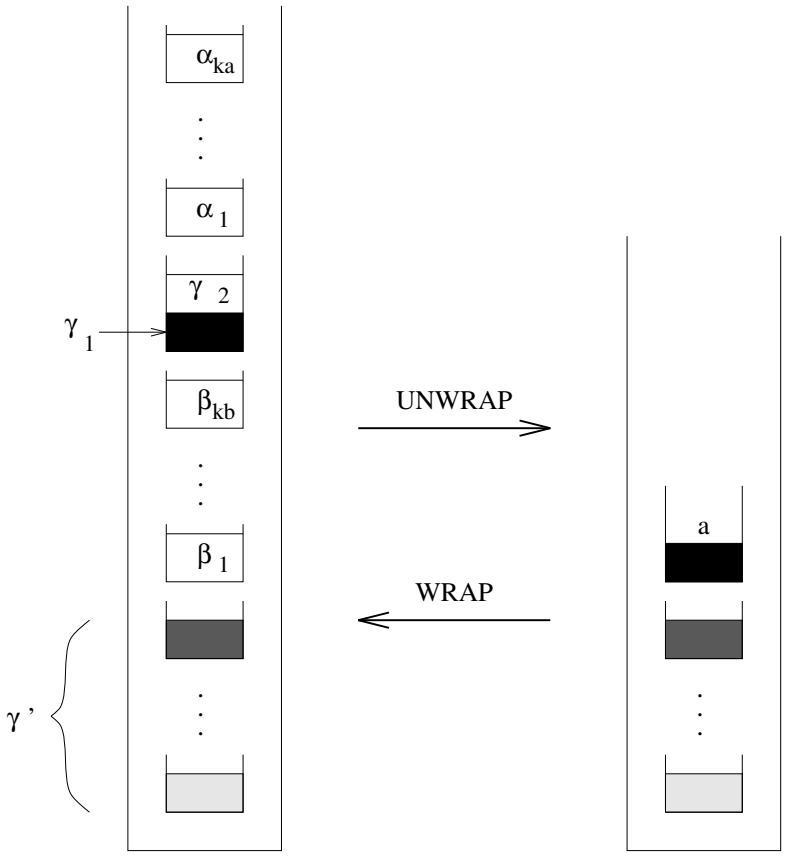


Figure 4.8: UNWRAP (BEPDA) and WRAP (EPDA) Moves

stacks are inserted immediately below and above the top stack.

2. In a POP move, an empty stack is popped from the top of the stack of stacks.

For the BEPDA, we get the following two moves:

1. In an UNWRAP move, finite (and possibly empty) sequences of stacks are removed immediately below and above (or "unwrapped around") a designated stack (which becomes the new top stack). Furthermore, a (possibly empty) sequence of stack symbols on the new top stack is popped and replaced by a single new stack symbol.

2. In a PUSH-NEW-STACK move, an empty stack is pushed onto the stack of stacks.

The two pairs of dual moves are illustrated in Figures 4.8 and 4.9, which are closely based on (Schabes and Vijay-Shanker, 1990, Figure 2).[15] On the left of Figure 4.8, the white rectangles correspond to stacks or parts of stacks that are removed (BEPDA) or added (EPDA) to the

[15] In that figure, there is a typographical error: the stack symbol 'a' should be in the "UNWRAP move" figure, not in the "PUSH move" figure.

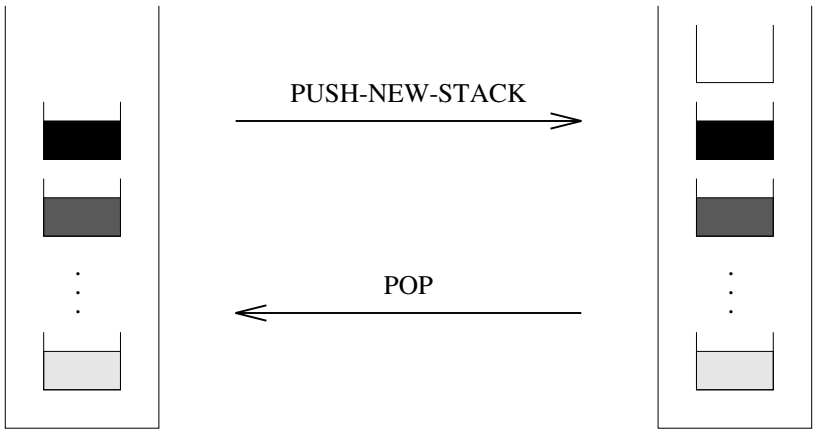


Figure 4.9: PUSH-NEW-STACK (BEPDA) and POP (EPDA) Moves

stack of stacks. Note that the EPDA starts with an initial start symbol and finishes with an empty stack; conversely, the BEPDA starts with an empty stack and finishes with a final stack symbol. We can thus define the BEPDA as follows, closely following the definition of the EPDA in (Vijay-Shanker, 1987) (though we exclude the final sets):

Definition 19 *A* **Bottom-Up Embedded Pushdown Automaton** *(BEPDA) is a 6-tuple* $(Q, \Sigma, \Gamma, \delta, q_0, Z_f)$ *where:*

1. Q *is a finite set of states;*
2. $q_0 \in Q$ *is the start state;*
3. Σ *is a finite set of input symbols;*
4. Γ *is a finite set of stack symbols;*
5. $Z_f \in \Gamma \cup \{\epsilon\}$ *is the final stack symbol, which is on the stack at the end of a run;*
6. δ *is a mapping from* $Q \times \Sigma \times (\lfloor\Gamma^+)^* \times \Gamma^* \times (\lfloor\Gamma^+)^*$ *to finite subsets of* $Q \times (\Gamma \cup \{\lfloor\})$. *The domain 5-tuple corresponds to current state, input symbol, stacks below designated stack to be removed, stack symbols to be removed from designated stack, and stacks above the designated stack to be removed. The range pair corresponds to the new state and the new stack symbol to be pushed on the designated stack.*

Here, '$\lfloor$*' is a symbol not in* Γ *which we will use to represent the bottom of the stack.*

An **instantaneous description** *(ID) for a BEPDA is a "snapshot" of the automaton. It is a member of* $Q \times (\lfloor\Gamma^+)^* \times \Sigma^* \times \Sigma^*$*; its four components represent the current state, the current configuration of the stack of stacks, the part of the input word that has been read, and the part of the input word that has not yet been read, respectively.*

The **move relation** *of a BEPDA is a (reflexive, transitive) relation* $\vdash$ *between two IDs. Let* $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_f)$ *be a BEPDA,* $I_1 = (q, \gamma, w_1, aw_2)$*,* $a \in \Sigma$*, an ID, and* I_2 *also an ID. We have* $I_1 \vdash I_2$ *iff one of the two following conditions obtains:*

1. *The following conditions all hold:*[16]

   (a) $\gamma = \gamma' \lfloor \beta_1 \lfloor \beta_2 \ldots \lfloor \beta_{k_b} \lfloor \gamma_1\gamma_2 \lfloor \alpha_1 \lfloor \alpha_2 \ldots \lfloor \alpha_{k_a}$ *for some* $\gamma' \in (\lfloor \Gamma^+)^*$*, and for* $\beta_i \in \Gamma^*, 1 \leq i \leq k_b$*,* $\alpha_i \in \Gamma^*, 1 \leq i \leq k_a$*,* $\gamma_1\gamma_2 \in \Gamma^+$*;*

   (b) $I_2 = (q', \gamma' \lfloor \gamma_1 Z, w_1 a, w_2)$*, for some* $q' \in Q$ *and* $Z \in \Gamma$*;*

   (c) *and* $(q', Z) \in \delta(q, a, \lfloor \beta_1 \ldots \lfloor \beta_{k_b}, \gamma_2, \lfloor \alpha_1 \ldots \lfloor \alpha_{k_a})$.

   This is the UNWRAP *move.*

2. $I_2 = (q', \gamma \lfloor, w_1, w_2)$ *and* $(q', \lfloor) \in \delta(q, \varepsilon, \varepsilon, \varepsilon, \varepsilon)$.
   This is the PUSH-NEW-STACK *move.*[17]

The **language accepted by** M *(by empty stack) is defined as follows:*

$$N(M) = \{w \mid (q_0, \varepsilon, \varepsilon, w) \overset{*}{\underset{M}{\vdash}} (q, Z_f, w, \varepsilon), q \in Q\}$$

We see that "language accepted by empty stack" is not entirely accurate, since we require the final stack symbol to be present on the stack at the end of the derivation. This reflects the fact that the BEPDA is the dual of the EPDA: an EPDA starts out with an initial symbol in the stack and ends up with an empty stack, whereas the run of a BEPDA goes the reverse direction.

We will occasionally use the following terminology. Suppose that $(q, Z) \in \delta(q, a, \lfloor \beta_1 \ldots \lfloor \beta_{k_b}, \gamma_2, \lfloor \alpha_1 \ldots \lfloor \alpha_{k_a})$. We will say that we *unwrap* $\lfloor \beta_1 \ldots \lfloor \beta_{k_b}, \gamma, \lfloor \alpha_1 \ldots \lfloor \alpha_{k_a}$ *around the designated stack and push* Z.

For convenience, we now introduce several moves that can be seen as special cases or combinations of the above moves. Some of these "composite" moves require the introduction of new intermediate states. However, these moves do not constitute a change in the formal definition of the automaton, since the new moves are defined entirely in terms of the two basic moves, UNWRAP and PUSH-NEW-STACK.

1. A single stack symbol can be placed on top of the top stack: a trivial UNWRAP is performed in which nothing is unwrapped (i.e., $k_b = 0 = k_a$, and $\gamma_2 = \epsilon$). This move will be called a PUSH. Formally, if we want to PUSH the symbol Z onto the top stack, we add (q_2, Z) to $\delta(q_1, \varepsilon, \varepsilon, \varepsilon, \varepsilon)$. Note that we can also PUSH more than one symbol onto the top stack, by introducing new intermediate states. Since the number of rules in δ is finite, only a finite number of new states is used.

[16] The notation matches that of Figure 4.8.

[17] Note the notational convention proposed here. We require that if the last component of the range tuple of δ is the bottom-of-stack sign ($\lfloor$), then the second, third, fourth and fifth component of the domain tuple must all be ϵ.

2. A sequence of stack symbols γ_2 can be removed from the top of the top stack, without any symbol being pushed: a trivial UNWRAP is performed in which the stack symbols to be removed plus one more symbol are removed from the designated stack , but no stacks are unwrapped (i.e., $k_b = 0 = k_a$). The additional symbol removed is then pushed back (as part of the UNWRAP move, not as a new move). This move will be called a POP. Formally, if we want to POP the sequence γ_2 off the top stack, we add (q, Z) to $\delta(q, \varepsilon, \varepsilon, Z\Gamma_2, \varepsilon)$ for each $Z \in \Upsilon$.

3. A single stack symbol can be placed in a new stack on top of the stack of stacks: first, a new empty stack is PUSH-NEW-STACKed and the automaton moves into a special state; then, a PUSH is performed. This move will be called a NEW-PUSH. If we also consume an input symbol, it will be called a SHIFT. Formally, if we want to SHIFT the symbol a onto a new top stack, we add a new state q_a^{SHIFT} to Q and the following to δ: $(q_a^{\text{SHIFT}}, [\ ) \in \delta(q, \varepsilon, \varepsilon, \varepsilon, \varepsilon)$ and $(q, a)) \in \delta(q_a^{\text{SHIFT}}, a, \varepsilon, \varepsilon, \varepsilon)$. (The NEW-PUSH case is handled analogously.)

4. The types of move that do not remove any symbols from the top stack (PUSH-NEW-STACK, PUSH, and NEW-PUSH) can be made contingent on the top symbol of the stack of stacks (say, Z), without this symbol being actually removed. To achieve this, we perform an UNWRAP in which the Z is removed and immediately pushed back on the top stack, while the machine transitions to a new state q_Z^{IF} that records that the condition is met. The subsequent PUSH-NEW-STACK, PUSH, or NEW-PUSH move then transitions from state q_Z^{IF} back to the original state q. Conditional execution will be indicated by IF Z. Formally, if we have, for example, IF Z PUSH-NEW-STACK, we add (q_Z^{IF}, Z) to $\delta(q, \varepsilon, \varepsilon, Z, \varepsilon)$ and $(q, [\ )$ to $\delta(q_Z^{\text{IF}}, \varepsilon, \varepsilon, \varepsilon, \varepsilon)$.

4.5.2 An Easy Construction of a BEPDA from a TAG

In this section, we present a method for constructing automata from TAGs, which will be called "the easy method". The construction will serve two purposes:

1. The construction will clearly show the relation between TAG grammars and automata.

2. By showing that the constructed automaton accepts exactly the language generated by the TAG, we can prove that $\mathcal{L}(\text{TAG}) \subseteq \mathcal{L}(\text{BEPDA})$.

Let $G = (V_{\text{N}}, V_{\text{T}}, S, I, A)$ be a TAG. We assume without loss of generality that all nodes labeled ε have no sibling nodes. We let $M = (Q, V_{\text{T}}, \Gamma_n \cup V_{\text{T}}, \delta, q, S)$ be a BEPDA. Let $Q = \{q\} \cup \{q_Z^\zeta \mid Z \in \Gamma_n \cup V_{\text{T}}, \zeta \in \{\text{IF, PUSH, SHIFT}\}\}$. Let Γ_n be a set of unique identifiers for the nodes in the trees, which are split into a top node and a bottom node. Let $\mathsf{label}(\tau, \alpha)$ be the label of the node with address α in tree τ (undefined if τ has no such address). We define:

$$\Gamma_n = \{A^{(\tau,\alpha,\zeta)} \mid A \in V_{\text{N}}, \tau \in I \cup A, \alpha \text{ an address in } \tau \text{ with } \mathsf{label}(\tau, \alpha) = A, \zeta \in \{\text{b}, \text{t}\}\}$$

Observe that the final stack symbol is the start symbol of the TAG. Now we define the transition relation δ. We shall give mnemonic names (which refer to TAG operations) to the different types

of transitions for subsequent easy reference; these names will be given in **boldface**, while the names of formal operations of the BEPDA (in terms of which δ is of course defined) will, as previously, be given in CAPS.

Let τ be a tree in $I \cup A$. We use "Gorn addresses" to denote positions in trees: the root node is denoted by ε, and if a node has address i, then its n daughters have addresses $i \cdot 1, \ldots, i \cdot n$ (where '$\cdot$' denotes concatenation).

1. **Terminal Shift:** For every frontier node of τ that is labeled with a terminal symbol, perform a SHIFT of that symbol. Specifically, if $a \in \Sigma$ is a terminal on the frontier of τ, add $(q_a, [\,)$ to $\delta(q, \varepsilon, \varepsilon, \varepsilon, \varepsilon)$ and (q, a) to $\delta(q_a, a, \varepsilon, \varepsilon, \varepsilon)$.

2. **Reduce:** Let η be a node labeled A in τ with address α, and let its daughters be $\eta_1, \ldots, \eta_n$, such that η_i is labeled $X_i \in V_{\mathrm{T}} \cup V_{\mathrm{N}}$, $1 \leq i \leq n$. Suppose the kth daughter dominates the footnode; set $k = 1$ if no daughter dominates the footnode (or if τ is an initial tree). For each such node η, perform the following UNWRAP move. The $n-k+1$st stack from the top is the designated stack. Remove the $k-1$ stacks $[X_1^{(\tau,\alpha\cdot 1,t)} \cdots [X_{k-1}^{(\tau,\alpha\cdot(k-1),t)}$ from below the designated stack, the $n-k$ stacks $[X_{k+1}^{(\tau,\alpha\cdot(k+1),t)} \cdots [X_n^{(\tau,\alpha\cdot n,t)}$ from above the designated stack, and symbol $X_k^{(\tau,\alpha\cdot k,t)}$ from the top of the designated stack. Then push $A^{(\tau,\alpha,b)}$ on top of the designated stack. Formally, add $(q, A^{(\tau,\alpha,b)})$ to $\delta(q, \varepsilon, [X_1^{(\tau,\alpha\cdot 1,t)} \cdots [X_{(k-1)}^{(\tau,\alpha\cdot(k-1),t)}, X_k^{(\tau,\alpha\cdot k,t)}, [X_{k+1}^{(\tau,\alpha\cdot(k+1),t)} \cdots [X_n^{(\tau,\alpha\cdot n,t)})$. We will say that we **reduce** $[X_1^{(\tau,\alpha\cdot 1,t)} \cdots [X_{k-1}^{(\tau,\alpha\cdot(k-1),t)}$, $X_k^{(\tau,\alpha\cdot k,t)}$, $[X_{k+1}^{(\tau,\alpha\cdot(k+1),t)} \cdots [X_n^{(\tau,\alpha\cdot n,t)}$ to $A^{(\tau,\alpha,b)}$.

3. **ε-Reduce:** Let η be a node labeled A in τ with address α and a single daughter labeled ε. For each such node η, NEW-PUSH the stack pair $[A^{(\tau,\alpha,b)}$ onto the stack of stacks.

4. **Adjoin:** For every node η labeled A at address α in tree τ at which an adjunction of tree τ' (whose footnode η' has address α') can take place, add a PUSH move of the (bottom version of the) foot node of the adjoined tree. Formally, add IF $A^{(\tau,\alpha,b)}$ PUSH $A^{(\tau',\alpha',b)}$ to δ. If the node is marked with the Null-Adjoining (NA) constraint, or with a Selective-Adjoining constraint (SA) which does not include τ', then the rule is not added to δ.

5. **Unadjoin:** For every root node η labeled A of an auxiliary tree τ, POP the stack symbol $A^{(\tau,\varepsilon,t)}$ that represents the root node. This in effect removes any trace of the adjoined tree from the automaton.

6. **Noadjoin:** For every node labeled A at address α that is not marked with the Obligatory-Adjoining (OA) constraint, add an UNWRAP move that changes the stack symbol from the bottom node to the top node. Formally, $(q, A^{(\tau,\alpha,t)}) \in \delta(q, \epsilon, \epsilon, A^{(\tau,\alpha,b)}, \epsilon)$.

Schabes and Vijay-Shanker (1990) present a construction of a deterministic LR(0) parser, which recognizes a subset of $\mathcal{L}$(TAG). Since it is an LR parser, the stack symbols represent sets of possible states in a left-to-right tree traversal of the trees in the grammar. The construction proposed here generates a potentially nondeterministic automaton, but does so for any TAG. It

is the equivalent for TAG of a general shift/reduce parser. It preserves a direct relation between stack symbols and the grammar itself, and requires no precompilation. Apart from allowing us to show equivalence between BEPDA and TAG, this construction also has the advantage that the resulting automaton and its stack symbols are immediately interpretable with respect to the source grammar. If this grammar is linguistically motivated, then the automaton can be described in linguistically motivated terms; thisis exploited in Chapter 6.

We now need to show that the language accepted (by empty stack) by the automaton is exactly the language generated by the grammar.

Lemma 24 *Let G be a TAG and M the BEPDA constructed from G by the easy method. Then $N(M) = L(G)$.*

Outline of the proof. The proof is based on induction on the number of adjunctions in the derivation, or, for the converse direction, on the depth of the stacks in the stack of stacks. We omit the details. ■

Lemma 25 *Let M be a BEPDA. Then there is a TAG G such that $N(M) = L(G)$.*

Outline of the proof. This can be shown by constructing the EPDA M', which is simply the dual of M. One can show by induction on the length of the run that every run of M is a run of M' in reverse. The result then follows from the known equivalence of EPDA and TAG (Vijay-Shanker, 1987). We omit the details. ■

The preceding two lemmas give us the following theorem.

Theorem 26 $\mathcal{L}(\text{BEPDA}) = \mathcal{L}(\text{TAG})$

4.6 The V-TAG Formalism

In this section, we present the tree-rewriting equivalent of UVG-DL, called V-TAG. We start out with definitions in Section 4.6.1, and propose a normal form in Section 4.6.2. A new variant of LIG that is used for subsequent proofs is introduced in Section 4.6.3. In Section 4.6.4, we define a formal automaton and prove equivalence with V-TAG. Section 4.6.5 discusses some formal properties of V-TAG, and Section 4.6.6 makes some remarks on parsing V-TAG.

4.6.1 Definitions of VMC-TAG and V-TAG

In this section, we will give a definition of the tree rewriting version of UVG-DL.

Definition 20 *A* **Vector MC-TAG** *(*VMC-TAG*) is a 5-tuple (V_N, V_T, S, V_I, V_A),*[18] *such that V_N and V_T are disjunct sets of nonterminal and terminal symbols, respectively, $S \in V_N$ is the start*

[18] We assume that trees are uniquely identified, but do not include the labels in the definition. Furthermore, nodes in trees can be uniquely identified by stating the tree and their address within the tree.

symbol, and V_I and V_A are both sets of sets of trees, called vectors. The sets in V_I each contain exactly one initial tree, in addition to an unbounded number of auxiliary trees, while the sets in V_A contain only auxiliary trees.

A derivation in a VMC-TAG is defined as in TAG, with the additional condition that for each tree vector, the same number of trees from that vector is used in the derivation.

The language generated by a V-TAG *G, L(G), is defined as for* TAG.

Thus, a derivation in a VMC-TAG proceeds as follows: a single vector in V_I is chosen. The unique initial tree contained in that element of V_I will serve as the initial tree for the derivation. Vectors from V_A may then be chosen and their members adjoined. There is no constraint on adjunction (of auxiliary trees from the chosen initial set or from members of V_A), other than that all members of an instance of a tree set must be adjoined *at some point during the derivation* if at least one member is adjoined. Note that this definition is like that of non-local MC-TAG (nlMC-TAG; Weir's "Type 3" (Weir, 1988, p.32), which is equivalent to his "Type 4") in that there is no locality restriction on the loci of adjunction for trees from one set. However, crucially, the requirement for simultaneous adjunction of trees in one set, which holds for all of the MC-TAGs defined by Weir (1988), has been lifted.

We now add dominance links.

Definition 21 *A* **Vector MC-TAG with Dominance Links** *(*VMC-TAG-DL *or* V-TAG*, for short) is a 5-tuple (V_N, V_T, S, V_I, V_A), where V_N, V_T, S, V_I, and V_A are as for VMC-TAG, except that the vectors in V_I and V_A are equipped with dominance links. For a given vector $v \in V_I \cup V_A$, the dominance links form a binary relation dom_v over the set of nodes in the trees of v such that if $\mathsf{dom}_v(\eta_1, \eta_2)$, then η_1 is the footnode of an auxiliary tree in v, and η_2 is any node in any tree of v.*

Derivations and the derived language are defined as for VMC-TAG, except that the dominance links provide a constraint on possible derivations: after a derivation is completed, the union of the dominance links of the grammar must be a subset of the dominance relation (transitive-reflexive closure of the immediate dominance relation) of the derived tree.

Observe that, if we have $\mathsf{dom}_v(\eta_1, \eta_2)$, we can eliminate cases in which η_1 and η_2 are in the same tree, since if η_2 is not the footnode, the link represents an unsatisfiable constraint, while if it is, it is trivially satisfied. We will therefore assume without comment that dominance links connect two different trees. We will say that two trees τ_1, τ_2 are *linked by* dom_v if there is a dominance link between a node in τ_1 and a node in τ_2. Two trees τ_1, τ_n are *connected by* dom_v if there are trees $\tau_2, \ldots, \tau_{n-1}$ such that τ_i is linked to τ_{i+1}, $1 \leq n - 1$.

We will assume that the trees are equipped with adjunction constraints in the way described in the literature. As shown by Vijay-Shanker (1987) and Vijay-Shanker and Joshi (1988), the formal properties of the formalisms are not altered if we replace the adjunction constraints by bounded feature structures.

We will now introduce some more terminology, which is similar to that used for derivations of UVG-DLs. If two nodes η_1 and η_2 from trees in vector v are linked by a dominance link such

that η_1 dominates η_2, then we will denote this link by l_{v,η_1,η_2}. We will say that η_1 has a *passive dominance requirement* of l_{v,η_1,η_2}, and that η_2 has an *active dominance requirement* of l_{v,η_1,η_2}. If the tree of which η_1 (η_2) is a node has been adjoined during a derivation, but the tree of which η_2 (η_1) is a node has not, the dominance requirement (passive or active) will be called *unfulfilled*. The multiset of unfulfilled active dominance requirements of a node η will be denoted by $\top(\eta)$, and the multiset of all passive dominance requirements will be denoted by $\bot(\eta)$. We extend this notation to derivations. Let ϱ be a (partial) derivation. We associate with ϱ a multi-set which represents all the unfulfilled active dominance requirements of nodes in trees that have been used ϱ, written $\top(\varrho)$, and a multiset which represents all the unfulfilled active dominance requirements of nodes in trees that have been used in ϱ, written $\top(\varrho)$.

These dominance links are essentially the same that are defined for MC-TAG-DL by Becker et al. (1991), and that have been used previously by Kroch (1989). They should not be confused with the "links" introduced by Joshi (1985), which relate nodes within a single tree and are used to denote linguistic dependencies, without affecting the formal definition of the system.

EXAMPLE 9

As an example, we give a linguistically inspired V-TAG-Δ. This grammar can derive sentence (35), page 24, which motivated the introduction of a multi-component system that does not require simultaneous adjunction (Section 3.4.3, page 56). The linguistic details of the trees do not matter here, and this example is only given to illustrate the formalism and to show how the formalism can derive certain structures, not to suggest a particular linguistic analysis. The grammar is given in Figure 4.10, it consists of two tree sets. Figure 4.11 shows the result of adjoining the tree rooted in C$'$ at the C$'$ node of the tree rooted in CP. The two trees recursive on VP remain unadjoined. The final derived tree is shown in Figure 4.12. □

Definition 22 *A* **Vector MC-TAG with Dominance Links and integrity** *(*V-TAG-Δ*) is a 5-tuple* $(V_{\mathrm{N}}, V_{\mathrm{T}}, S, V_I, V_A)$*, where* V_{N}*,* V_{T} *,* S*,* V_I*, and* V_A *are as for* V-TAG*, except that nodes in trees may be marked with the integrity constraint, written* Δ*.*

Derivations and the derived language are defined as for V-TAG*, except that the integrity constraints provide a constraint on possible derivations: after a derivation is completed, the subtree rooted in a node with integrity may not have any unfulfilled active dominance requirements other than those that originate with the root node itself.*

We will now define several types of restricted V-TAGs, which are similar to the restricted {}-LIG varieties introduce in Section 4.3.1, page 66. We start out with "linear restriction", which will be given a narrower definition for the tree rewriting case than for the string rewriting case.

Definition 23 *A* **linearly-restricted** *derivation of a terminal string* w *in a* V-TAG *or a* V-TAG-Δ *is a derivation* ϱ *such that the total number of vectors used during the derivation is linearly bounded by* $|w|$*.*

We let $L_R(G) = \{w \mid \text{there is a derivation } \varrho \text{ of } w \text{ in } G \text{ such that } \varrho \text{ is linearly-restricted}\}$, and we let $\mathcal{L}_R(\mathcal{F}) = \{L_R(G) \mid G \in \mathcal{F}\}$. If G is a V-TAG (V-TAG-Δ) such that every $L_R(G) = L(G)$, we say that G is linearly restricted.

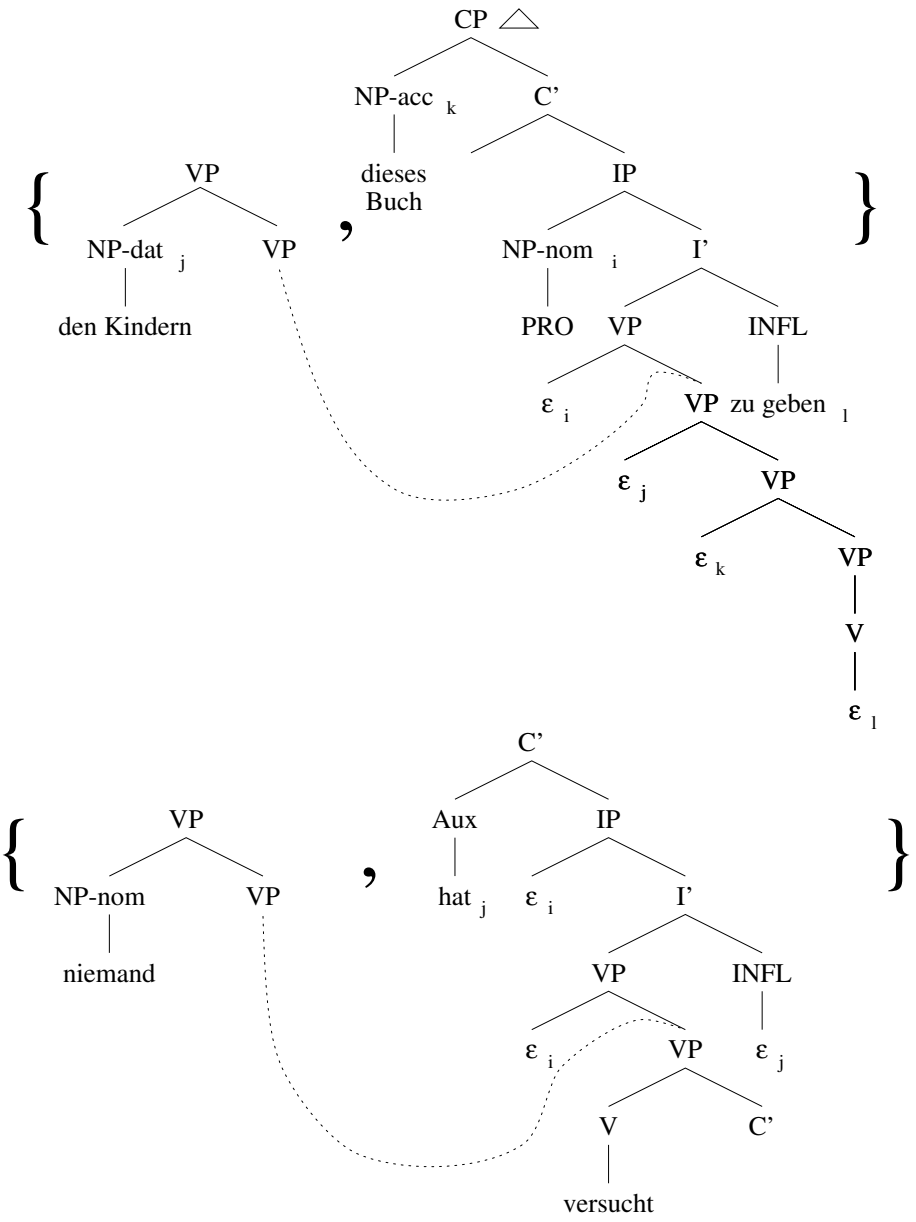


Figure 4.10: Tree sets in V-TAG G

Definition 24 *Let q be a function over the positive integers. A* **q-form-restricted** *derivation in a* V-TAG *is a derivation ϱ of w with $w \in V_{\mathrm{T}}^*$ such that the total number of unfulfilled dominance requirements (active or passive) of any subtree of the derived tree is bounded by $q(|w|)$.*

We let $L^q_{\mathrm{fR}}(G) = \{w | \text{there is a } q\text{-form-restricted derivation of } w\}$, and we let $\mathcal{L}^q_{\mathrm{fR}}(\mathcal{F}) = \{L^q_{\mathrm{fR}}(G) | G \in \mathcal{F}\}$. If G is a V-TAG such that every $L^q_{\mathrm{fR}}(G) = L(G)$, we say that G is q-form-restricted. Clearly, a linearly restricted derivation is also a q-form-restricted derivation for some $q \in O(n)$, but the converse is not true. Note that a c-form-restricted V-TAG for $c \in O(1)$ is equivalent to a TAG.

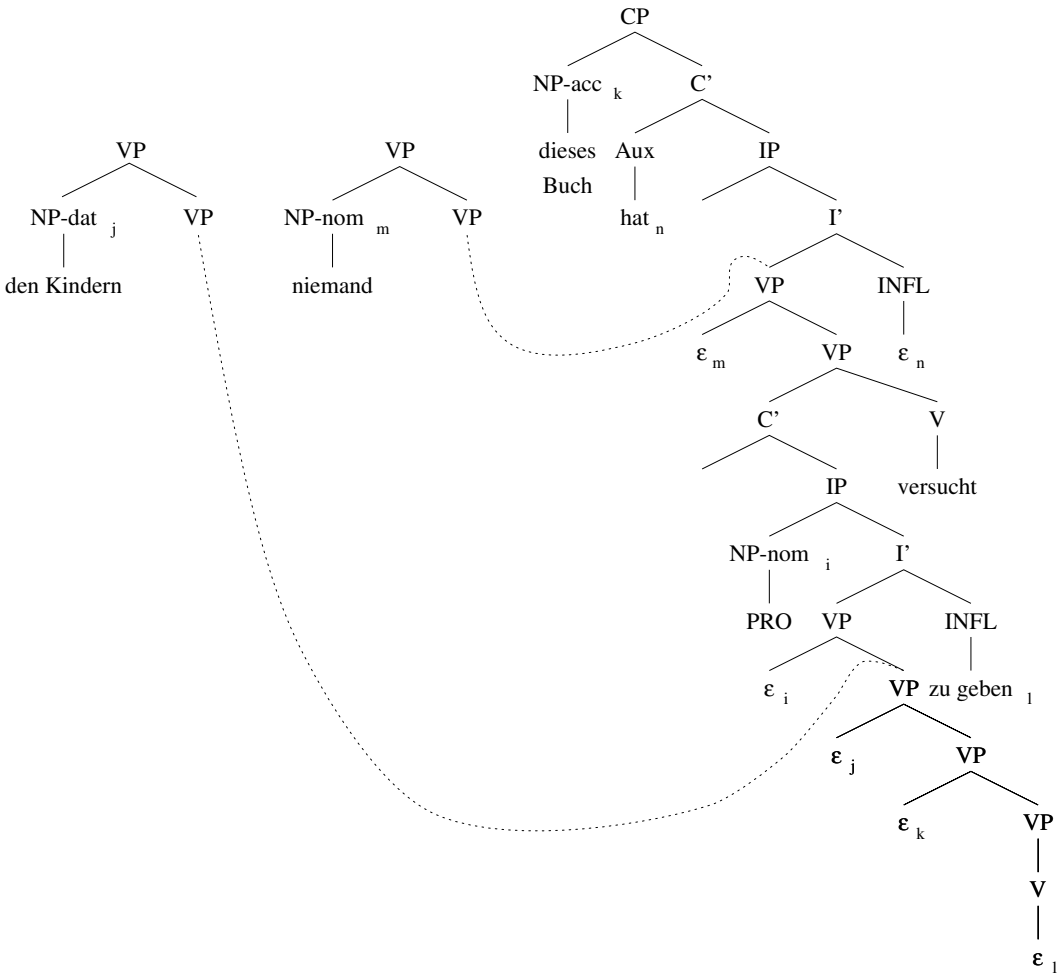


Figure 4.11: After adjoining matrix clause into subordinate clause

4.6.2 Normal Form

The following normal form will be used subsequently to prove equivalence to the formal automaton and to a LIG-variant.

Definition 25 *A* **strict** V-TAG (V-TAG-Δ) *is a* V-TAG *(*V-TAG-Δ*) in which all vectors are such that every tree in the vector is connected to every other tree in the vector.*

Theorem 27 *For every* V-TAG G *there is a strict* V-TAG G' *such that* $L(G) = L(G')$.

The proof is very similar to that of Theorem 17, page 89, and we omit it.

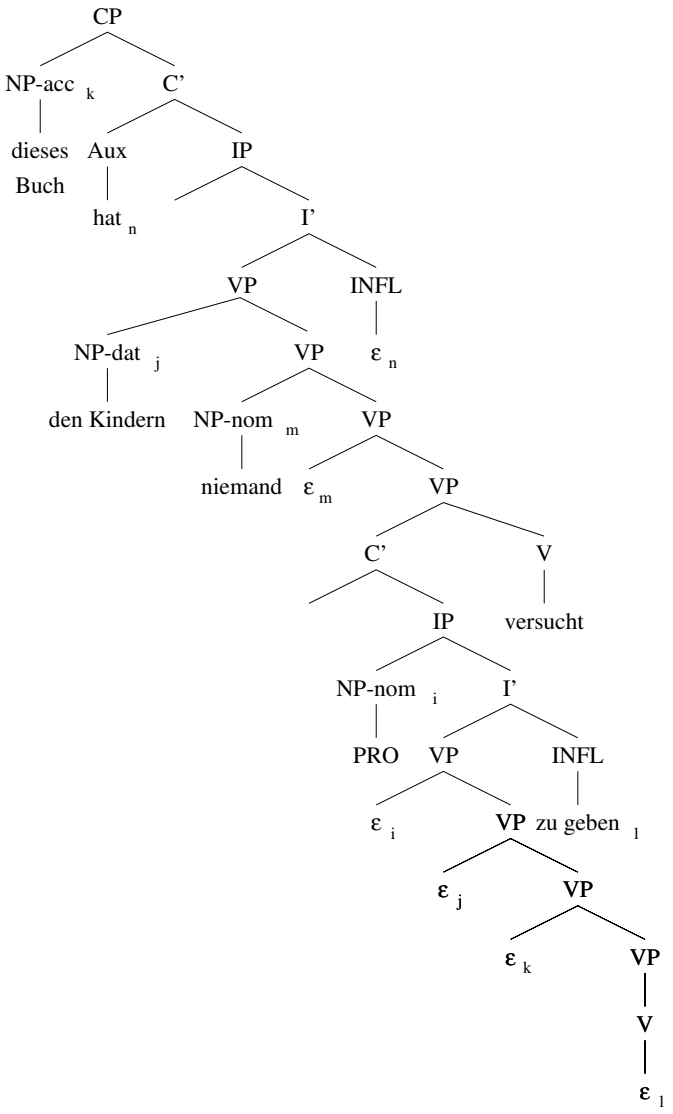


Figure 4.12: Final derived tree

4.6.3 LIG Variants

UVG-DL, the string-rewriting version of V-TAG, is weakly equivalent to {}-LIG. Is there a LIG-variant which is in weakly equivalent to V-TAG? We would expect this to be the case, since the original stack-valued LIG ([]-LIG) is in fact equivalent to TAG. In []-LIG, the stack is used to record the sequence of adjunctions in the TAG and thus the dependencies between the part of the tree above the adjunction point and the subtree below it. In {}-LIG, the set is used to record the "vertical" dependencies between applications of rules related by dominance links in the UVG-DL. In V-TAG, both types of dependencies will need to be recorded. The dependencies created by adjunction cannot be modeled by a set, since the sequence is crucial, while the dependencies created by adjoining two trees connected by a domainance link cannot be modeled by a stack,

since order of adjunction is free. Therefore, we will need both a stack and a multiset.[19] This system will be called a stack/multiset-valued LIG ([]{}-LIG). Formally, it is defined as follows:

Definition 26 *A* **stack/multiset-valued Linear Index Grammar** *([]{}-LIG) is a 6-tuple* $(V_{\mathrm{N}}, V_{\mathrm{T}}, V_{[\mathrm{I}]}, V_{\{\mathrm{I}\}}, P, S)$*, where* V_{N} *and* V_{T} *are the sets of terminals and non-terminals, respectively,* $V_{[\mathrm{I}]}$ *is the set of stack indices,* $V_{\{\mathrm{I}\}}$ *is the set of multiset indices,* S *is the start symbol and* P *is a set of productions of one of the following two forms:*

1. $p : A[\cdot\cdot r]s \longrightarrow v_0 B_1[r_1]s_1 v_1 \ldots v_{k-1} B_k[\cdot\cdot r_k]s_k v_k \ldots v_{n-1} B_n[r_n]s_n v_n$
2. $p : A[r]s \longrightarrow v_0 B_1[r_1]s_1 v_1 \ldots v_{k-1} B_k[r_k]s_k v_k \ldots v_{n-1} B_n[r_n]s_n v_n$

Here, n *is an integer,* $A, B_1, \ldots, B_n \in V_{\mathrm{N}}$*,* $r, r_1, \ldots, r_n \in V_{[\mathrm{I}]}^*$*,* $s, s_1, \ldots, s_n$ *are multisets of members of* $V_{\{\mathrm{I}\}}$*, and* $v_0, \ldots, v_n \in V_{\mathrm{T}}^*$*.*

The derivation relation $\Longrightarrow$ *for a []{}-LIG is defined as follows. Let* $\beta, \gamma \in (V_{\mathrm{N}}[V_{[\mathrm{I}]}^*]\mathcal{M}(V_{\mathrm{I}}) \cup V_{\mathrm{T}})^*$*,* $r, r', r_1, \ldots, r_n \in V_{[\mathrm{I}]}^*$*,* $t, t_1, \ldots, t_n$ *multisets of members of* $V_{\{\mathrm{I}\}}$*. If* $p \in P$ *is a rule of type (i), we have:*

$$\beta A[r'r]t\gamma \Longrightarrow \beta v_0 B_1[r_1]t_1 v_1 \ldots v_{k-1} B_k[r'r_k]t_k v_k \ldots v_{n-1} B_n t_n v_n \gamma$$

such that $t = \cup_{i=1}^{n}(t_i \setminus s_i) \cup s$*. If* $p \in P$ *is a rule of type (ii), we have:*

$$\beta A[r]t\gamma \Longrightarrow \beta v_0 B_1[r_1]t_1 v_1 \ldots v_{k-1} B_k[r_k]t_k v_k \ldots v_{n-1} B_n t_n v_n \gamma$$

such that $t = \cup_{i=1}^{n}(t_i \setminus s_i) \cup s$*.* $L(G)$ *for a []{}-LIG is defined as usual.*

Note that the "usual definition" of the language generated by a []{}-LIG implies that all index symbols – both stack and multiset – must be used up during a derivation; words of the generated language consist only of terminal symbols.

We can extend the definition of []{}-LIG to *stack/multiset-valued Linear Index Grammar with integrity* ([]{}-LIG-Δ) in a manner parallel to the extension of {}-LIG to {}-LIG-Δ (Definition 6, page 72). We omit the details.

We now give a normal form for []{}-LIG which is motivated principally by the requirements of the following proof.

Definition 27 *A []{}-LIG* $G = (V_{\mathrm{N}}, V_{\mathrm{T}}, V_{[\mathrm{I}]}, V_{\{\mathrm{I}\}}, P, S)$ *is in* **restricted multiset index binary normal form** *or RIBNF if all productions in* P *are of one of the following forms (where* $A, B \in V_{\mathrm{N}}$*,* $r, r_1, r_2 \in V_{\mathrm{I}}$*,* $f \in V_{\{\mathrm{I}\}}$ *and* $\alpha \in (V_{\mathrm{T}} \cup V_{\mathrm{N}})^*$*):*

[19] Observe that using two stacks would make the system Turing-equivalent, since they could simulate the tape of a Turing machine. However, such a simulation is not possible using a multiset or a multiset and a stack.

1. $A[\cdots] \longrightarrow B_1[\cdots r]B_2$
2. $A[\cdots] \longrightarrow B_1 B_2[\cdots r]$
3. $A[\cdots r] \longrightarrow B_1[\cdots]B_2$
4. $A[\cdots r] \longrightarrow B_1 B_2[\cdots]$
5. $A[] \longrightarrow w$
6. $A[\cdots] \longrightarrow B[\cdots]\{f\}$
7. $A[\cdots]\{f\} \longrightarrow B[\cdots]$
8. $A[\cdots] \longrightarrow B[\cdots]$

Theorem 28 *For any* []{}-LIG *(*[]{}-LIG-Δ*), there is an equivalent* {}-LIG *(*{}-LIG-Δ*) in RIBNF.*

Outline of the proof. We can sequence the proof into several parts. First, we use the techniques employed in the proof of Theorem 3, page 75, to isolate multiset index symbols in productions of type (vi) and (vii). Then, we use techniques employed by Vijay-Shanker and Weir (1992, Section 3.2) to eliminate []{}-LIG productions of type (ii), and to have the right-hand side of production be all terminals or all nonterminals with stack symbols. Finally, we use standard techniques developed for CFGs to reduce the number of nonterminals to two or fewer, as we have done previously in the proof of Theorem 4, page 76. We do not give a full proof here, since the details are straightforward but tedious. ■

We will show that []{}-LIG is weakly equivalent to V-TAG. We do so in two steps; here, we show inclusion of $\mathcal{L}$([]{}-LIG) in $\mathcal{L}$(V-TAG); the converse is postponed to Section 4.6.4.

Lemma 29 $\mathcal{L}$([]{}-LIG) $\subseteq$ $\mathcal{L}$(V-TAG)

Proof. We will follow the proof of $\mathcal{L}$([]-LIG) $\subseteq$ $\mathcal{L}$(TAG) given by Vijay-Shanker and Weir (1992, Sections 3.2 and 3.3). They give the proof in two steps, first constructing a head grammar (HG), and thence a TAG. We telescope the two steps into one. Observe that our normal form for []{}-LIG is stricter than theirs for []-LIG, anticipating their (equally strict) normal form for HGs.

Let $G = (V_{\mathrm{N}}, V_{\mathrm{T}}, V_{[\mathrm{I}]}, V_{\{\mathrm{I}\}}, P, S)$ be a []{}-LIG in RIBNF. We construct a V-TAG $G_V = (V_{\mathrm{N}}^{(V)}, V_{\mathrm{T}}, S, V_I, V_A)$, where $V_{\mathrm{N}}^{(V)} = V_{\mathrm{N}} \cup \{(A_1, A_2, r) \mid A_1, A_2 \in V_{\mathrm{T}} \text{ and } r \in V_{[\mathrm{I}]} \cup \{\varepsilon\}\}$, and V_I and V_A are defined as follows.

1. If $p \in P$ is a production of type (i), (ii), (iii), or (iv), then for all $C \in V_{\mathrm{N}}^{(V)}$ add a vector to V_A whose single element is a tree of type (i), (ii), (iii), or (iv), respectively, as shown in Figure 4.13.
2. If $p \in P$ is a production of type (v), then add a vector whose single element is a tree of type (v) (Figure 4.14).

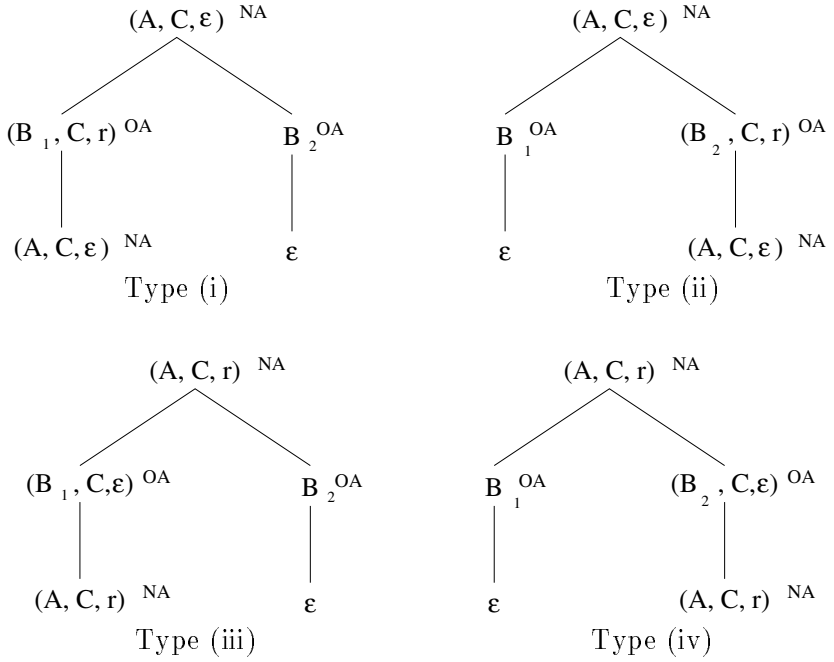


Figure 4.13: Construction of grammar G_V

3. For all $A, B, C \in V_{\mathrm{N}}^{(V)}$, and $r \in V_{[\mathrm{I}]} \cup \{\varepsilon\}$, add four vectors to V_A whose single elements are a tree of type (vi), (vii), and (ix), respectively (Figure 4.14).

4. Add a vector to V_I containing as its single element a tree of type (viii) (see Figure 4.14).

5. Let $p \in P, p : A_1 \longrightarrow B_1\{f\}$, with $A_1, B_1 \in V_{\mathrm{N}}$ and $t \in V_{\{\mathrm{I}\}}$, be a production of type (vi). For each production $p' \in P, p' : A_2\{f\} \longrightarrow B_2$ of type (vii), and for every $C, D \in V_{\mathrm{N}}^{(V)}$, add a vector to V_A of type (x) (Figure 4.14). We will call these vectors *dominance vectors*.

6. Let p be a vector of type (viii). Then for every $C \in V_{\mathrm{N}}^{(V)}$, add a production of type (xi) (Figure 4.14) to A_V.

We will show by induction that $L(G) = L(G_V)$.

We first show $L(G) \subseteq L(G_V)$. Specifically, we show that for all integers k, for all $[]\{\}$-LIGs G, if there is a derivation in G of string w that uses a total of k instances of productions of type (vi) (and hence the same number of productions of type (vii)), then there is a derivation in G_V of w which uses a total of k dominance-vectors.

The base case ($k = 0$) reduces to showing that the language generated by a LIG is included in the language generated by the TAG constructed from it as above. This was proved by Vijay-Shanker and Weir (1992, Sections 3.2 and 3.3).[20]

[20] The inclusion of $\mathcal{L}([]$-LIG) in $\mathcal{L}$(TAG) was first shown by Vijay-Shanker (1987), where a different construction (involving the EPDA) was used.

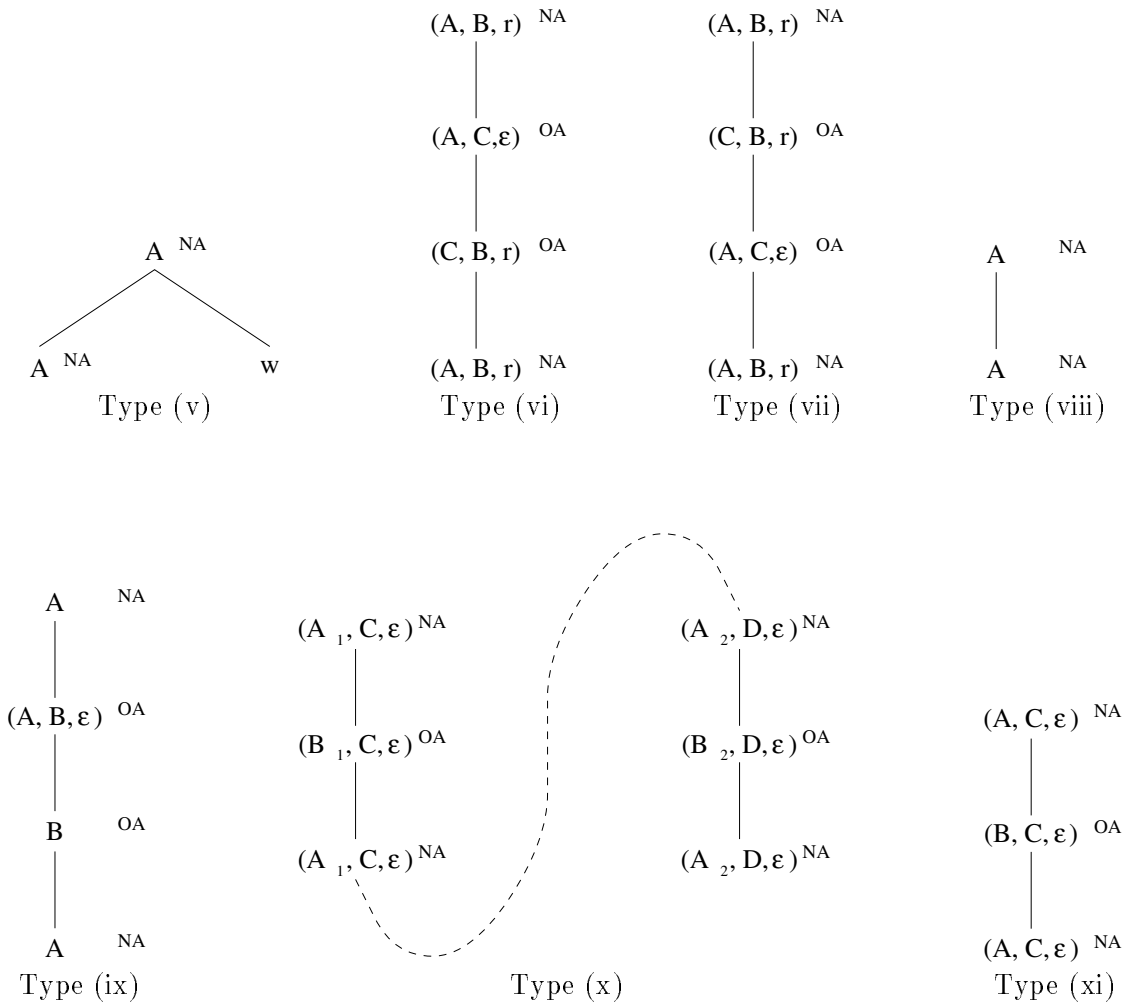


Figure 4.14: Construction of grammar G_V (continued)

For the induction step, assume that G is a $[]\{\}$-LIG in which there is a derivation ϱ of string w that uses a total of k instances of rules of type (vi). We construct a new grammar $G' = (V_{\mathrm{N}}, V_{\mathrm{T}}, V_{[\mathrm{I}]}, V_{\{\mathrm{I}\}}, P', S)$ as follows. We choose a rule p_1 of type (vi) and a rule p_2 of type (vii) such that they manipulate the same multiset index symbol, and such that they are both used in ϱ with an instance of p_2 removing the multiset index symbol introduced by an instance of p_1. Let P' be the set of productions obtained from P by adding to it two productions p_1', p_2' of type (viii) which are like p_1 and p_2, respectively, except that they neither add nor remove a multiset index symbol. There is a derivation ϱ' in G' of w which is like ϱ except that an instance of application of p_1 has been replaced by an application of p_1', and similarly for p_2, p_2'. Clearly, ϱ' uses $k-1$ instances of rules of type (vi) (and the same number of rules of type (vii)). By induction hypothesis, there is a derivation ϱ_V' of w in the V-TAG derived from G', G_V', which uses $k-1$ dominance-vectors. From the construction of G_V', it can easily be seen that in the derived tree of

ϱ'_V, the tree derived from with p'_1 (there is only one) dominates that derived from p'_2. Therefore, we can obtain a new derivation ϱ''_V in which we use the dominance-vector derived from p_1 and p_2 instead of the two singleton vectors derived from p'_1 and p'_2, respectively. Clearly, ϱ''_V is also a derivation in G_V and it uses k instances of dominance-vectors. This concludes the induction step.

We now show $L(G_V) \subseteq L(G)$. Specifically, we show that for all integers k, for all V-TAGs G', if there is a derivation in G' of string w that uses a total of k dominance-vectors, then there is a derivation in G of w which uses a total of k instance of productions of type (vi) (and hence the same number of productions of type (vii).

The base case ($k = 0$) reduces to showing that the language generated by a TAG constructed as above is included in the language generated by the LIG from which it was constructed by this method, which, again, was proved by Vijay-Shanker and Weir (1992, Sections 3.2, 3.3).

For the induction step, assume that $G_V = (V_{\mathrm{N}}^V, V_{\mathrm{T}}, S, V_I, V_A)$ is a V-TAG as constructed above in which there is a derivation ϱ_V of string w that uses a total of k dominance-vectors. Let $G = (V_{\mathrm{N}}, V_{\mathrm{T}}, V_{[\mathrm{I}]}, V_{\{\mathrm{I}\}}, P, S)$ be the []{}-LIG from which G_V was constructed. We let $G'_V = (V_{\mathrm{N}}^V, V_{\mathrm{T}}, S, V_I, V'_A)$ be a new V-TAG, and construct V'_A as follows. For some f such that ϱ_V uses a dominance-vector $v \in V_A$ that is constructed from rules $p_1, p_2 \in P$ that insert and remove, respectively, multiset index symbol f, introduce two new vectors v_1, v_2 of type (xi) into V_A that each contain one of the trees of v. There is a derivation ϱ'_V in G'_V which is identical to ϱ_V, except that it uses v_1, v_2 where G_V uses v. Clearly, ϱ'_V uses $k-1$ instances of dominance-vectors. By inspection of the construction of V-TAGs from []{}-LIGs, we can see that there is a []{}-LIG G' such that G'_V is derived from it by this construction (which is not a matter of course), and furthermore, that G' is identical to G, except that its set of productions contain additionally two productions p'_1, p'_2 that are identical to p_1, p_2, except that they neither add nor remove multiset stack symbols. By induction hypothesis, we have a derivation ϱ' in G' of w which uses $k-1$ instances of productions of type (vi). By the method of construction, if the tree derived from p'_1 dominates the tree derived from p'_2 in the derived tree of ϱ'_V, then the same is true in the derivation tree of ϱ'. Therefore, there is a derivation ϱ in G' which uses p_1, p_2 instead of p'_1, p'_2, and hence uses k instances of productions of type (vi). But ϱ uses no productions that are not also productions of G. Hence ϱ is also a valid derivation of w in G, and we are done.

We conclude that $L(G) = L(G_V)$. ■

4.6.4 The {}-BEPDA

In this section, we define the formal automaton {}-BEPDA, and prove that it is equivalent to V-TAG.

{}-BEPDA: Definition

Recall that in a V-TAG, we have sets of trees that are linked by dominance constraints. All trees from a tree set need to be adjoined during the derivation process, though not necessarily simultaneously, in such a way that the dominance constraints hold. These dominance links can connect the foot node of one tree to an internal node in another tree. Thus, in a bottom-up

derivation, when we reach a node which is linked to the foot node of another tree by a dominance link, we must be able to record the fact that at some node dominating the current one the linked tree must be adjoined. In order to achieve this, we will extend the stack language of the BEPDA by adding multisets of indices from a separate index language to each stack in the stack of stacks. These sets of indices are passed upward from unwrapped stacks to the new top stacks, which corresponds to passing indices up through the derived tree; they are never copied (the {}-BEPDA is thus in fact linear). When an index is discharged, an adjunction of the corresponding tree is initiated. Since the indices are grouped in sets, they can be discharged in any order, just as there is no order requirement on the adjunction of trees in V-TAG.

The crucial part is the definition of the transition relation δ. The moves defined for the BEPDA are augmented by instructions on stack operations. Basically, any move can either be dependent on the presence of a single index in the index set of the top stack, which is removed when the move has been executed, or one or more symbols can be added to the index set of the top stack. The moves now operate as follows:

1. In the UNWRAP move, the index set of the new top stack is the union of all the index sets of the unwrapped stacks (including the designated stack that now has become the top stack).
2. In the PUSH-NEW-STACK move, the index set associated with the new stack is empty.

We now provide a formal definition.

Definition 28 *A* **Multiset-Indexed Bottom-Up Embedded Pushdown Automaton** *({}-BEPDA) is a 7-tuple* $(Q, \Sigma, \Gamma, \Upsilon, \delta, q_0, Z_f)$ *where:*

1. Q *is a finite set of states;*
2. $q_0 \in Q$ *is the start state;*
3. Σ *is a finite set of input symbols;*
4. Γ *is a finite set of stack symbols;*
5. Υ *is a finite set of index symbols;*
6. $Z_f \in \Gamma \cup \{\varepsilon\}$ *is the final stack symbol, which is on the stack at the end of a run;*
7. δ *is a mapping from* $Q \times \Sigma \times (\{\!\!\{\Gamma^+ \times \mathcal{M}(\Upsilon)\}\!\!\})^* \times (\Gamma^* \times \mathcal{M}(\Upsilon)) \times (\{\!\!\{\Gamma^+ \times \mathcal{M}(\Upsilon)\}\!\!\})^*$ *to finite subsets of* $Q \times ((\Gamma \times \mathcal{M}(\Upsilon)) \cup \{\{\!\!\{\}\!\!\}\})$.[21] *The domain 5-tuple corresponds to current state, input symbol, stacks with index symbols below the designated stack to be removed, stack symbols and index symbols to be removed from the designated stack, and stacks with index symbols above the designated stack to be removed. The range pair corresponds to the new state and the new stack symbol to be pushed on the designated stack, and the new index symbols to be added to the designated stack's multiset of indices.*

[21] The fastidious reader will have observed that notation is again being abused, since string-notation is being mixed with multiset-notation. Again, it is clear how to use a pure string-based representation, but this would make the presentation much more arduous.

As in the case of the BEPDA, we use '$\blacksquare$' (not in Γ) as the symbol representing the bottom of the stack.

An **instantaneous description** *(ID) for a {}-BEPDA is a "snapshot" of the automaton. It is a member of $Q \times (\blacksquare\Gamma^+ \times \mathcal{M}(\Upsilon))^* \times \Sigma^* \times \Sigma^*$; its four components represent the current state, the current configuration of the stack of stacks, the part of the input word that has been read, and the part of the input word that has not yet been read, respectively.*

The **move relation** *of a {}-*BEPDA *is a (reflexive, transitive) relation $\vdash$ between two IDs. Let $M = (Q, \Sigma, \Gamma, \Upsilon, \delta, q_0, Z)$ be a {}-*BEPDA*, $I_1 = (q, \gamma, w_1, aw_2)$, $a \in \Sigma$, an ID, and I_2 also an ID. We have $I_1 \vdash I_2$ iff one of the two following conditions obtains:*

1. *The following conditions all hold:*
   (a) $\gamma = \gamma'(\blacksquare\beta_1, t_1)(\blacksquare\beta_2, t_2)\cdots(\blacksquare\beta_{k-1}, t_{k-1})(\blacksquare\gamma_1\gamma_2, t_k)(\blacksquare\beta_{k+1}, t_{k+1})\cdots(\blacksquare\beta_n, t_n)$ *for some* $\gamma' \in ((\blacksquare\Gamma^+) \times \mathcal{M}(\Upsilon))^*$, *and for* $\beta_i \in \Gamma^+, 1 \leq i \leq n$, $t_i \in \mathcal{M}(\Upsilon), 1 \leq i \leq k_a$, *and* $\gamma_1\gamma_2 \in \Gamma^+$;
   (b) $I_2 = (q', (\gamma'\blacksquare\gamma_1 Z, t), w_1 a, w_2)$, *for some* $q' \in Q$ *and* $Z \in \Gamma$;
   (c) $(q', (Z, s)) \in \delta(q, a, (\blacksquare\beta_1, s_1)\cdots(\blacksquare\beta_{k-1}, s_{k-1}), (\gamma_2, s_k), (\blacksquare\beta_{k+1}, s_{k+1})\cdots(\blacksquare\beta_n, s_n))$;
   (d) *and* $t = \cup_{i=1}^{n}(t_i \setminus s_i) \cup s$.

   This is the UNWRAP *move.*
2. $I_2 = (q', \gamma(\blacksquare, \emptyset), w_1, aw_2)$ *and* $(q', (\blacksquare, \emptyset)) \in \delta(q, \varepsilon, \varepsilon, (\varepsilon, \emptyset), \varepsilon)$.
   This is the PUSH-NEW-STACK *move.*[22]

*The language accepted (by null stack) by a {}-*BEPDA *$M = (Q, \Sigma, \Gamma, \Upsilon, \delta, q_0, Z_f)$ is defined to be*

$$N(M) = \{w \mid (q_0, \varepsilon, \varepsilon, w) \vdash^* (q, Z_f, w, \varepsilon), q \in Q\}.$$

As in the case of the BEPDA, the term "acceptance by null stack" is not entirely accurate.

We will say that an UNWRAP move of the automaton *collects the indices $t_1, \ldots, t_n$ into t in accordance with δ*. A set of final states can be included in the formal definition of the {}-BEPDA, and acceptance by final state defined in the usual way.

Observe that we can extend the definition somewhat by associating a set of indices not with every stack, but with every stack symbol. Suppose we require that any operations on index sets be performed on the index set of the top stack symbol in a stack only, and at the same allow these index symbols to be passed freely between the index sets of stack symbols of the same stack. It is clear that the "extended definition" (as we will call it) does not affect the formal properties of the system. We will make use of this extended definition in Section 6.3, where we define a {}-BEPDAfor modeling linguistic performance.

[22] We use the same notation as previously: we require that if the last component of the range tuple of δ is the bottom-of-stack sign, then the second, third, fourth and fifth component of the domain tuple must all be ε.

As in the case of the simple BEPDA, we can define composite moves. These moves are the same as for the BEPDA (page 99), and we describe in detail only the effects on the multiset of index symbols.

1. In a PUSH, a single stack symbol can be placed on top of the top stack, and new index symbols added to the multiset of the top stack. Formally, if we want to PUSH the pair (Z, t) onto the top stack, where Z is a stack symbol and t is a multiset of indices, we add $(q_2, (Z, t))$ to $\delta(q_1, \varepsilon, \varepsilon, (\varepsilon, \emptyset), \varepsilon)$.

2. A sequence of stack symbols γ_2 can be removed from the top of the top stack and a multiset of index symbols t can simultaneously be removed from the index set associated with the top stack, without any symbol being pushed. This move will be called a POP. Formally, if we want to POP the sequence γ_2 off the top stack and remove index symbols t (a multiset), we add $(q, (Z, \emptyset))$ to $\delta(q, \varepsilon, \varepsilon, (Z\Gamma_2, t), \varepsilon)$ for each $Z \in \Upsilon$.

3. A single stack symbol can be placed in a new stack on top of the stack of stacks and the new top stack can be given an index set. This move will be called a NEW-PUSH. If we also consume an input symbol, it will be called a SHIFT. Formally, if we want to SHIFT the symbol a onto a new top stack and give it index multiset t, we add a new state $(q_a^{\text{SHIFT}}$ to Q and $(q_a^{\text{SHIFT}}, \boxed{\ }\ )$ to $\delta(q, \varepsilon, \varepsilon, (\varepsilon, \emptyset), \varepsilon)$ and $(q, (a, t))$ to $\delta(q_a^{\text{SHIFT}}, a, \varepsilon, (\varepsilon, \emptyset), \varepsilon)$.

4. The types of move that do not remove any symbols from the top stack (PUSH-NEW-STACK, PUSH, and NEW-PUSH) can be made contingent on the top symbol of the stack of stacks (say, Z), without this symbol being actually removed. To achieve this, we perform an UNWRAP in which the Z is removed and immediately pushed back on the top stack, while the machine transitions to a special state q_Z^{IF} that records that the condition is met. The subsequent PUSH-NEW-STACK, PUSH, or NEW-PUSH move then transitions from state q_Z^{IF} back to the original state q. Conditional execution will be indicated by IF Z. Formally, if we have, for example, IF Z PUSH-NEW-STACK, we add $(q_Z^{\text{IF}}, (Z, \emptyset))$ to $\delta(q, \varepsilon, \varepsilon, (Z, \emptyset), \varepsilon)$ and $(q, (\boxed{\ }, \emptyset))$ to $\delta(q_Z^{\text{IF}}, \varepsilon, \varepsilon, (\varepsilon, \emptyset), \varepsilon)$.

An Easy Construction of a {}-BEPDA from a V-TAG

In this subsection, we present a method for constructing a {}-BEPDA from a V-TAG which, like the previous construction of a BEPDA from a TAG on which it is based (Section 4.5.2, page 100), will be called "the easy method".

Let $G = (V_{\text{N}}, V_{\text{T}}, S, V_I, V_A)$ be a V-TAG. We assume without loss of generality that all nodes labeled ε have no sibling nodes. We let $M = (Q, V_{\text{T}}, \Gamma_n \cup V_{\text{T}}, \Upsilon, \delta, q_0, S)$ be a {}-BEPDA. Let $Q = \{q_0\} \cup \{q_Z^\zeta \mid Z \in \Gamma_n \cup V_{\text{T}}, \zeta \in \{\text{IF, PUSH, SHIFT}\}\}$. Γ_n is a set of unique identifiers for the nodes in the trees, which are split into a top node and a bottom node. More precisely, let $\mathsf{label}(\tau, \alpha)$ be the label of the node with address α in tree τ (undefined if τ has no such address). We then define Γ_n as follows:

$$\Gamma_n = \{A^{(\tau,\alpha,\zeta)} \mid A \in V_{\text{N}}, \tau \in V_I \cup V_A, \alpha \text{ an address in } \tau \text{ with } \mathsf{label}(\tau, \alpha) = A, \zeta \in \{\text{b}, \text{t}\}\}$$

To define the set of index symbols Υ, we let one index symbol represent each dominance link in V_I or V_A. Specifically, let Υ be $\{l_{v,\eta_1,\eta_2} \mid \mathsf{dom}_v(\eta_1,\eta_2), v \in V_I \cup V_A\}$. We will assume that active dominance requirements are associated with the bottom version of the nodes, while the passive dominance links are associated with the top version. This will allow dominance requirements associated with a node in a tree to be fulfilled by trees that are adjoined into that node (or into the tree adjoined into that node, and so on).

Now we define the transition relation δ. As before, we give mnemonic names (which refer to TAG operations) to the different types of transitions for subsequent easy reference; these names will be given in **boldface**, while the names of formal operations of the {}-BEPDA (in terms of which δ is of course defined) will, as previously, be given in CAPS.

1. **Terminal Shift:** For every frontier node of τ that is labeled with a terminal symbol, add a SHIFT of that symbol. Specifically, if $a \in \Sigma$ is a terminal on the frontier of τ, add $(q_a, (\llbracket, \emptyset))$ to $\delta(q, \varepsilon, \varepsilon, (\varepsilon, \emptyset), \varepsilon)$ and $(q, (a, \emptyset))$ to $\delta(q_a, a, \varepsilon, (\varepsilon, \emptyset), \varepsilon)$.

2. **Reduce:** Let η be a node labeled A in τ with address α, and let its daughters be $\eta_1, \ldots, \eta_n$, such that η_i is labeled $X_i \in V_\mathrm{T} \cup V_\mathrm{N}$, $1 \leq i \leq n$. Suppose the kth daughter dominates the footnode; set $k = 1$ if no daughter dominates the footnode (or if τ is an initial tree). For each such node η, add the following UNWRAP move. The $n-k+1$st stack from the top is the designated stack. Remove the $k-1$ stacks $\llbracket X_1^{(\tau,\alpha\cdot 1,t)} \cdots \llbracket X_{k-1}^{(\tau,\alpha\cdot(k-1),t)}$ from below the designated stack, the $n-k$ stacks $\llbracket X_{k+1}^{(\tau,\alpha\cdot(k+1),t)} \cdots \llbracket X_n^{(\tau,\alpha\cdot n,t)}$ from above the designated stack, and symbol $X_k^{(\tau,\alpha\cdot k,t)}$ from the top of the designated stack. Then push $A^{(\tau,\alpha,b)}$ on top of the designated stack. The index set of the designated stack will be the union of $\top(\eta)$ and of the index sets of each η_i, from which have been removed the symbols in $\bot(\eta_i)$. Formally, add $(q, (A^{(\tau,\alpha,b)}, \top(\eta)))$ to $\delta(q, \varepsilon, (\llbracket X_1^{(\tau,\alpha\cdot 1,t)}, \bot(\eta_1)) \cdots (\llbracket X_{(k-1)}^{(\tau,\alpha\cdot(k-1),t)}, \bot(\eta_{k-1})), (X_k^{(\tau,\alpha\cdot k,t)}, \bot(\eta_k)), (\llbracket X_{k+1}^{(\tau,\alpha\cdot(k+1),t)}, \bot(\eta_{k+1})) \cdots (\llbracket X_n^{(\tau,\alpha\cdot n,t)}, \bot(\eta_n)))$.

3. **ε-Reduce:** Let η be a node labeled A in τ with address α and a single daughter labeled ε. For each such node η, NEW-PUSH the stack/multiset pair $(\llbracket A^{(\tau,\alpha,b)}, \top(\eta))$ onto the stack of stacks.

4. **Adjoin:** For every node η labeled A at address α at which an adjunction of tree τ' (whose footnode η' has address α') can take place, add a PUSH move of the (bottom version of the) foot node of the adjoined tree. The index set of the top stack will be its previous index set with $\top(\eta')$ added. Formally, add IF $A^{(\tau,\alpha,b)}$ PUSH $(A^{(\tau',\alpha',b)}, \top(\eta'))$ to δ. If the node is marked with the Null-Adjoining (NA) constraint, or with a Selective-Adjoining constraint (SA) which does not include τ', then the rule is not added to δ.

5. **Unadjoin:** For every root node η labeled A of an auxiliary tree τ, POP the stack symbol $A^{(\tau,\varepsilon,t)}$ that represents the root node and the index symbols $\bot(\eta)$. This in effect removes any trace of the adjoined tree from the automaton.

6. **Noadjoin:** For every node labeled A at address α that is not marked with the Obligatory-Adjoining (OA) constraint, add an UNWRAP move that changes the stack symbol from the bottom node to the top node. Formally, $(q, (A^{(\tau,\alpha,t)}, \emptyset)) \in \delta(q, \varepsilon, \varepsilon, (A^{(\tau,\alpha,b)}, \emptyset), \varepsilon)$.

In addition, if we use the "extended definition" mentioned above, in which each stack symbol (and not only each stack) is associated with an index set, we have a **raise** move, which moves an index symbol up from an index set of a stack symbol to the index set of a stack symbol above it in the same stack. The **lower** move is defined similarly. We omit formal definitions, since these moves are not required for the formal construction of a {}-BEPDAform a V-TAG.

For an example, we refer to Section 6.3, page 214.

Equivalence Between V-TAG, []{}-LIG, and {}-BEPDA

We now conclude the proof of equivalence which we started with Lemma 29 by proving two further inclusion lemmas.

Lemma 30 $\mathcal{L}$(V-TAG) $\subseteq$ $\mathcal{L}$({}-BEPDA). *Specifically, the* {}-BEPDA *constructed by the easy method from a* V-TAG *accepts exactly the language the grammar generates.*

Proof. Let G be a strict V-TAG and M the {}-BEPDA derived from G by the easy method. We will use induction on the number of dominance links used during the derivation to show that $L(G) = N(M)$.

We first show $L(G) \subseteq N(M)$. Specifically, we show that for all integers k, for all V-TAGs G, if there is a derivation in G of string w that uses vectors containing a total of k dominance links, then there is a run of the {}-BEPDA constructed by the easy method from G that accepts w (by empty stack) and that removes a total of k symbols from index sets during the run (and, hence, inserts k symbols as well).

The base case ($k = 0$) follows from the equivalence of TAG and BEPDA, and the observation that the easy method for constructing {}-BEPDAs from V-TAGs extends the easy method for the construction of BEPDAs from TAGs.

For the induction step, assume that we have a derivation ϱ in G of a string w which uses vectors containing a total of k dominance links. We form a new grammar G' which is identical to G, except as follows. Let v be a vector in V_I or V_A that is used during ϱ and that includes at least one dominance link, say link l between nodes η_1 and η_2. We add a new vector v' to V_I' or V_A', as the case may be, which is identical to v, except that we remove $l = \mathsf{dom}_v(eta_1, \eta_2)$ from dom_v to form $\mathsf{dom}_{v'}$. If v' now is such that not all trees are connected to all other trees, we modify v' by distributing the trees in v' among two vectors, v' and v'', such that in each of these vectors all trees are connected to all other trees (but of course to none in the other vector). We see that G' is strict as well. Clearly there is a derivation ϱ' in G' of w, which is identical to ϱ, except that ϱ' uses v' (and perhaps v'') where ϱ uses v. Thus, ϱ' uses vectors containing a total of $k-1$ dominance links. By induction hypothesis, the automaton M' constructed from G' by the easy method can recognize w in a run that removes k index symbols. From inspection of the easy method, it can be seen that M' and M are identical, except that M' has additional moves derived from vector v' (and perhaps v''). Specifically, M' has an additional **Reduce** or **Adjoin** move associated with η_2 (depending on whether η_2 is not or is a footnode of its tree), in which it does not add symbol l_{v,η_1,η_2} to the index multiset associated with the top stack after the move, and

M' has an additional **Adjoin** move associated with η_2, in which he same symbol is not removed from the index multiset of the top stack. But since η_1 does in fact dominate η_2 in the derived tree of ϱ and hence ϱ', we have a run of M' which does not use the rules derived from v', but those derived from v. But then the run of M' is also a valid run of M, and we are done.

We now show $N(M) \subseteq L(G)$. Specifically, we show that for all {}-BEPDAs M constructed by the easy method from a strict V-TAG G, for all integers k, if there is a run of M that accepts string w and that removes a total of k symbols from index sets during the run (and, hence, inserts k symbols as well), then there is a derivation in G of w which uses vectors containing a total of k dominance links.

The base case ($k = 0$) follows from the equivalence of TAG and BEPDA, and the observation that the easy method for constructing {}-BEPDAs from V-TAGs extends the easy method for the construction of BEPDAs from TAGs.

For the induction step, assume that we have a run μ of the automaton M which accepts w such that during μ k index symbols are added and then removed from index multisets. Construct a new automaton M' as follows. Choose an **Adjoin** rule in M which is used in μ and which removes at least one index symbol l_{v,η_1,η_2} for some nodes η_1, η_2 from the index set. Add a new **Adjoin** rule to M' that is like the old one, except that symbol l_{v,η_1,η_2} is not removed from the index set. Determine a rule which may introduce the index symbol in question in μ, and add to M' a rule that is like the old one, except that symbol l_{v,η_1,η_2} is not added to the index set. Clearly, there is a run of M' which accepts w but which removes a total of only $k-1$ index symbols from the set. From inspection of the "easy method", it can easily (nicht wahr) be seen that there is a grammar G' such that M' is the unique {}-BEPDA derived from it. G' is like G, except that dominance link l_{v,η_1,η_2} has been removed from some vector v to form a new vector v', which has been added to G'. (If necessary, we divide v' into two vectors v' and v'' to preserve strictness as in the converse direction.) By induction hypothesis, there is a derivation ϱ' in G' of w which uses vectors with a total of $k-1$ dominance links. Since in the derived tree of ϱ' η_1 in fact dominates η_2, we have a second derivation ϱ of w which is like ϱ', except that instead of using v' (and perhaps v'') as does ϱ', it uses v. Clearly, this derivation uses k dominance links. Furthermore, since ϱ does not use v' (or v''), it is also a valid legal derivation in G, and we are done.

We conclude that $N(M) = L(G)$. ■

Lemma 31 $\mathcal{L}(\{\}\text{-BEPDA}) \subseteq \mathcal{L}([]\{\}\text{-LIG})$

Proof. Let $G = (Q, \Sigma, \Gamma, \Upsilon, \delta, q_0, Z_f)$ be a {}-BEPDA. We construct a []{}-LIG $G' = (V_{\mathrm{N}}, V_{\mathrm{T}}, V_{[\mathrm{I}]}, V_{\{\mathrm{I}\}}, P, S)$, where:

- $V_{\mathrm{N}} = \{\langle q_1, q_2\rangle \mid q_1, q_2 \in Q\} \cup \{S\}$
- $V_{\mathrm{T}} = \Sigma$
- $V_{[\mathrm{I}]} = \Gamma$
- $V_{\{\mathrm{I}\}} = \Upsilon$

The set P contains the following productions (and no others):

1. $S[\cdots]\{\} \longrightarrow \langle q_0, q\rangle[\cdots]\{\}$ for all q in Q;
2. $\langle q, q_{m+1}\rangle[\cdots\xi]s \longrightarrow a\langle q_1, q_2\rangle[\xi_1]s_1 \cdots \langle q_{k-1}, q_k\rangle[\xi_{k-1}]s_{k-1}\langle q_k, q_{k+1}\rangle[\cdots\xi_k]s_k \langle q_{k+1}, q_{k+2}\rangle[\xi_{k+1}]s_{k+1} \cdots\langle q_m, q_{m+1}\rangle[\xi_m]s_m$ for all $q, q_1, \ldots, q_{m+1}$ in Q, a in $\Sigma \cup \{\varepsilon\}$, $\xi, \xi_1, \ldots, \xi_m$ in Γ^*, and $s, s_1, \ldots, s_m$, multisets of members of Υ, such that $\delta(q_1, a, (\blacksquare\xi_1, s_1)\cdots(\blacksquare\xi_{k-1}, s_{k-1}), \xi_k s_k, (\blacksquare\xi_{k+1}, s_{k+1})\cdots(\blacksquare\xi_m, s_m))$ contains $(q, (\xi, s))$.

We will show by induction on the number of steps n that there is a derivation ϱ in G of the form $\langle q, q'\rangle[\xi]t \stackrel{n}{\Longrightarrow}_G w_1$ iff there is a run ϱ' of M of the form $(q, \gamma, \varepsilon, w_1 w_2) \stackrel{n}{\vdash}_M (q', \gamma(\blacksquare\xi, t), w_1, w_2)$ for all integers m, n, $\xi \in \Gamma^*$, t a multiset of elements of Υ, $w_1, w_2 \in V_{\mathrm{T}}^*$, $\gamma \in (\blacksquare\Gamma^* \times \mathcal{M}(\Upsilon))^*$, and $q, q' \in Q$.

We first show the "only if" direction.

The base case ($n = 0$) is trivial, since vacuous.

For the induction step, assume that we have a derivation ϱ in G of the form $\langle q_1, q_2, \rangle[\xi\xi']t \stackrel{n}{\Longrightarrow}_G w$, with the symbols as defined above, and ξ' also in Γ^*. Let $q_1 = q, q_2, \ldots, q_m \in Q$ and let the first production used in the derivation be $p : \langle q, q'\rangle[\cdots\xi']s \longrightarrow a\langle q_1, q_2\rangle[\xi_1]s_1 \cdots \langle q_{k-1}, q_k\rangle\ [\xi_{k-1}]s_{k-1}\langle q_k, q_{k+1}\rangle[\cdots\xi'_k]s_k\langle q_{k+1}, q_{k+2}\rangle[\xi_{k+1}]s_{k+1}\cdots\langle q_m, q'\rangle[\xi_m]s_m$. Let $\xi_k = \xi\xi'_k$. We then have $w = w_1 \cdots w_m$, $w_1, \ldots, w_m \in V_{\mathrm{T}}^*$, such that $\langle q_i, q_{i+1}\rangle[\xi_i]t_i \stackrel{n_i}{\Longrightarrow}_G w_i$, $1 \leq i \leq m$, with $n_i < n$, such that $t = \cup_{i=1}^m (t_i \setminus s_i) \cup s$. By induction hypothesis there are runs ϱ_i of the automaton of the form $(q_i, \gamma_i, \varepsilon, w_i) \stackrel{n_i}{\vdash}_M ((q_{i+1}, \gamma_i(\blacksquare\xi_i, t_i), w_i, \varepsilon)$, for γ_i arbitrary stack configurations. Runs $\varrho_1, \ldots, \varrho_m$ can be executed successively, yielding the run ϱ'' $(q_1, \gamma, \varepsilon, w_1 \cdots w_m) \stackrel{n-1}{\vdash}_M ((q_m, \gamma(\blacksquare\xi_1, t_1)\cdots(\blacksquare\xi_m, t_m), w_1 \cdots w_m, \varepsilon)$, for some stack configuration γ. By construction of G, we have a rule of the form $(q, (\xi, s)) \in \delta(q_1, a, (\blacksquare\xi_1, s_1) \cdots(\blacksquare\xi_{k-1}, s_{k-1}), \xi_k s_k, (\blacksquare\xi_{k+1}, s_{k+1})\cdots(\blacksquare\xi_m, s_m))$. Since $t = \cup_{i=1}^m (t_i \setminus s_i) \cup s$, we conclude that we have a derivation of the form $(q, \gamma, \varepsilon, w_1 w_2) \stackrel{n}{\vdash}_M ((q', \gamma(\blacksquare\xi\xi', t), w_1, w_2)$, and we are done.

The "if" direction is shown analogously; we omit the details. ■

The following two theorems and one corollary now immediately follow from Lemmas 29, 30, and 31.

Theorem 32 $\mathcal{L}$(V-TAG) = $\mathcal{L}([]\{\}$-LIG)

Theorem 33 $\mathcal{L}$(V-TAG) = $\mathcal{L}(\{\}$-BEPDA)

Corollary 34 $\mathcal{L}([]\{\}$-LIG) = $\mathcal{L}(\{\}$-BEPDA)

4.6.5 Formal Properties of V-TAG

Theorem 35 *1.* $\mathcal{L}_{\mathcal{R}}$(V-TAG) $\subseteq \mathcal{L}$(CSG).

2. $\mathcal{L}_{\mathcal{R}}$(V-TAG-$\Delta$) $\subseteq \mathcal{L}$(CSG).

Proof. (i) Let G be a V-TAG, and M_G the {}-BEPDA derived from it by the easy construction. We construct a linear bounded automaton M which accepts $L_R(G)$. The construction will only be given informally; the formal details can easily be worked out. M executes a run of M_G. It has as its workspace the input string w plus a number of blank tape squares linearly bounded in $|w|$. M nondeterministically simulates a run of M_G by recording the configuration of the stack of stacks (including index sets) and the state. Since the number of vectors used in the derivation is linearly bounded, the number of trees adjoined during the derivation is linearly bounded as well, as is the number of dominance links used. Therefore, the number of stack and index symbols that M needs to write during the simulation is also linearly bounded.

(ii) This case is shown analogously. (The integrity constraints are coded in the finite-state control of the automaton along with the rest of the grammar.) ■

Since lexicalized V-TAGs are linearly restricted, we immediately obtain the following corollary:

Corollary 36 $\mathcal{L}$(V-TAG-Δ_{Lex}) $\subseteq \mathcal{L}$(CSG)

4.6.6 Parsing V-TAG

We show that the type of V-TAG we are interested in for linguistic use, lexicalized V-TAG, is polynomially parsable.[23]

Theorem 37 *Let q be a polynomial function over the integers. Then a q-form restricted* V-TAG *is parsable in deterministic polynomial time.*

Proof. Let G be a V-TAG. We use the CKY parser for TAG defined by Vijay-Shanker (1987, p.110). Unlike Vijay-Shanker, we split every node into a top and a bottom version, as we did for the easy construction of the {}-BEPDA (Section 4.6.4, page 115). This gives us the following set of nodes:

$$\Gamma_n = \{A^{(\tau,\alpha,\zeta)} \mid A \in V_{\text{N}}, \tau \in V_I \cup V_A, \alpha \text{ an address in } \tau \text{ with } \mathsf{label}(\tau,\alpha) = A, \zeta \in \{\text{b},\text{t}\}\}$$

The five cases he defines (p.108ff) are augmented as follows. In Cases 1 through 4 (corresponding to context-free expansions in elementary trees), nodes μ_1 and μ_2 are the top versions, and μ is the bottom version of the nodes in question. In Case 5, μ_1 is the bottom version, μ_2 the top version, and the resulting node is the top version of μ_1. We need an additional case (we use T for the recognition matrix, not A):

Case 6: If a node $\mu_1 = A^{(\tau,\alpha,\text{b})} \in T[i,j,k,l]$, then
$\mu_2 = A^{(\tau,\alpha,\text{t})} \in T[i,j,k,l]$.

[23] This section is based on joint work with Tilman Becker. A less terse discussion can be found in (Becker and Rambow, 1994).

It is clear how to restrict these cases to implement the adjunction constraints. We model the dominance links in a manner similar to the parser for UVG-DL (Section 4.4.4, page 94). Let a link-counter L be an array whose elements are indexed on the dominance links of G, and whose values are integers. The sum, norm, and less-than relation are defined as for $\top$-counters in Section 4.4.4. The elements in the sets in the four-dimensional T matrix are pairs consisting of a node and an L-counter. An entry (η, γ) in $T[i,j,k,l]$ means that there is a partial derivation in which the node η in a (perhaps derived) auxiliary tree dominates the substring $a_{i+1} \cdots a_j \eta' a_{k+1} \cdots a_l$, where η' is (the label of) the footnode of some auxiliary tree of G, with unfulfilled dominance links of γ. Or, if η is a node of an initial tree, an entry (η, γ) in $T[i,j,j,l]$ means there is a partial derivation in which the η in a (perhaps derived) initial tree dominates the substring $a_{i+1} \cdots a_{l+1}$ with unfulfilled dominance links of γ (which must of course be positive). Let l be a link. If $\gamma(l)$ is positive, then the unfulfilled requirement is for active dominance, if it is negative, it is for passive dominance.

The parser is identical to Vijay-Shanker's except that we need to specify which actions are performed on the L-counters by the various cases. Recall that the multiset of unfulfilled active dominance requirements of a node η is denoted by $\top(\eta)$, and the multiset of all passive dominance requirements is denoted by $\bot(\eta)$.

- Cases 1 through 4. We let $\gamma_\mu := \gamma_{\mu_1} + \gamma_{\mu_2} + \top(\eta)$. (In unary cases, we omit the γ_{μ_2}).
- Case 5. Let $\mu_1^{(\mathrm{b})}$and $\mu_1^{(\mathrm{t})}$ denote the bottom and top versions of μ_1, respectively. Then if μ_1 dominates a footnode, we have:

  $$\gamma_{\mu_1^{(\mathrm{t})}} := \gamma_{\mu_1^{(\mathrm{b})}} + \gamma_{\mu_2} + \bot(\mu_1)$$

  If μ_1 does not dominates a footnode, we have:

  $$\gamma_{\mu_1^{(\mathrm{t})}} := \gamma_{\mu_1^{(\mathrm{b})}} + \gamma_{\mu_2}$$

  but the move is only valid if $\gamma_{\mu_1^{(\mathrm{b})}} + \gamma_{\mu_2} \geq 0$.
- Case 6. We set $\gamma_{\mu_1^{(\mathrm{t})}} := \gamma_{\mu_1^{(\mathrm{b})}} + \bot(\mu_1)$.

In all six cases, after calculating the new γ, the entry is discarded if $|\gamma| \geq q(n)$.

The recognition of a string $a_1 \cdots a_n$ is successful if for some j, $0 \leq j \leq n$, and some η, a root node of an initial tree, we have $(\eta^{(\mathrm{t})}, \gamma_0) \in T[0,j,j,n]$.

The correctness of the recognition algorithm for TAG is proven by Vijay-Shanker (1987). It can easily be seen by induction on the number of dominance links that the L-counters correctly impose the dominance constraints.

The time complexity of the algorithm is that of Vijay-Shanker's algorithm, $O(n^6)$, multiplied by a factor representing the maximal number of elements of each cell of matrix T. Since $|\gamma| \leq q(n)$, we have that the number of possible L-counters is bounded by $O(q(n)^{|L|})$ (where $|L|$ is the total number of links in G), and the the time complexity of the algorithm is in $O(|G| q(n)^{2|L|} n^6)$, which is polynomial in n if q is. ■

We construct a parse forest using back pointers.

The following corollary follows.

Corollary 38 V-TAG-Δ_{Lex} *is polynomially parsable.*

4.7 Relation to Other Formalisms

In this section, we discuss the relation between the formalisms developed in this chapter and other formalisms that have been proposed in computational linguistics. Section 4.7.1 discusses certain extensions to categorial systems, specifically, to Steedman's CCG. Section 4.7.2 compares UVG-DL to "quasi-trees", a generalization of trees proposed by Vijay-Shanker. Section 4.7.3 suggests ways of using results from this chapter to limit the formal and computational power of unification-based formalisms such as HPSG. We close in Section 4.7.4 with a brief remark on dependency grammars. Recall that we have discussed Reape's word-order domains in Section 3.3.6, page 51, since they were found to be similar in spirit (though not formally equivalent) to FO-TAG. We will discuss Karttunen's "radical lexicalism" and Uszkoreit's GPSG-based approach in Section 5.6, page 194, since these systems can fruitfully be compared to the formalisms presented in this chapter only after discussing linguistic issues pertaining to them.

4.7.1 {}-CCG

In categorial grammars, terminal symbols are associated with categories, which can either be elementary, or complex categories that represent functors from a specified type into another specified type. In the categorial grammar (CG) of Ajdukiewicz (1935) and Bar-Hillel (1953), derivations consist in applying functional categories to other categories (*function application*). The resulting system is weakly equivalent to CFG. Steedman (1985) extends CG by allowing further combinatory operations on categories, in particular *function composition*.[24] In function composition, the arguments of one functor are added to the end of the list of argument categories of another functor. This means that categories in a derivation can grow unboundedly, but that the structure of the derived categories is always that of a stack. This extremely informal description of CCG suggests that is weakly equivalent to LIG, which in fact was shown by Weir (1988).

Hoffman (1992) discusses the implementation of a grammar for Turkish in CCG. Like German, the order of arguments of a verb is free, but unlike German, arguments can also appear behind the verb. In cases of embedded clauses, arguments of embedded verbs may leave their clause. This leads to two problems. First, it is unclear whether CCG can handle all long-distance scrambling that is observed in Turkish. Second, even in the case of purely local (clause-internal) scrambling, the fact that CCG categories encode both subcategorization frame and word order means that a large number of categories must be associated with each verb. This is linguistically unappealing. Instead, Hoffman extends CCG by allowing (multi-)sets of argument types, rather than just argument types. Furthermore, the directionality can remain unspecified. In this manner a transitive

[24] *Type raising* is limited to the lexicon and therefore is not of interest formally.

verb can be associated with the category S | {$\mathrm{NP}_{\mathrm{NOM}}$, $\mathrm{NP}_{\mathrm{ACC}}$ }. This system she calls {}-CCG (set-CCG). The definition of function composition is extended to function categories whose arguments are sets in such a way that the union of these sets is formed. This is illustrated in the following example, taken from (Hoffman, 1994) (we omit the semantics).

(2) a. Esra'nın$_i$ Fatma [t_i gittiğini] biliyor.
Esra-GEN$_i$ Fatma [t_i go-GER-3SG-ACC] know-PROG.
Fatma knows that Esra left.

b.
$$\dfrac{\dfrac{\text{Esra-gen}\quad \text{Fatma}\quad \dfrac{\text{go-gerund-acc}\quad \text{knows.}}{S \mid \{N_{\mathrm{GEN}}\ ,N_{\mathrm{NOM}}\ \}}<\mathrm{B}}{S \mid \{N_{\mathrm{GEN}}\ \}}<}{S}<$$
N_{GEN} N_{NOM} $S_{Na} \mid \{N_{\mathrm{GEN}}\ \}$ $S \mid \{N_{\mathrm{NOM}}\ , S_{Na}\}$

Here, the argument sets of the two verbs are merged, and the resulting complex functor can be applied to the arguments of both verbs in any order. This brief and highly informal description motivates the conjecture that {}-CCG is weakly equivalent to {}-LIG.[25] While these issues requires further study (and the conjecture awaits a proof), the similarity between the two approaches to free word order in a categorial and a traditional phrase-structure approach are readily apparent.

4.7.2 Quasi-Trees

This section discusses the relation between UVG-DL and "Quasi-Trees" as introduced by Vijay-Shanker (1992). Vijay-Shanker starts from the observation that the traditional definition of TAG is incompatible with a unification-based approach because the trees of a TAG start out as fully specified objects, which are later modified; in particular, immediate dominance relations that hold in a tree need not hold after another tree is adjoined into it. In order to arrive at a definition that is compatible with a unification-based approach, he makes three minimal assumptions about the nature of the objects used for the representation of natural language syntax:

- The first assumption (left implicit) is that these objects represent phrase-structure.
- The second assumption is that they "give a sufficiently enlarged domain of locality that allows localization of dependencies such as subcategorization, and filler-gap" (Vijay-Shanker, 1992, p.486).
- The third assumption is that dominance relations can be stated between different parts of the representation.

[25] In fact, the equivalence will only hold if all arguments of functors *must* be sets. If there is an option of using the curried notation of simple CCG, as in fact proposed by Hoffman (1992) for {}-CCG, then the system will be equivalent to a more powerful system, perhaps to []{}-LIG (and hence to V-TAG).

These assumptions lead Vijay-Shanker to define quasi-trees, which are partial descriptions of trees in which "quasi-nodes" (partial descriptions of nodes) are related by dominance constraints. (Linear precedence between nodes is fully specified, though one could imagine a formulation in which linear precedence may also be underspecified.) There are two ways of interpreting quasi-trees: either quasi-trees can be seen as data structures in their own right; or quasi-trees can be seen as descriptions of trees whose denotations are sets of (regular) trees.[26] If quasi-trees are defined as data structures, we can define operations such as adjunction and substitution and notions such as "derived structure". If quasi-trees are defined as descriptions, we can instead conjoin the descriptions (and perhaps add some equations) and investigate the resulting denoted set of trees. In fact, the minimal tree ("minimal" in some sense to be defined) in this set will correspond to the structure derived by substitution and adjunction in the data-structure perspective. While Vijay-Shanker (1992) is biased towards the description-based interpretation, his exposition does not exclude the data-structure interpretation. We will adopt the data-structure interpretation of quasi-trees here, since it will greatly facilitate the comparison with the formalisms elaborated in this thesis; we leave a precise discussion of the relation between the two interpretations of quasi-trees to future research.

We will therefore define quasi-trees to be structures consisting of pairs of nodes, called quasi-nodes, such that one is the "top" quasi-node and the other is the "bottom" quasi-node. The top and bottom quasi-node of a pair are linked by a dominance constraint. Bottom quasi-nodes immediately dominate top quasi-nodes of other quasi-node pairs, and each top quasi-node is immediately dominated by exactly one bottom quasi-node. For simplicity, we will assume that there is only a bottom root quasi-node (no top quasi-node), and that bottom frontier quasi-nodes (those bottom quasi-nodes that do not immediately dominate any top quasi-nodes) are omitted (i.e., there are only top quasi-nodes on the frontier). Furthermore, we will assume that each quasi-node has a label, and is equipped with a finite feature structure. A sample quasi-tree is shown in Figure 4.15 (quasi-tree α_5 of Vijay-Shanker (1992, p.488)).

We now need to define combination operations. We follow Vijay-Shanker (1992, Section 2.5) in defining *substitution* as the operation of forming a quasi-node pair from a frontier node of one tree (which becomes the top node) and the root node of another tree (which becomes the bottom node). As always, a dominance link relates the two quasi-nodes of the newly formed pair. Adjunction need not, in fact, be defined separately: it suffices to say that a pair of quasi-nodes is "broken up", thus forming two quasi-trees. We then perform two substitutions. The two previously paired quasi-nodes now form new pairs with quasi-nodes of the "adjoined" tree, as long as the dominance relation that relates the quasi-nodes of the original pair is maintained. Under this view, adjunction reduces to two substitutions. Observe that nothing keeps us from breaking up more than one pair of quasi-nodes in either of two quasi-trees, and then performing more than two substitutions (as long as dominance constraints are respected); there are no operations in regular TAG that correspond to such operations.

We now turn to the definition of derived structures. First, observe that the result of a (complex) substitution, even if it involves breaking up pairs of quasi-nodes, is a quasi-tree. We will say that a quasi-tree is *derived* if in all quasi-node pairs, the two quasi-nodes are equated, meaning that their label is the same and the two feature structures are unified, and furthermore, if all frontier

[26] The interpretation of feature structures presents a similar choice.

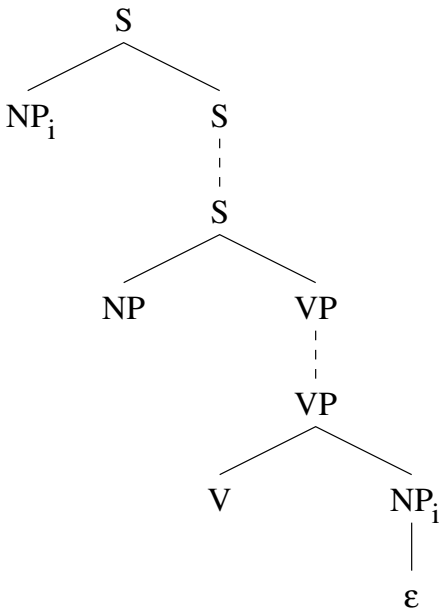


Figure 4.15: Sample quasi-tree

quasi-nodes have terminal labels. The string associated with this quasi-tree is defined in the usual way.

We have now fully defined a formalism (if informally): its data structures (quasi-trees), its combination operation (substitution), and the notion of derived structure. We will call this formalism Quasi-Tree Substitution Grammar (QTSG). It can easily be seen that all examples discussed by Vijay-Shanker (1992) are derivations in QTSG (where the grammars are just the set of trees involved in the example). The question arises as to the formal and computational properties of QTSG.

Quasi-trees are very similar to UVG-DL. Left-hand side nonterminals correspond to bottom quasi-nodes, and right-hand side nonterminals correspond to top quasi-nodes. Thus, Figure 4.15 can also be seen as a set in a UVG-DL.[27] The requirement that labels match and feature structures unify in the derived quasi-tree corresponds to the rewriting step in UVG-DL.[28] Since it is clear that QTSG and UVG-DL are essentially notational variants, we will not define QTSG formally, nor prove the equivalence formally. We can now transfer the formal results obtained for lexicalized UVG-DL, in particular polynomial parsability and inclusion in $\mathcal{L}$(CSG), to lexicalized QTSG. Conversely, we can see that UVG-DL minimally implements the three assumptions that underlie quasi-trees: the choice of string-rewriting rules reflects the choice of phrase-structure representations; the choice of sets of such rules reflects the desire for an extended domain of locality; and dominance

[27] There are two complicating factors. First, in UVG-DL, the dominance relation in an initial set need not define a tree (as they do in a QTSG. However, we can see the UVG-DL sets as a shorthand notation for all possible tree-like structures that are compatible with it, of which there must always be a finite number. Second, as defined by Vijay-Shanker (1992), quasi-trees may contain "normal" nodes. However, since no operation (substitution or adjunction) can be performed at these nodes, they do not alter the formal properties of the system and are just a notational device.

[28] We omit the details involving the role of the start symbol.

links clearly reflect the linguistic need to express dominance constraints. Interestingly, we have motivated UVG-DL not from more abstract methodological considerations, as quasi-trees are by Vijay-Shanker, but by an investigation of a certain set of linguistic data. The fact that the two approaches converge on equivalent representations we take to mean that both approaches are on the right track in capturing both intuitions about linguistic data and meta-intuitions about the proper representation for natural language syntax.

However, it should be noted that we have found UVG-DL lacking in two respects: first, we have found the need for integrity, and second, we have argued (though not proved) that certain formal languages relevant to natural language syntax cannot be derived in a UVG-DL, in particular, the copy language $\{ww|w \in \{a,b\}^*\}$ (Conjecture 7, page 78). Presumably, quasi-trees have the same shortcomings. While the first point is a technical detail (one could include some form of integrity in the definition of quasi-trees), the second point is a more profound issue. Vijay-Shanker claims that quasi-trees represent the minimal methodological assumptions that need be made for the description of natural language syntax. If, however, quasi-trees are not able to generate Dutch cross-serial dependencies, then the formal requirements for the adequate representation of natural language syntax go beyond those that can be derived from these purely methodological considerations. Data from specific natural languages must be taken into account. We may conclude that the operation of adjunction in TAG then represents something more than merely inserting structure into another structure while preserving the partial descriptions. Put differently, the notion of *tree rewriting* defined in analogy to string rewriting cannot be replaced by the operation of substitution, just as the operation of string rewriting cannot be replaced by the operation of concatenation. This is expressed particularly clearly in the two types of LIG, stack-valued []-LIG and multiset-valued {}-LIG. The stack corresponds to single tree rewriting, and the set corresponds to multi-component substitution. If Conjecture 7 is correct, one data structure cannot simulate the operations of the other, and both are therefore needed. However, this question requires further study, and it is not clear that the notion of adjunction as defined for regular TAG is the only solution to the problem of the copy language (and the Dutch dependencies).

4.7.3 HPSG and Unconstrained Unification-Based Formalisms

HPSG (Pollard and Sag, 1987; Pollard and Sag, 1994) uses unconstrained feature structures as its formal basis, which are Turing-equivalent. However, it is not necessarily the case that the full power of the system is used in the linguistic analyses that are expressed in it. HPSG analyses include a "backbone" which is a context-free phrase-structure tree. In addition, various mechanisms have been proposed to handle certain linguistic phenomena that relate two nodes within a tree. One of these is a multiset-valued feature that is passed along the phrase-structure tree from daughter node to mother node. Multiset-valued features have been proposed for the SLASH feature which handles *wh*-dependencies (Pollard and Sag, 1991, Chapter 4), and for certain semantic purposes, including the representation of stored quantifiers in a mechanism similar to Cooper-storage. Another use may be the representation of anti-coreference constraints arising from Principle C of Binding Theory (be it that of (Chomsky, 1981) or of Pollard and Sag (1991)). If an interesting fragment of HPSG can be isolated that uses only the context-free backbone and multiset-valued feature structures, then such a system can be modeled with {}-LIG. If we can show that this model meets the various restrictions we have defined for {}-LIG (which appears to be a

reasonable expectation, given the fact that all long-distance phenomena are tied to lexical items), the formal results obtained in this chapter for restricted versions of {}-LIG can be transferred. This is important both theoretically and practically. Theoretically, it would be interesting if it turned out that the linguistic principles formulated in HPSG naturally lead to certain restricted uses of the unification-based formalism, and if these uses turned out to be formally equivalent to independently suggested frameworks. Clearly this would represent an important insight into the nature of grammatical competence, especially since HPSG, unlike TAG-based linguistic theories, is not based on a formally restricted system. Therefore, any formal restrictions that are derived from HPSG's linguistic principles and parameters appear less arbitrary (to linguists) than those that are assumed *prior* to the linguistic enterprise, as is the case in TAG-based linguistic theories. On the practical side, formal equivalences can guide the building of applications such as parsers for existing HPSG grammars.

In a related vein, it has been proposed that HPSG analyses can be "compiled" into TAGs in order to obtain a computationally more tractable system (Kasper, 1992), thus sidestepping the issue of building parsers for HPSG directly. The systems introduced in this chapter may serve as targets for compilations in cases in which the formal power of TAG is inadequate, such as free word-order effects, or the semantic phenomena mentioned in the previous paragraph which are currently given HPSG analyses using multiset-valued features. Becker (1993, Chapter 5.4) sketches in some detail how such a compilation process could work on a fragment of HPSG that uses multiset-valued features to handle long-distance scrambling in German. The target formalism he uses can be taken to be V-TAG. However, there may be advantages in choosing a LIG-based formalism as the target for compilation, or a string-rewriting formalism rather than a tree-rewriting formalism (i.e., UVG-DL rather than V-TAG). We leave these issues for further study.

4.7.4 Dependency Grammars

Dependency grammars (DG), first introduced by Tesnière (1959), have remained popular tools for the expression of syntax (Hudson, 1984; Mel'čuk, 1988; Sgall et al., 1986), and have received increasing attention lately because of their natural (and in fact, necessary) orientation towards the lexicon. Gaifman (1965) showed that projective DGs and CFGs are weakly equivalent; in a projective DG, the nodes of the tree can be ordered (on a line) without crossing arcs. However, it has always been acknowledged among DG syntacticians that certain natural language phenomena require non-projective DGs. Unfortunately, non-projective DGs appear to be very powerful formalisms, and while parsing algorithms have been proposed (Covington, 1990), their complexity has not been analyzed and does not seem to be polynomial. Therefore, the question arises whether one can characterize the non-projective DGs in a more fine-grained way, so as to be able to isolate certain linguistic phenomena that, while requiring non-projectivity, still allow for relatively restricted generative power and polynomial parsing algorithms. Rambow and Joshi (1994) investigate the relation between simple TAG and DG, and suggest that certain types of non-projectivity, including that arising from Dutch cross-serial dependencies, can be modeled in a TAG, and can therefore be parsed in polynomial time. It would be interesting to investigate how the non-projectivity arising form the word order variations discussed in this thesis could be modeled by the extensions to simple TAG proposed in this chapter. We leave this question to further research.

Chapter 5

A Grammar for German

This chapter discusses how the formal system developed in Chapter 4 can be used to account for the range of German data outlined in Chapter 2 in a principles-and-parameters type methodology.[1] The chapter is structured as follows. Section 5.1 summarizes the issues involved in using a mathematical formalism for the representation of linguistic theory. We continue in Section 5.2 by discussing, in broad strokes, possible models of grammar. We propose a model which we attempt to justify by appealing to methodological principles of parsimony. We relate this model to the formal systems developed previously. In Section 5.3 we describe in detail the theory of grammar that we propose to implement, and define principles and parameters that we express in terms of the underlying formal systems. We will call this theory of grammar "V-grammar". For the convenience of the reader, the principles and parameters of V-grammar are summarized in Section 5.3.7. Section 5.4 then presents a fragment of a V-grammar of German, which accounts for the linguistic data presented in Chapter 2. Section 5.5 sketches how the approach allows for parametric variation in two other Germanic languages, Yiddish and Dutch. Finally, Section 5.6 compares V-grammar to related linguistic theories.

5.1 Introduction: Mathematical Formalisms and Linguistic Theory

In Chapter 3, we analyzed the formal requirements of several syntactic phenomena of German. In Chapter 4, we proposed formal systems, in particular UVG-DL and V-TAG,[2] for the representation of grammatical structures from which the German phenomena can be derived. These formalisms were motivated generally by the desire to use a formalism that shares the extended domain of locality of TAG, and the specific features were derived from very specific requirements

[1] Often, "principles-and-parameters theory" is used a synonym for GB. This is of course misleading (if not preposterous), as the term "principles-and-parameters" designates a research methodology, and other theories of grammar, such as HPSG, also fall within this methodological paradigm.

[2] In this and the following sections, we will not make a distinction between a formalism and its variant with integrity, i.e., between UVG-DL and UVG-DL-Δ, or between V-TAG and V-TAG-Δ. Reference to a formalism will be understood to be to the version with integrity.

of the data.

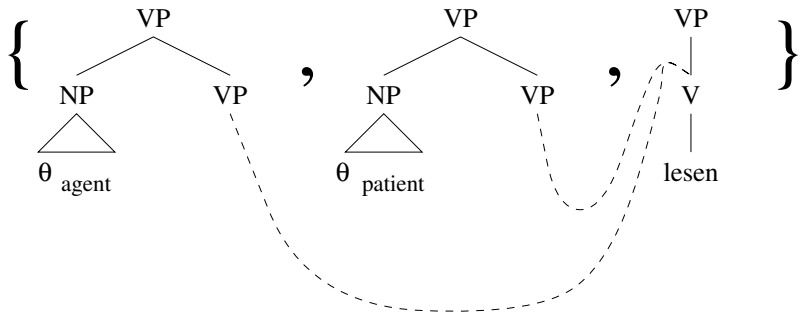


Figure 5.1: A lexical set for German

While we now have a formalism which we know can derive the data, we are still left with the question of how to derive the data in the formalism, which is the question of syntactic theory. Like TAG, V-TAG is not a syntactic theory, and there is a wide choice of possible ways of defining syntactic theories in the formalism. In fact, the choice is even wider in V-TAG than in TAG. Consider the representation in Figure 5.1. The formal analysis of the data shows that we need a tree set that looks somewhat like this one (perhaps among others) to allow for long-distance scrambling. Beyond that, we are free to choose node labels, tree shapes, tree sizes, and so on. (Figure 5.1 makes minimal assumptions.) But now consider the case of English. Since topicalization can be handled by simple TAGs, and since English lacks scrambling, we can simply use tree sets which contain a single tree, as shown in Figure 5.2. We have put the tree in a set, to indicate that we are using V-TAG. (We could of course simply use TAG for English, and V-TAG for German, and assume that languages are parametrized with respect to the formal system that their syntax is instantiated in, but that seems to be a particularly unappealing theory. Let us assume that V-TAG represents the syntax of all languages.) But we could also be tempted to assume that English, initially, looks like German: that except for directionality parameters (and of course the phonological realization of the lexemes), the English entry for *read* is the same as that for German *lesen*. This hypothesis is shown in Figure 5.3.

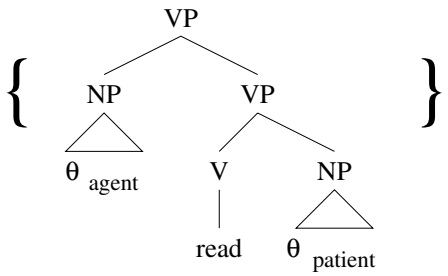


Figure 5.2: The "traditional analysis": a lexical tree for English

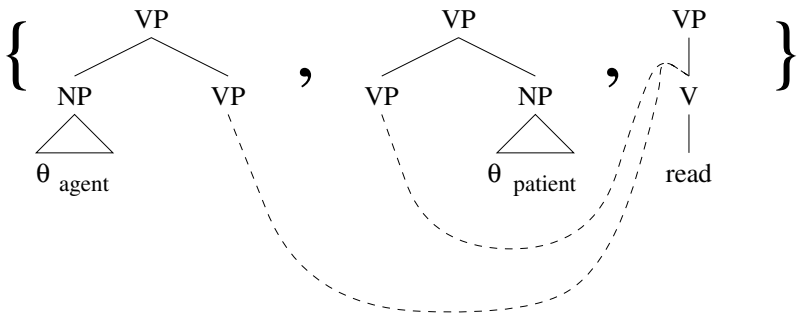


Figure 5.3: The alternative: a lexical set for English

What could possibly be the motivation for choosing what appears to be a more complex representation for English? Just as the German set allows us not only to derive multi-clausal word order variation but also clause-internal scrambling, we may expect that syntactic variation in English, such as passive, the double-object construction, or topicalization, could be derived from such an initial set. The promise that this approach holds is that we may be able to dispense with clause-internal transformations altogether, and derive all syntactic structures through operations defined in the underlying formal systems.

In this chapter, we will pursue an approach based on the idea that the lexical representation is universally like that in Figure 5.1 or Figure 5.3, and that all syntactic phenomena are derived within the mathematical system. Section 5.2 will motivate such an approach in a broad manner, while Section 5.3 will discuss the details of implementing a theory of grammar. We would, however, like to emphasize again that, while this particular choice of how to implement a grammatical theory in V-TAG is, perhaps, *suggested* by the formalism, it is not *determined* by it. Should the linguistic investigation in this chapter ultimately fail, it would not discredit the formal analysis that has been performed in the preceding chapters.

5.2 Theories of Grammar

5.2.1 Levels of Representation and Degrees of Freedom

Traditional transformation theories of grammar, such as the so-called Standard Theory and Government and Binding Theory (GB), assume that a deep structure of some sort (called D-Structure in GB), which includes lexemes and represents certain properties of these lexemes (such as valency), is transformed by some mechanism (or mechanisms) into a surface representation, from which a string in the language can be read off, or easily derived. In the Standard Theory of the seventies, the deep structure was generated by a component of the grammar consisting of context-free phrase-structure rules. This approach was abandoned in the GB theory of the eighties, since the transformations obviated the advantages of using a constrained formalism for the generation of the deep structure, while the formulation of the grammar in this framework did not allow interesting generalizations of the type that have now become commonly desired in principles-and-parameters

type methodologies such as GB. In a principles-and-parameters paradigm, the goal of linguistic description is to define a set of universal principles which are obeyed in all languages, and a set of parameters, whose different settings for different languages accounts for cross-linguistic variation. In GB, no language- or construction-specific transformations are allowed. Instead, any element of D-structure can move anywhere ("move-α"). Principles and parameters constrain the possible D-structures. Furthermore, D-structure is constrained by the lexical properties of the lexemes that it contains. Principles and parameters also constrain the possible applications of move-α either directly (constraints on movement) or indirectly (through constraints on S-structure).[3] This is represented in Figure 5.4.

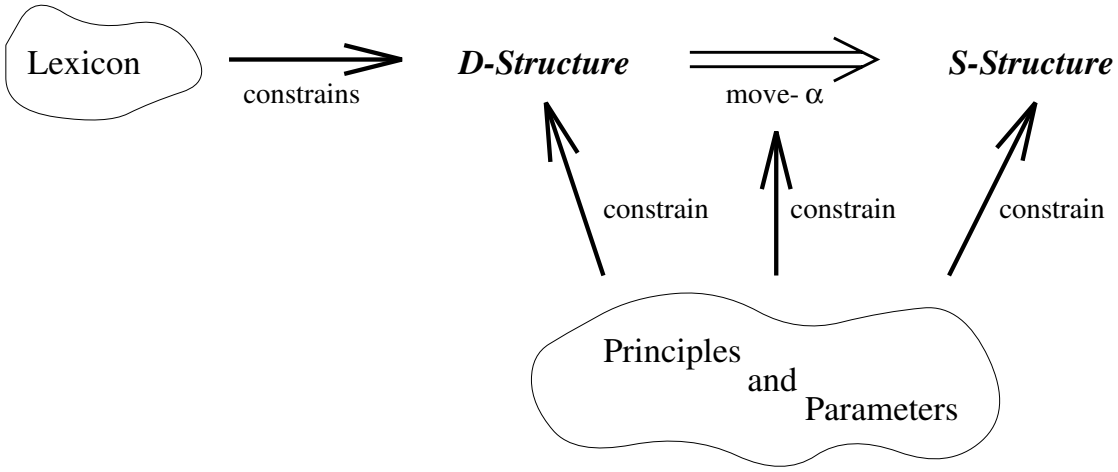


Figure 5.4: The structure of derivations in Government & Binding Theory

Tree Adjoining Grammar has been used as a metalanguage for linguistics because the formalism allows for the factoring of recursion and unbounded dependencies apart from the representation of local dependencies. TAG is a mathematical model of tree combination. Local dependencies are expressed locally on elementary trees; all unbounded dependencies arise during derivations as an effect of the mathematically defined derivation operations, namely adjunction and substitution. The significance of this approach was first explored by Kroch & Joshi (1985; 1987) and Kroch (1987). While much work has been done in this framework (which will be mentioned in this chapter as it becomes relevant), the most rigorous exposition of the theory of grammar that emerges if TAG is used as a metalanguage is given by Frank (1992). The theory is not actually non-transformational. As in GB, transformations operate on a D-structure and derive an S-structure, constrained by principles and parameters. However, these structures are constrained in size by the Condition on Elementary Tree Minimality (CETM) to a lexical element and its (lexical and functional) projections (Frank, 1992, p.54). Furthermore, nonterminals appearing on the frontier of an elementary structure must (be part of a chain which is) selected by the head, either through θ-assignment or predication, as specified by the TAG version of the Projection Principle (Frank, 1992, p.56). Thus, the result of the transformations are at most clause-sized.

[3]There have been proposals that the application of parameters should be limited to the lexicon (Borer, 1983). As we will see, the proposal made here is in the same spirit. However, in GB, it is impossible not to require that principles constrain further stages of the derivation.

Multiclausal structures are derived by using the formal[4] operations of adjunction and substitution, i.e., non-transformationally. The great advantage of this two-pronged approach is that the domain of applicability of move-α is greatly restricted, namely to an extended projection. A great simplification in the formulation of principles and parameters in the grammar ensues. This theory of grammar, which will be called the "basic TAG approach", is illustrated in Figure 5.5.

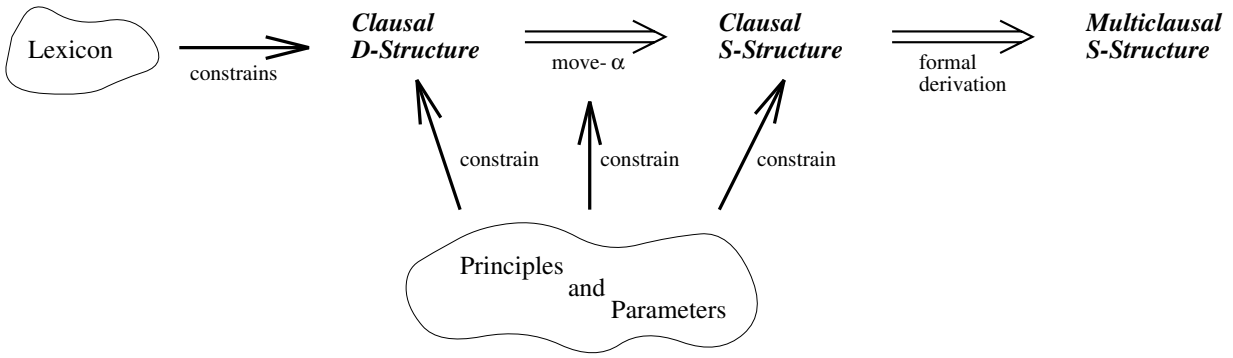


Figure 5.5: The "basic TAG approach": using simple TAG as a metalanguage for linguistics

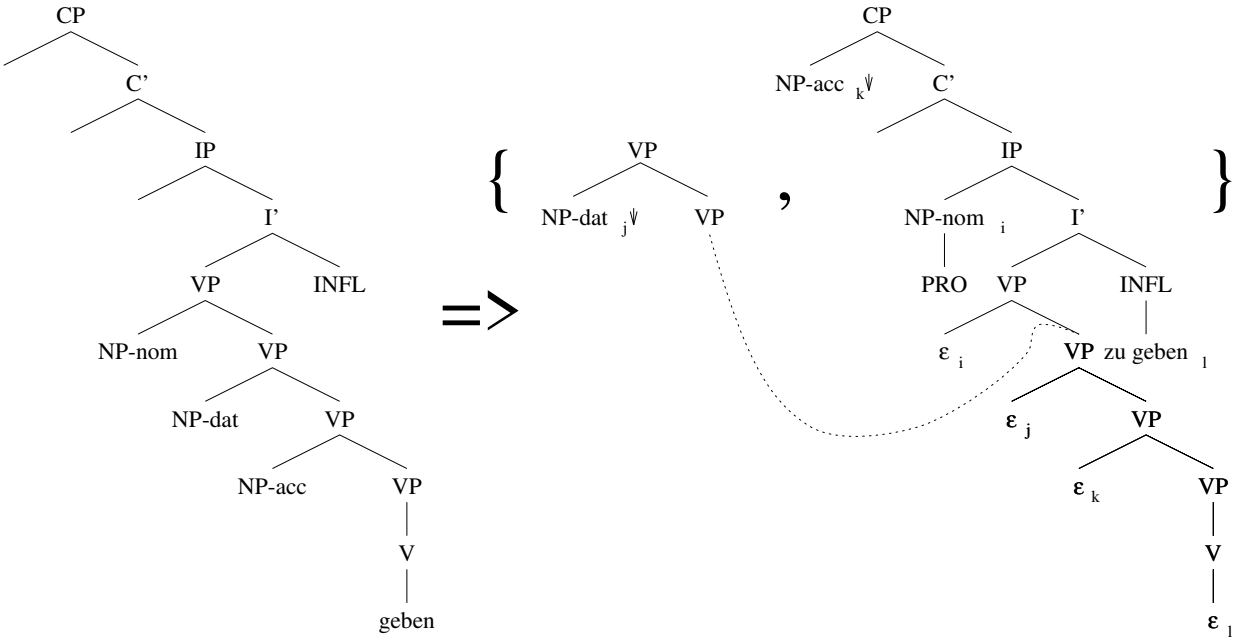


Figure 5.6: Derivation of *geben* initial tree set

The theory of grammar that will be sketched in this chapter will build on the basic TAG approach. In particular, it will retain the notions that the grammar consists of elementary structures that contain lexical heads and their projections, and that these elementary structures are combined using formal operations to derive multi-clausal structures. We could follow the basic TAG approach

[4]In this chapter, as in the entire thesis, the term "formal" will be used in the sense of "relating to a mathematically defined formal system". The term is also frequently used in the context of GB. There, the usage of the term is somewhat vague. We will use the term "formalized" to refer to this somewhat vaguer concept.

in defining clause-sized D-structure-like trees from which the actual elementary structures are derived. As we have seen in Chapter 3, the structures in the grammar will have to include sets of trees, so we could extend the formal operation allowed during the first derivation step to include "detaching" an argument from its projection (though of course it remains linked by a dominance link). This can be thought of as underspecified movement, whose destination has not yet been fixed. An example is shown in Figure 5.6. The tree at left shows a standard verbal tree with functional projections, assuming the VP-internal subject hypothesis. The tree on the right shows underspecified scrambling of the indirect object, in addition to several standard movements: the direct object *wh*-moves to the SPEC(CP) position; the subject raises to SPEC(IP), where it is realized as PRO; the verb raises to INFL. We could elaborate this approach by defining constraints on underspecified movement, and propose a grammar for German in a conservative extension of the basic TAG approach, with many direct references to GB analyses.

However, we will not choose that approach. In this thesis, we will explore an approach in which we can dispense with the transformational step altogether. The trees, elementary structures in a TAG-based theory of grammar, will themselves be derived formally and non-transformationally from even more elementary structures, which can be thought of as sets of very small trees. There are several reasons for attempting such an extension of the formal apparatus:

1. The hybrid nature of the basic TAG approach may be considered undesirable in itself for reasons of methodological parsimony.

2. Similarly, it could be considered unmotivated to move from a tree (clausal D-Structure) to sets of trees (clausal S-Structure) back to a single tree. The requirement that the initial structure is a simple tree seems unmotivated, given the formal insight that we will need sets in any case at some point during the derivation.

3. The lack of formal basis for the transformational stage of the basic TAG approach means that machine implementations of tree adjoining grammars must start with the set of derived trees, thus requiring a complete elaboration of all derivations for all items in the lexicon by hand.[5] This is what in fact is done in the LTAG project (Abeillé et al., 1990), a large-scale implemented tree adjoining grammar for English.

4. The elimination of the transformational component (move-α) from the theory increases its predictive power since the freedom in formulating principles and parameters is now restricted: in a nontransformational approach, both the movement itself and the resulting structure (clausal S-structure) are no longer possible loci of application of principles and parameters.

How will the clausal derivations be handled in a non-transformational way? The core proposal of this chapter is that the lexicon of a natural language can be represented as a set of trees of

[5]Metarules are another means of deriving many related trees from a single underlying representation. Metarules for TAGs have been defined by Becker (1990); a naturally restricted version of metarules, based on the two principles of the TAG framework, namely the factoring of recursion and the extended domain of locality, can be defined which does not increase the formal power of the system. Becker (1993) sketches how metarules could be used in the formulation of a principled theory of grammar. Vijay-Shanker and Schabes (1992) propose a similar method of deriving lexical entries, which they call "lexical rules". These approaches have the disadvantage that they introduce a new type of operation, while the proposal made here is based entirely on formal generative systems.

height 1, or, put differently, of context-free string-rewriting rules (henceforth, "rules" for short). These rules are linked by dominance links, and nonterminal symbols can be marked with the integrity constraint, as defined in Section 4.3, page 66. More formally speaking, the claim is that the lexicon is a grammar in the UVG-DL-Δ formalism. The derivation proceeds in two steps:

- In the *lexical* derivation, rules from a single set in the UVG-DL grammar are used in a partial derivation, which can be characterized by a set of trees. (The derivation need not form a single tree since some rules may not be used.)[6] This tree set may include dominance links and integrity constraints, but they are just those that are specified in the lexical rule set. These structures correspond to the clausal S-structures that make up the grammar in the basic TAG approach.
- In the *syntactic* derivation, the tree sets derived during the lexical derivation are taken to form a V-TAG grammar. The sets are combined using adjunction and substitution in the usual way to form larger structures.

Thus, while the lexicon and syntax of a natural language can be described by a V-TAG, this is an epiphenomenon of the way that the derivations proceed – the syntactic V-TAG grammar is really the result of derivations in the underlying lexical UVG-DL grammar.

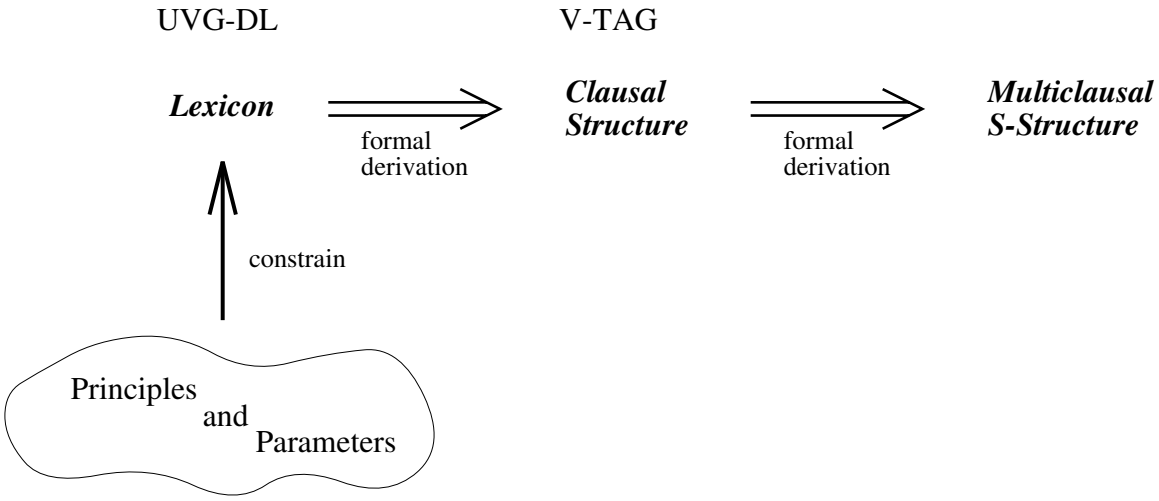


Figure 5.7: Using V-TAG as a metalanguage for linguistics: V-grammar

The question may arise why an intermediate step, the constitution of the V-TAG grammar from the derivation trees of the UVG-DL grammar, is needed at all. Direct formal motivation comes from languages that display cross-serial dependencies, such as Dutch and Swiss-German. If Conjecture 7, page 78 (that the copy language cannot be derived by a UVG-DL) is true, then UVG-DL is not powerful enough to derive the relevant syntactic structures. Indirect formal motivation

[6] Alternatively, instead of viewing these sets as specifications of partial derivations, we can also think of the lexical sets as sets of trees of height 1, and the lexical derivation as using the operation of tree substitution (but not adjunction).

comes from the fact that German allows both scrambling and long-distance topicalization, as discussed in Section 2.3, page 23. The appealing linguistic analysis of topicalization that we would like to exploit relies on tree adjunction. Finally, more general linguistic motivation for a two-step derivation can be seen in the fact that certain syntactic phenomena (e.g., passive transformation) have frequently been analyzed as lexical processes that apply to single lexical items, as opposed to syntactic processes which relate different lexical items.

In this approach, the only locus at which principles and parameters come to bear is the lexicon. All derivations proceed from the lexicon in a manner defined and constrained by the underlying formal systems. The approach, which we will call *V-grammar* in reference to the formal system in which it is implemented, is illustrated in Figure 5.7.

V-grammar bears certain similarities to the "minimalist" program recently proposed by Chomsky (1992). Broadly speaking, both approaches attempt to eliminate levels of representation in order to eliminate degrees of freedom in the theory. Specifically, both approaches eliminate D-structure in favor of an incremental composition of phrase structure from chunks of phrase structure projected from lexical items. The consequences of this representational parsimony are similar in both approaches. Chomsky (1992, p.5):

> Beyond PF options and lexical arbitrariness (which I henceforth ignore), variation is limited to nonsubstantive parts of the lexicon and general properties of lexical items. If so, there is only one computational system and one lexicon, apart from this limited kind of variety.

Chomsky's argument is driven by considerations of learnability: beyond the arbitrariness of the relation between the linguistic (lexical) sign and the learner's observed (or internal) reality ("Saussurean arbitrariness"),[7] which is irreducible, the linguistic variation that needs to be learned should be localized in one component of the theory. This component should be the lexicon, since evidence of lexical variation is most directly accessible to the learner. Furthermore, lexical variation of this kind should be limited. V-grammar can be seen as an attempt to translate this argument as narrowly as possible. In V-grammar, the only locus of application of principles and parameters is the lexicon itself. The derivational machinery – what Chomsky calls the "computational system" – is supplied in this approach by the mathematical formulation of the underlying representation formalisms, and is not subject to parametrization, or the application of principles of universal grammar. Principles and parameter types determine the types of variation (beyond Saussurean arbitrariness) that are allowed in the lexicon.

While the lexicon-based nature of the approach that is proposed here may appear to be motivated by Chomsky's argument about learnability, it crucially differs from Minimalism: there, the locus of application of principles and parameters is not principally the lexicon, which, as in GB, is not really defined *qua* level of representation, but the interface levels of LF and PF (though parametrization may in addition affect the lexicon). The model is illustrated in Figure 5.8. If we augment the model of V-grammar by an LF-like level (say, by using "synchronized" derivations

[7] We sidestep the issue of whether the arbitrariness is, as Saussure claimed and Chomsky assumes (p.4), within the sign, or a property of the relation of the entire sign (whose components are related by necessity) to reality (Benveniste, 1966).

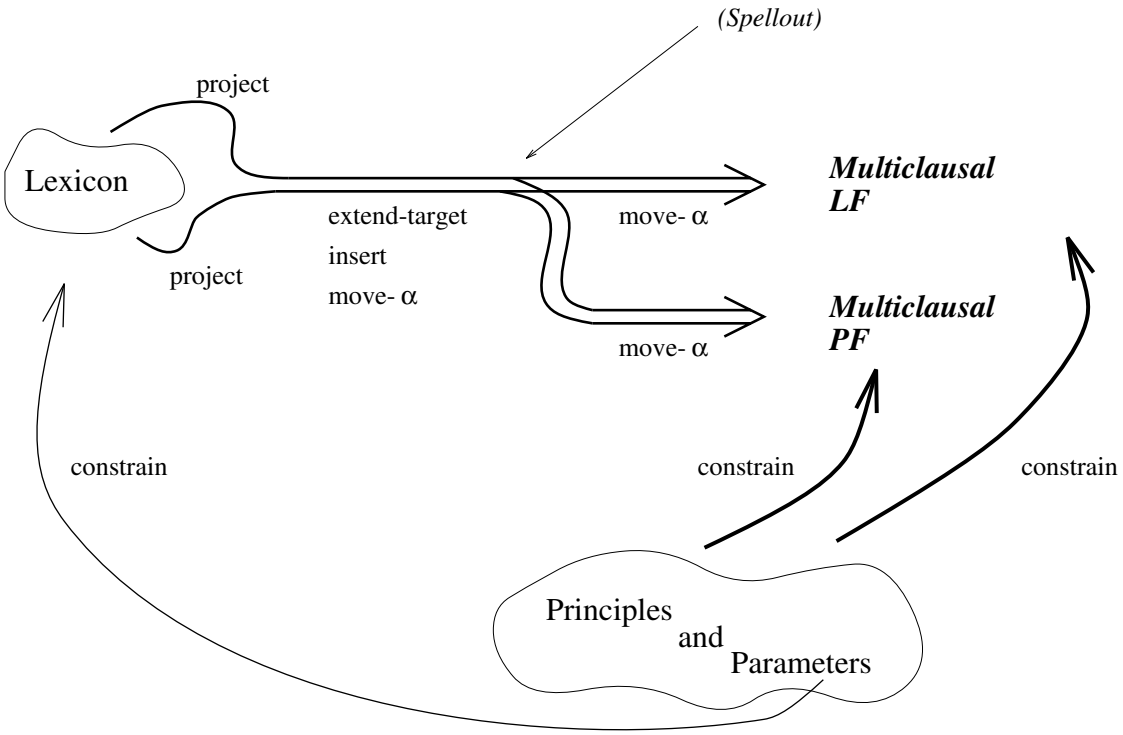


Figure 5.8: The structure of derivations in Minimalism

in the spirit of (Schabes and Shieber, 1990) – see Figure 5.9), the applicability of principles and parameters is still restricted to the lexicon itself. The question whether principles and parameters apply at the interface levels (as opposed to the lexicon) is necessary is an empirical one.

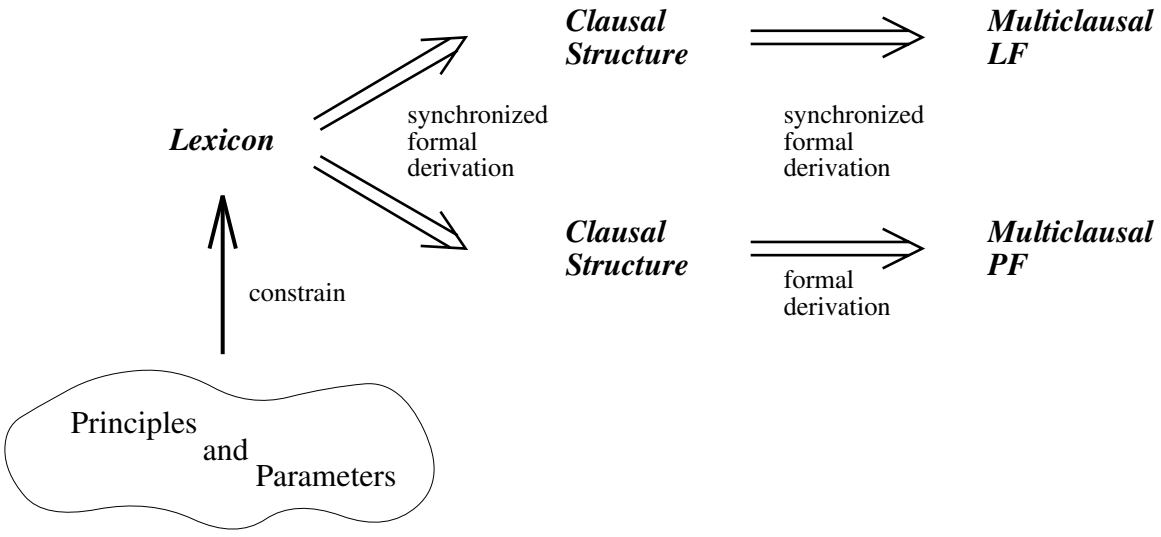


Figure 5.9: V-grammar with LF

5.2.2 Principles, Parameters, and Stipulations

There is no purely rational set of criteria that define "valid" principles and parameters. For example, statements that are less visibly related to the facts that they explain need not be less stipulative. While a good ratio of stipulations to explained facts may validate the stipulations, there may be yet other stipulations from which they can be derived. And there is no guarantee that a large number of facts can be derived from every "actual" principle or parameter. What is stipulative is therefore, at a given point in time of the research effort, a matter of judgment. Let us explore this issue just a bit more, and try to justify the claim just advanced.

Much GB work in syntax contains a technical part and an "intuitive" part. The technical part usually develops the machinery required to cover the data set outlined by the author; this part is formalized, in the sense that the authors establishes chains of causality using terminology that is given a precise definition in the framework ("government", "Chomsky-adjunction", "chain", and so on). In a separate section, let us call it the "deep explanations" section, the author often tries to motivate the machinery that has been introduced in the formalized section by relating it discursively to other observable phenomena in the language and to linguists' intuitions. For example, to motivate the previously suggested lack of verb raising in English an author may argue that English INFL is "too weak" to attract V, or English Agr is "opaque" to θ-role assignment (Pollock, 1989). This terminology is not given a precise meaning in terms of other GB terminology (or in terms of the formal apparatus, if a true mathematical formalism is used), and therefore has no explanatory value at all within that apparatus. Rambow (1994) provides a reinterpretation of Pollock (1989)'s analysis of English and French word-order and of Larson (1988)'s account of the double-object construction in English in V-grammar. Not surprisingly, while the analyses look rather different, the "deep explanations" provided by the original authors can by and large be carried over to the V-grammar analysis. Thus, these "deep explanations" can only make stipulations seem more *plausible*, not justify them as principles or parameters from *within* the actual theory.

If we extend the notion of Saussurean arbitrariness beyond the lexicon to the grammatical system itself, the issue becomes even more murky. Clearly, the difference in headedness of V in English and Japanese must be reduced to a parameter, whether it express the directionality directly, or whether the directionality is derived indirectly. (Of course, historical accounts can relate languages diachronically, but language change is certainly subject to arbitrariness as well.) But if arbitrariness is a characteristic of the grammatical system, then it is impossible to distinguish stipulations from principles and parameters purely on the basis of the way they are formulated. The resistance of certain stipulations over time to attempts to derive them from other stipulations may then be taken as evidence that they are indeed principles or parameters. In this sense the status of principlehood (or parameterhood) is meta- (or a-) theoretical.

In the approach we are proposing here, we can at least limit possible principles and parameters, since principles and parameters must be statable in the mathematical formalism. For example, possible principles and parameters include:

- The subject must be in SPEC(IP).
- Feature names, feature values, and node labels are parametrized.

- The set of closed rules (barriers) is parametrized.

However, it is not possible to state that barrierhood of a node can be voided by adjunction, that inflectional material can be lowered, or that the number of bounding nodes that can be passed is parametrized. These constraints from the formalism limit the range of possible theories of language.

Ultimately, of course, the key issue is learnability, addressed briefly above in Section 5.2.1. Learnability, after all, was Chomsky's original motivation for a principles-and-parameters methodology (Chomsky, 1986b). If a mathematical formalism adequately models the (biological) human language faculty, then stipulations that cannot even be formulated surely cannot be cognitively plausible principles and parameters. One could also put forward the stronger claim that principles and parameters expressible in the mathematical model are also biologically plausible, since the mathematical model exactly predicts possible biological states. Of course, whether a specific proposed parameter is actually learnable from the data a child is exposed to is a different matter and must be argued separately. We will, however, not address issues of learnability in this thesis.

5.3 A Theory of Grammar for V-TAG

In Section 5.2, we sketched how the formal systems developed in Chapter 4, principally UVG-DL and V-TAG, can be used to express a theory of grammar. In this section, we flesh out that proposal and develop a principled approach to the representation of syntax. We start out in Section 5.3.1 by introducing and justifying one of the central aspects of this approach, namely the wide use of features. We then discuss the role that lexical heads play in this approach[8] in Section 5.3.2. Section 5.3.3 discusses categorial projection and the notion of agreement. We discuss locality restrictions and the question of the size and shape of elementary structures in Section 5.3.4. The derivation and definition of syntactic tree sets is presented in Section 5.3.5. We end on a brief note about the interface between syntax and morphology in Section 5.3.6.

5.3.1 The Use of Features

The linguistic theory proposed in this chapter uses features for the representation of what in GB is represented by functional categories and the notions expressed by the X′ schema. We observe that in and of itself, the issue of whether the information contained in a node label such as 'C′' should be expressed by a node label or by features (be it without any hosting node label, as in HPSG, or be it on a host label such as 'VP'), has no empirical content whatsoever. However, since our notation departs from the accepted standard, we will attempt to justify this departure, and we will do so in three ways. First, researchers in frameworks with fixed node labels and projection schemas, such as GB, have on occasion seen the need to question this fixedness. Second, the choice of node labels can limit the type of analyses available in a TAG framework because of the technical definition of tree adjunction. Third, the desire to elaborate a non-transformational account of phrasal projection, including the phenomenon of head mobility, forces us to express categories

[8] This approach is also called the "adjoined-head approach" in (Rambow, 1994).

as features. We will examine these three points in turn; we turn to the issue of expressing the X′-schema in features in Section 5.3.3.

Within GB, it has been argued that neither a system of functional categories nor the X′ schema have turned out to be linguistic universals with great predictive power: Iatridou (1990) presents cross-linguistic evidence against (Pollock, 1989) and against the universality of specific functional projections, and Fukui and Speas (1986) and Grimshaw (1990) argue for language-internal variations in the X′ schema. Furthermore, the need for the "merging" of categories (such that a single category has properties of, for example, both the I and the C projection) has been identified in the GB literature; see (Haider, 1988; Bayer and Kornfilt, 1989; Rizzi, 1990), and (Heycock and Kroch, 1992), who give compelling evidence from conjunctive constructions in German, to which we return in Section 5.4.11. It seems impossible to give such "merged" categories any formal meaning other than by expressing the categories as features.

Within a TAG-based theory, the problem is as follows. TAG requires that root and foot node of auxiliary trees have the same node label. This formal requirement may influence the linguistic analysis. Consider, for example, an analysis of *wh*-movement proposed by Frank (1992). He discusses the difference between so-called "bridge verbs" and "non-bridge verbs" in English, exemplified by the following pair. Bridge verbs allow for the extraction of arguments from embedded clauses, while a weak island effect exists in the case of non-bridge verbs:

(1) a. What do you think that you saw?

b. ?? What do you regret that you saw?

Frank proposes to account for the difference by saying that bridge verbs have the option of either being C′-auxiliary trees (i.e., they are rooted in and subcategorize for C′), or being CP-auxiliary trees (i.e., they are rooted in and subcategorize for CP). Non-bridge verbs, on the other hand, are obligatorily rooted in CP. Assuming the usual phrase structure (without CP or C′ recursion), the facts follow straightforwardly. However, the variation in categorial subcategorization appears to be motivated primarily by the requirements of the formalism (namely that root node and foot node be of the same category).

Furthermore, Frank does not address the issue of *wh*-movement out of infinitivals. This kind of extraction is unproblematically licit in Standard German, and therefore of particular interest here. The problem is that non-bridge verbs now also allow extraction:

(2) What$_i$ do you regret seeing t$_i$?

Thus not only can bridge verbs be rooted in C′, but all verbs may be if they are adjoined into infinitival clauses. But the restriction on the type of embedded clause will, supposedly, be handled by features, so that the categorial information on the root node is not sufficient and a combination of categorial information and features is needed to provide the correct analyses. So the difference between bridge verbs and non-bridge verbs is really expressed on a restriction on what kinds of features can occur on what kinds of foot nodes. Methodologically, it would be nice if we could use only node labels or only features for regulating *wh*-extraction.[9] In a feature-based account,

[9]Of course, one could argue that infinitival clauses are rooted in IP (or VP). Therefore the subcategorization

the node labels are less informative, and thus the requirement on label equality is less restrictive.

Finally, as Frank (1992, Section 3.6) and Hegarty (1993a) note, it is impossible to adjoin functional heads if the projection they head is labeled differently from the lexical head. This is because the adjunction of the functional head, presumably motivated by some feature clash, must occur on a node in the projection of the lexical head. But then the definition of adjunction forces the root node of the adjoined tree to have the label of the lexical projection, and it cannot have its own label. The relevant information must be expressed in the form of features. But since we wish, for independent reasons, to incrementally derive the phrasal projection, we are forced to abandon separate labels for separate functional projections.

We have argued so far for an increased use of features in the development of a theory of grammar in our formalisms. How do we use the features? We need to develop a principled approach to assigning features. While co-specification of feature structures of any two nodes in a set of rules or trees is in principle possible, we would like to limit the features to be local to each individual tree or rewrite rule in a set. This will reduce degrees of freedom in the formulation of grammatical theory. Furthermore, we wish to relate the way features are used to the descriptive content of a rewrite rule. We suggest the following meta-principle:

(3) **Meta-Principle of Feature Structure Assignment in a Lexicon-**UVG-DL:

Assignment of (potentially) different values to a feature of the left-hand nonterminal and any right-hand nonterminal must be licensed by the presence or absence in the rule of the descriptive content of the features in question.

This meta-principle means that a feature can only change value below and above a rewrite rule if that rewrite rule contains an element that licenses such a change in value. For example, in a rule such as VP $\longrightarrow$ *has* VP, the left-hand side VP could have a value [+] for the feature [fin] while the right-hand side VP has the value [-], since the lexical item *has* is in fact finite. However, if we have a rule VP $\longrightarrow$ *to have* VP, then such a feature assignment would not be allowed. Equipped with this meta-principle, we now turn our attention to the central players in our theory of grammar, the heads.

5.3.2 Heads

We have argued in Section 5.3.1 that we should not distinguish between a lexical category and its node label on the one hand and a functional projection of that category and its node label on the other hand. Instead, we should express the information contributed by the the functional head by features. Therefore, we will start out by assuming that we have a set of features that corresponds to what is traditionally expressed categorially, features such as [±V], [±T], [±C]. Of these, the lexical category [±V] is privileged, since it will determine the node label.[10] We will assume that this set is parametrized across languages, though perhaps a core set of features

information can also be expressed categorially, like the bridge/non-bridge distinction. However, then we must accept that finite clauses that subcategorize for infinitivals (IP or VP) can be rooted in IP or VP – again, the formalism is forcing us to take an unintuitive position.

[10] It may also be possible to dispense with node labels altogether – we retain them for the sake of readability.

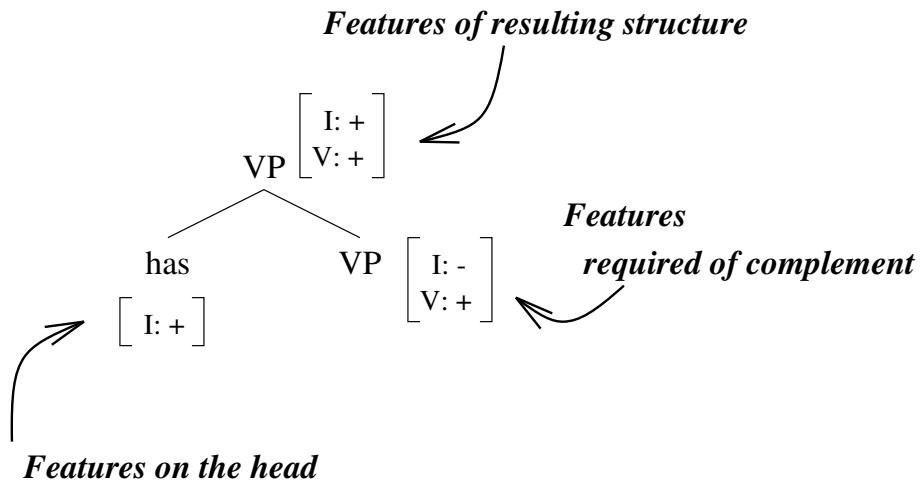


Figure 5.10: Sample head: English auxiliary

is universal – we leave this question aside, since we will concentrate on one language in this thesis. This issue of course equally affects all theories using such notions. This gives us our first parameter, the **Categorial Feature** parameter. We will furthermore assume that these parameters form a partial order, which corresponds to traditional assumptions about C-selection (categorial selection). We will therefore call this parameter the **C-selection** parameter. We will use "C-select" (as a verb) to refer to immediate domination in the C-selection hierarchy, while "C-dominate" will refer to the reflexive-transitive closure of C-select.

We will take the contribution of categorial features to be the definitional characteristic of a head, not its position in a structural configuration. A sample head is shown in Figure 5.10, an English auxiliary head. Each head introduces some new categorial features,[11] and places requirements on the categorial features which must be present in its complement. The resulting structure then has positive values on the union of the features of the complement and of the head. We see how the categorial features already present in the complement are passed up by the new head and are now also present in the new resulting structure. By convention, we will indicate only the positive features on the head itself (the "features on the head"). All other features are assumed not to be contributed by the head. The head will be referred to by its positive features, so that the head in Figure 5.10 is a "[+I] head".

Let us be more precise about feature requirements for heads.

(4) **Head Feature Principle**:

1. All and only the features C-dominated by the features on the head, except for features that are themselves on the head, must be instantiated positively in the complement.
2. A feature of the resulting headed structure is positive if and only if it is a feature on

[11] When we say that categorial features are "present" or "introduced", we will mean that they have the value +, or are given the value +.

the head, or instantiated positively for the complement.

The first clause implies specifically that the features on the head must be instantiated negatively in the complement of the head. As a consequence, we see that a categorial feature can only be introduced once by a single head in a given projection. Once introduced, categorial features percolate up the spine. (Exceptional cases such as CP-recursion will be handled by special means.) Note that the "features on the head" are not actually feature structures in the formalism and do not unify with anything – they are just a notational device that lets us name a head and that determines how the features of the top VP are related to the features of the bottom VP.

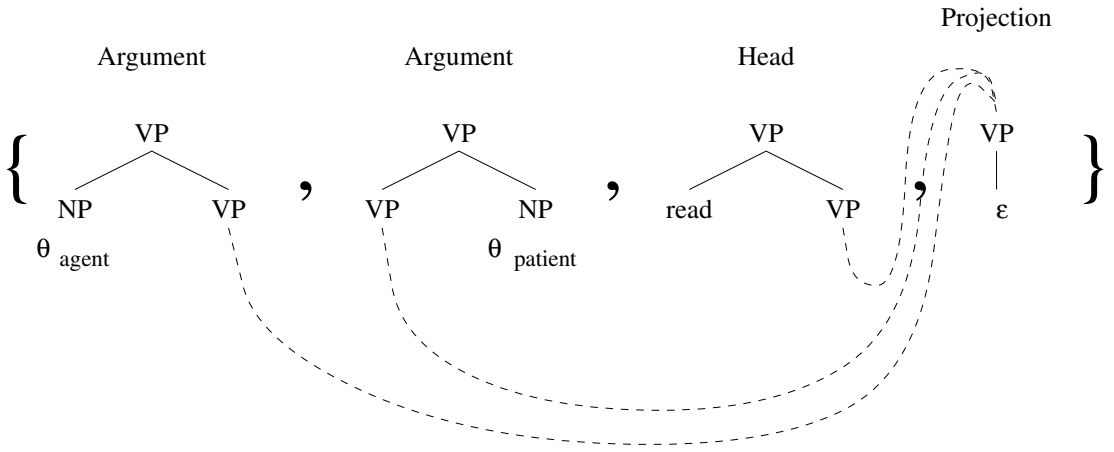


Figure 5.11: Sample lexical entry: English verb

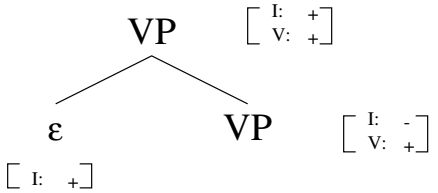
There are three types of heads:

1. Lexical heads are heads that contain a lexical stem, for example the head anchored in *read* in Figure 5.11 above.

2. (Overt) auxiliary heads, for example Figure 5.10. These correspond to overt functional heads in the GB framework. They are overt heads that may combine with lexical verbs but have no subcategorization frame of their own.

3. Empty heads, for example:

   (5)



   The head does not contribute any lexical material. The empty terminal symbol is actually not necessary, either formally or linguistically. In fact, from a linguistic point of view

the relevant feature information could simply be introduced by adjoining a "degenerate" auxiliary tree consisting of a single VP node and the relevant feature content. However, this is not compatible with the definition of features provided in feature structure TAG (FTAG) and that is being assumed here. Therefore, the empty head must consist of at least two distinct VP nodes; one may as well add the node labeled with the empty terminal string for clarity.

Let us address empty heads in slightly more detail. At first, the use of empty heads may seem puzzling, and merely a reinterpretation of the trace left by head movement. However, the proposed system is not merely a "static" reinterpretation of head movement since empty heads (as opposed to the traces left by head movement) have been proposed elsewhere in the literature for other phenomena, and are therefore needed on independent grounds. We will briefly review some cases in which empty heads have been proposed:

- A lexically null C^0 specified [+wh] may license a *wh*-phrase in SPEC(CP):

  (6) I wonder [who $e_{[+\mathrm{wh},+\mathrm{C}]}$ John saw]

  The empty C^0 is lexically licensed by *wonder*, and it in turns licenses the]wh-phrase. In general, empty C^0 heads are not freely available, and cannot license a *wh*-phrase in SPEC(CP):

  (7) *Who $e_{[+\mathrm{wh},+\mathrm{C}]}$ John saw

- In the English present subjunctive, an empty I^0 is licensed by specific lexical items such as *lest* and *require (that)*. This empty head can be interpreted as an empty modal auxiliary, roughly equivalent to *should*. It is this empty head that assigns nominative case to the subject.

  (8) a. ...lest the student $e_{[+\mathrm{T\text{-}fin},+\mathrm{AgrS}]}$ not $\mathrm{eat}_{[+\mathrm{V}]}$ his beans

  b. We require that the student $e_{[+\mathrm{T\text{-}fin},+\mathrm{AgrS}]}$ not $\mathrm{eat}_{[+\mathrm{V}]}$ his beans

We therefore conclude that the use of empty heads does not constitute an undue complication of the explanatory machinery. We will always require that empty heads must be specifically licensed by another head.

The principal source of explanatory power and of cross-linguistic variation is the set of heads that are licensed in a particular language. The **Head Feature Parameter** is a language-specific list of available heads, along with directionality information. Furthermore, we have the **Sentential Head Feature Parameter**, which specifies which categorial features must be present in root nodes of root clauses. Note that none of the parameters that we have introduced so far are specific to the representation we have chosen: any analysis in the traditional system of lexical and functional projections must also specify similar parameters. A derivation in this system consists in choosing lexical representations, assigning features in accordance with the Head Feature Parameter, and adjoining until the feature requirements determined by the Sentential Head Feature Parameter are met. The exact nature of derivations will be explored further in the following sections.

5.3.3 The X′ Schema, Predication, and Specifier-Head Agreement

The X′ schema (Figure 5.12) has become widely accepted as the representation of the phrase structure projected from both lexical and functional heads. It is purely structural in the sense that the terms "specifier" and "complement" designate *positions*, rather than a particular relation between the head and constituents in these positions. As we have argued in Section 5.3.1, we will represent information about the structure projected from heads in terms of features. However, this decision still leaves open exactly what will be represented. Generally speaking, there are two options. The first option is to maintain the spirit of the X′ schema and simply represent positions. This could be done by introducing a feature "Bar" that takes values 0, 1, or 2 (as is in fact often done). The second option is to directly represent the type of relation that holds between the head and other constituents. In order to decide between these two options, we will examine the range of relations that may hold between a head and the constituents in the specifier and complement positions, and investigate what elements of the structural configuration are crucial.

1. The specifier and complement may be related by subcategorization (or θ–role assignment) requirements to the head.
2. The complement may be the argument of a semantic operator such as tense or mood.
3. The specifier may agree with the head with respect to certain features.
4. The head and complement may form an open predication structure, whose subject is the specifier.

We will discuss these fours relations in turn.

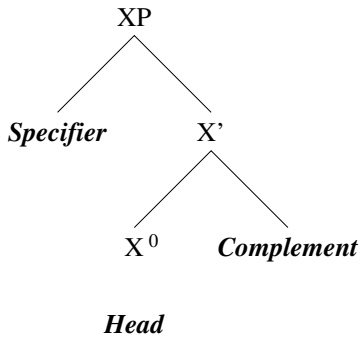


Figure 5.12: The X′ schema

First, let us consider subcategorization. The X′ schema relates the head to two positions, the complement and the specifier position. While transitive verbs fit nicely into this schema, intransitive verbs must be kept from acquiring a complement, and unaccusative verbs from acquiring a specifier. Ditransitive verbs must be explained by invoking additional machinery. Thus, some independent notion of licensing arguments will be needed, and the relation between a head and its arguments cannot be derived from the structural configuration of the X′ schema. Furthermore,

in the approach pursued here, the relation between head and arguments is already fixed in the lexical set (see Section 5.3.4). There is no further need for licensing of arguments.

Second, consider semantic operators. Semantic operators such as tense, negation, and perhaps pragmatic operators such as modality are operators of one argument. The X′ schema, however, has two positions (which would correspond to argument positions of the operator-head). There is therefore no particular need to postulate an X′ schema for these semantic operators, and we will not do so.

Thirdly, we consider agreement. The notion of specifier-head agreement, as proposed by Chomsky (1992) and Watanabe (1993) is interesting because it limits the way in which features can be used to achieve agreement by postulating that the two agreeing constituents must be in a specific structural configuration, namely that of specifier and head. This is appealing since it restricts the degrees of freedom in the theory, and we will therefore adopt this notion. We note that the notion of agreement does not involve the complement, the complement just happens to be there.

Finally, we discuss predication. Predication, contrary to the semantic operations such as tense that we have just discussed, can indeed be seen as a function of two arguments: the head turns the complement into an open predication, whose subject is the constituent in the specifier position. Thus, it seems that predication either requires the X′ schema for its expression, or that we can replace the X′ schema by the notion of predication altogether, since it is the only type of relation which really needs the X′ schema as such. This the approach taken in (Rambow, 1992b), where the analysis of Heycock (1991) is adopted (and adapted). Heycock, following Williams (1980) and Rothstein (1983), proposes that there is a primitive syntactic relation of predication. Based on an analysis of the distribution of expletives in various languages, she concludes that syntactic predication cannot be derived from argument structure or Case-assignment. Predication is a relation that holds between a maximal projection – the predicate – and an adjoined position – the subject. The syntax of a clause must therefore satisfy concurrently thematic requirements and requirements arising from predication. This can be achieved by assuming the VP-internal subject hypothesis. At D-Structure, the projection of V contains all of the thematic arguments of the verb. At S-Structure, the maximal projections of heads form predication structures that require subjects. Since heads can move, S-structure can contain several nested subject-predicate structures (the nesting happening in the predicate). Her analysis is particularly appealing for German because of the V2 phenomenon. The sentence-initial position must be occupied by a single element, but this element need not be an argument of the verb. Therefore, it is appealing to propose a second type of licensing relation, which is orthogonal to argument structure, and which involves a single subject. In (Rambow, 1992b), the notion of predication is implemented in feature structures, and it is used to force both subject-verb agreement (by stipulating that the subject of the I-predication must be the grammatical subject that receives nominative Case), and to force the V2 effect, at the C level. However, in this thesis we will abandon the notion of predication as a primitive notion and rely only on the notion of Spec-head agreement. We do so because the notion of Spec-head agreement is independently needed in the system, and because this notion can then derive the facts that were previously derived using the notion of predication:

1. The relation between the I-head and the grammatical subject can of course straightforwardly be reinterpreted as the Spec-head relation between the subject and an AgrS head, which happens to coincide with the T-fin head. (Here, we follow the spirit of Pollock (1989).)

2. The relationship between the C-head and the element in the *Vorfeld* of the V2 construction can be derived from Spec-head agreement between the *Vorfeld* constituent and a Top head that coincides with the C-head.

Both of these points are independently motivated by technical issues in the implementation of German syntax in a TAG-like framework. The splitting of I into T-fin and AgrS is motivated by facts from the Third Construction: on the one hand it is clear that the extraposed clause must be adjoined above the finite verb (since the extraposed clause follows the finite verb in clause-final position); at the same time the availability of word orders such as that in (18), page 16, shows that the matrix subject must c-command the extraposed clause (since the embedded object can place itself between matrix subject and matrix verb). We therefore must conclude that T and AgrS do not coincide. The splitting of C into Top[12] and C (or AgrW, as we will call it, and C) is motivated by the way successive-cyclic movement is handled in TAGs. In cases of extraction from embedded clauses with overt complementizers (in both English and German), the extracted element is in the specifier position of the C-projection headed by the overt complementizer. As suggested by Frank (1992) for English, a feature mismatch must force adjunction of the matrix clause, since such structures are not licit on their own (neither in English nor in Standard German). But this feature can be interpreted as an agreement feature, which is lacking on the [+C] head (the complementizer). We will therefore for the purpose of this thesis adopt the notion of Spec-head agreement, and refrain from a separate implementation of the notion of predication. We leave a deeper investigation into the relation between these two notions to future work.

From our examination of the relations within the X′ schema, we conclude that the structural configuration in itself is of little interest. In keeping with the goal of eliminating both unnecessary notions and unnecessary structure, we will therefore reject an explicit representation of the X′ schema as a purely structural device and instead opt for an explicit representation of the content of the relations that may hold between a head and the constituents in the specifier and/or complement positions. Under this approach, we have three types of categorial features:

- **Semantic operator categorial features** contribute directly to the interpretation of the sentence, such as V (which establishes a proposition), the tense operator T-fin (or T-inf), and C, which we will take to have a pragmatic function related to modality and illocutionary force, as evidenced by the fact that in English, German, and French, questions and imperatives are typically analyzed as involving movement of the verb to C.[13] (The relation between C and modality or illocutionary force is not straightforward; we refer to Brandt et al. (1992) for a fuller discussion.)

- **Case agreement categorial features** license Case. These are the AgrX features. In this thesis, we will only be concerned with AgrS, but an analysis of Passive is proposed in

[12]The existence of a separate Top head has been suggested independently (Culicover, 1993; Grimshaw, 1993), specifically for German recently by Brandt et al. (1992) (who infelicitously call it INFL) and Müller and Sternefeld (1993). Such a head also can be used to explain a morpho-syntactic paradox in Yiddish (see Section 5.5.1) and embedded topicalization in English (since we could assume that the head could appear below C in English, but not German).

[13]Bhatt and Yoon (1991) propose that C be associated with the grammatical category mood. While this is possible, it conflicts with the mirror principle, since in German, the mood morpheme occurs within the agreement morpheme, whereas C can be above AgrS (when a non-subject is in the *Vorfeld*).

(Rambow, 1994) that uses AgrO and AgrIO.

- **Semantic interpretation categorial features** license the presence of an element with features that indicate how it contributes to semantic interpretation. These features are Top and AgrW (though see below for the Top feature), both indicating that the element bearing the feature is λ-abstracted with respect to the proposition. (This of course does not determine the pragmatic use made of a topicalized element, it just makes it available to the informational component (Vallduvi, 1992).)

We will assume that each categorial feature is associated with both its functions (agreement or semantic operator) and a morphological realization. For example, the [+V] head is associated with the lexical root, and [+T-fin] with tense morphology. We briefly return to the issue of the morphology/syntax interface in Section 5.3.6.

We will in the following assume a version of the analysis of topicalization proposed by Chomsky (1977) (see also (Koster, 1975)). Chomsky suggests that topicalization is the result of using an empty topicalization operator, which is marked [+wh]. This operator then requires further adjunction of an overt element. We leave open whether the topicalized element is adjoined to the maximal projection or incorporated into the operator in some other way. (In the former case, we will ensure that such adjunction is licit in cases where otherwise adjunction to maximal projections is not, e.g., to the CP in German V2 constructions. We omit the details.) Since the topicalization operator is itself marked [+wh], we will henceforth only speak of the AgrW categorial feature, and omit the Top categorial feature.

We see that our proposed types of categories are either motivated by semantic notions (operators or abstractions for semantic interpretation), or by overt syntactic facts (Case). Orthogonally to this distinction, we find a formal distinction: some categorial features are agreement-checking features and require an agreement partner (the Case features and the semantic interpretation features), while others are merely operators and only require a complement. For the agreement categorial features, we will require that the agreement is satisfied, i.e., that an agreement partner is found that has a matching agreement feature.

(9) **Agreement Satisfaction Principle:** All agreement relations must be satisfied.

We will enforce the Agreement Satisfaction Principle using a feature [sat: $\pm$]. An AgrX head will have [sat: -] at its top VP node, and an argument or adjunct looking for an AgrX head will have [sat: -] at its bottom VP node. This is shown in Figure 5.13 for AgrS. (Features not specified here will actually make sure that the element is in spec-head agreement with the appropriate head.) All other arguments and adjuncts have feature [sat: +] at both VP nodes, which should be taken to mean that there is no unsatisfied need for spec-head agreement.

5.3.4 Syntactic Locality and the Lexical Sets

Now that we have specified how we will express phrase structure, let us turn to the question of what phrase structure our grammars will express. Recall from Section 5.2.1 that we propose to

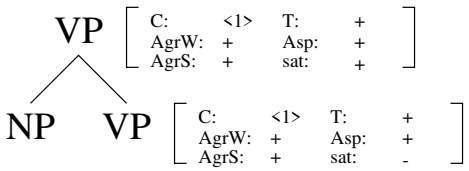


Figure 5.13: A nominal argument looking for a head with which to agree

formally represent the lexicon as a grammar in the UVG-DL formalism. During a first phase of the derivation, called the lexical derivation, we derive syntactic sets of trees from the lexical sets of small trees of the UVG-DL. These syntactic sets, which together form a grammar in the V-TAG formalism, are then combined to derive multi-lexemic structures. In this section, we will identify principles and parameters that apply to the lexical sets, and show how the lexical derivation works.

In the case of TAG-based linguistic theories (including the "basic TAG approach"), a problem arises that need not be addressed in this form by the principles suggested within the GB framework, namely the size of the elementary structures in the grammar. Frank (1992) proposes two principles, the Condition on Elementary Tree Minimality (CETM; p.54) and the Projection Principle (TAG version; p.56). Let us discuss them in turn.

The Condition on Elementary Tree Minimality requires that every elementary structure consist of the extended projection of a single lexical head in the sense of Grimshaw (1991). In our framework, the elementary structures are the lexical sets, to which categorial features are assigned. As we have seen, the Head Feature Principle, along with the C-selection parameter, restricts the features that can be added to a head to exactly those that make up an extended projection. In particular, functional categories not related to the lexical category are excluded. Thus, all we need to do is restrict these sets to a single lexical head.

(10) The **Condition on Elementary Tree Minimality** (V-TAG version, preliminary): every lexical set contains exactly one lexical head.

The Projection Principle (TAG version) requires that non-terminals which appear along the frontier of an elementary structure must be part of a chain whose tail is selected in that tree, either through θ-role assignment or predication. We see that Frank's Projection Principle conflates what is in standard GB the θ–Criterion (by requiring structure to be licensed) and the Projection Principle (by appealing to chains). (In fact, by allowing structure to be licensed by predication as well as by θ–role assignment, the θ–criterion is somewhat diluted, and it is not clear how Frank can enforce it without adding principles that apply to derived trees.) We will therefore encode the θ–criterion directly.

(11) The **θ–Criterion** (V-TAG version): frontier nonterminal nodes[14] in a lexical set are assigned θ–roles by its lexical head. Furthermore, all θ–roles of the head are discharged in this manner.

[14] In a UVG-DL vector, the frontier nodes are the nonterminals on the right-hand side of the productions in which no dominance link originates.

What is achieved in GB by the Projection Principle – requiring that the θ–criterion hold at all levels of representation – is now achieved by the formalism: UVG-DL and V-TAG are both defined in such a way that structure present in an elementary set must be used in a valid derivation, and furthermore, no nonterminal symbols may remain on the frontier when a valid derivation has terminated. We therefore need not state the Projection Principle separately. We can conveniently fold the θ–criterion into the CETM.

(12) The **Condition on Elementary Tree Minimality** (V-TAG version, final): every lexical set contains exactly one lexical head. There is a bijection between θ–roles the head assigns and the frontier nonterminal nodes of the set.

Observe that the CETM now corresponds exactly to what we have termed "co-occurrence constraints" in Section 3.2.2, page 37, which we used in our formal analysis of the data. Furthermore, this requirement implies the formal notion of lexicalization (see page 88). It is because of the CETM that the formal results obtained for UVG-DL-Δ_{Lex} and V-TAG-Δ_{Lex} apply to the linguistic grammars we will develop, including polynomial parsability.

We have stated what the lexical sets contain, and now need to address the problem of how the contents are arranged. We will assume that each lexical entry contains a projection from a head that bears only the feature [+X] to a node labeled XP, where X is the lexical category of the lexical head of that set (i.e., V, N, ...). As we have said before, the [+X] head is associated morphologically with the root of the lexeme. In German, the bare root never appears without additional morphology, whether finite or non-finite. For example, the root *reparier-* 'to repair' takes suffix *-en* for the infinitive, *-t* for third person singular present indicative, and so on; similarly, nouns require Case and never appear as bare roots. We therefore conclude that the [+X] head is in fact always an empty head, which we will call the *projection*.[15] The lexical head – the head containing the root and some additional morphemes – is therefore adjoined separately to this projection, as are other, non-lexical heads, such as auxiliaries and complementizers (to VPs) or determiners (to NPs). In addition to a projection tree and the lexical head, each set contains subcategorization structures, one for each θ–role. These are rules of the form XP $\longrightarrow$ XP YP, where X is the category of the lexical head in question, and Y is the category of the argument. (In Figure 5.14, as in the remainder of this thesis, these rules will be represented as trees of height one, with the left-hand side nonterminal as the label of the root node, and the right-hand side symbols labeling the daughter nodes.) The right-hand side XP is connected to the projection tree by a dominance link, in order to enforce a structural relation between an argument and the [+X] head which licenses it. An sample set can be seen in Figure 5.14. This concludes our discussion of the structures that make up lexical sets. The sum of these stipulations we will call the **lexical set structure principle**.

We now address the role of features in the representation for the lexicon on the formal basis of UVG-DL. These features will always have bounded values;[16] as discussed on page 86, the use of bounded feature structures does not change the formal properties of the lexical formalism. One

[15] Should there be evidence in a language that the root form can indeed appear without additional (null) morphemes, then of course the projection tree would be projected from this root and there would be no separate head. Nothing in our exposition actually hinges on this question.

[16] The reason why we can use such a limited version of feature structures is that the formalism, UVG-DL-Δ, provides descriptive tools that correspond to the unbounded feature structures in pure feature-based formalisms,

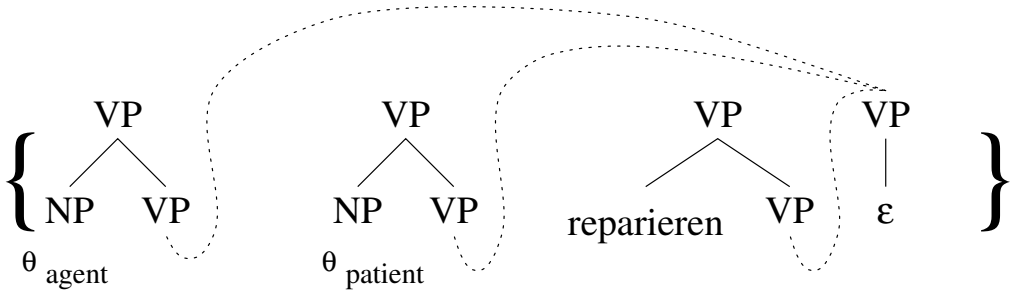


Figure 5.14: Lexical set for *reparieren* 'to repair'

feature structure is associated with every nonterminal symbol in a UVG-DL with features. This should be contrasted with the case of TAG and other TAG-derived tree rewriting systems: in the definition of feature-based TAGs given by Vijay-Shanker (1987), a *pair* of feature structures is associated with every node.[17] The reason for the difference is of course that we are using a string rewriting system to derive a tree rewriting system.

There is an important feature of our formal systems that we have not yet mentioned, namely integrity. Recall from Chapter 4 that the subtrees rooted in nodes marked with integrity constraints may not have any unfulfilled dominance links. Linguistically, this means that nothing can "move" out of such subtrees. We can therefore use the formally defined integrity constraint to implement a notion of "barrier", in the sense of Chomsky (1986a), loosely speaking. In fact, we have used the term "barrier" in our formal system to refer to a node label (nonterminal symbol) which, when it occurs in a rule, is always marked with the integrity constraint. We will use this notion to define the **Barrier Parameter**, which will define a set of node labels with associated feature structures which are barriers (in the formal sense). For example, if we say that CP, but not IP is a barrier (in the formal sense), then that captures some of the empirical content of Chomsky's stipulation that stipulation that IP cannot be a barrier (except by inheritance). It should be stressed that the similarity to the barriers of Chomsky (1986a) does not extend much beyond the superficial. In Chomsky's system, one barrier can be crossed but not two, and barrierhood can be voided by adjunction. In UVG-DL and V-TAG, barriers are absolute and rigid. It is possible that the barrier that we will propose for German – essentially CPs with satisfied spec-head agreement – will turn out to be universal, especially since it can be motivated from the needs of semantic interpretation. We leave this issue aside, and assume that the set of barriers is parametrized. (Note that the specification of the barriers for a given language will most likely involve not only the node labels, but also features.)

5.3.5 The Lexical Derivation and the Syntactic Sets

The lexical derivation starts out by assigning features to the head in accordance with the Head Feature parameter. We may think of this step as a separate morphological derivation; see Sec-

such as HPSG (Pollard and Sag, 1987; Pollard and Sag, 1994).

[17] Vijay-Shanker (1992) argues that this approach is essentially the same as the single-structured approach proposed by Harbusch (1990), since the double feature structures in FTAG are used to delay unification until all adjunctions have occurred, while in her account unification may have to be undone.

tion 5.3.6 for further discussion. Empty heads with categorial features dominated by the features on the head are added to the set (and thus specifically licensed by them). In this manner, despite the appearance of empty heads, the grammar is still lexicalized in the formal sense. The lexical derivation itself is actually a partial derivation in the UVG-DL formalism, using a single lexical set. The lexical derivation is represented as a (partial) derivation tree. The derivation is partial, since we will not require all rules from the lexical set (or vector, in terms of the formalism) to be used in the derivation. Unused rules will remain unattached to the projection in the resulting derivation tree, though any dominance link they have in the UVG-DL vector will be represented in the partial derivation tree.

An important constraint is imposed on the derivation: we assume that the semantics of each lexical item must be fully specified *before* it enters into the syntactic derivation (where the semantics of the lexical structures is composed to derive the semantics of the whole sentence). We will refer to this requirement as the **Lexical Interpretation Principle**. This principle has three consequences.

1. The head must adjoin to the projection during the lexical derivation, since otherwise the projection (the syntactic structure associated with the lexical item) cannot be interpreted. (This has the same effect as prohibiting movement of the head out of its extended projection.[18])
2. All semantic features, namely the semantic operator and the semantic interpretation categorial features, must be fully specified (as + or −) on the projection tree. The case agreement categorial features, however, need not be specified, nor need any features on trees that are not adjoined to the projection tree, since they do not contribute to the semantic interpretation.
3. From the requirement that all features be specified during the lexical stage we further deduce that non-lexical heads (auxiliaries, complementizers) must be adjoined during the lexical derivation. This is in keeping with the intuition of "extended projections" (Grimshaw, 1991).

While the Lexical Interpretation Principle may at first appear to apply to derivations, or to syntactic representations, in contradiction to our intention to devise a system in which all principles

[18]It has been proposed that in cases of incorporation (Baker, 1988), heads may move out of their maximal projections. A possible candidate for an incorporation analysis is the morphological causative found in many languages, such as Japanese (the example is from (Heycock, 1987, (1))):

(13) Mitiko-ga Taroo-ni ik-ase-ta
Mitiko-NOM Taroo-DAT go-CAUS-PAST
Mitiko caused Taroo to go

The addition of the causative morpheme *-ase-* adds a new actant (the causer) which receives nominative case, and the agent now receives dative case instead of the regular nominative. Under an incorporation analysis, the structure starts out as a biclausal structure, with the causative morpheme heading its own projection. The main verb, *ik-*, then moves out of its own projection and incorporates with the causative morpheme. In order to allow for the effects of incorporation, we will allow for a head to adjoin directly to the head of a different projection. For example, the TAG analysis that Heycock (1987) proposes for the Japanese causative can be straightforwardly adapted to the framework that is being developed here (in particular since it uses multi-component adjunction). Observe that in any case, we require that all heads must be adjoined during the lexical derivation, meaning that incorporation is a lexical process. We leave the implementation of incorporation to future work.

and parameters only apply in the lexicon (Section 5.2.1), this is in fact not the case. Since values for features are bounded, and are taken from finite sets, there really is no "unspecified" feature. A node with an "unspecified" feature is just a shorthand for a set of nodes, one for each possible value for that feature. Thus, this principle really only regulates how we can use our notation to represent groups of derivations, not the actual derivations themselves.

Now let us turn to the use of features in the syntactic formalism, V-TAG. As noted on page 103, we assume that V-TAG is equipped with adjoining constraints, and that these can be replaced without change in formal power by pairs of bounded feature structures, as in simple TAG (Vijay-Shanker, 1987). How do we assign features to the trees in the V-TAG? We have proposed that the V-TAG is the set of (partial) derivation trees of the lexical UVG-DL. We will assume that the feature structures of the syntactic V-TAG are just those assigned to the rewrite rules of the lexical UVG-DL. In this way, the resulting trees will have the usual double feature structures, but each feature structure will be contributed by a different string rewriting rule: given a node labeled A, the top feature structure is from the string rewriting rule that introduces A in its right-hand side, while the bottom feature node is from the left-hand side terminal of the rewriting rule that is used to rewrite that nonterminal. Since the UVG-DL derivation may only be partial, we do not require that feature structures unify in a rewrite step. In fact, we precisely do not unify the feature structures so that the resulting derivation tree can have two feature structures per node, as required by the definition of features in TAG-like systems such as V-TAG. If a mismatch occurs, then it has the effect of forcing adjunction at that node. In the end, the derived tree of a successful derivation in the syntactic V-TAG will of course be the derivation tree in the strict sense for the lexical UVG-DL, since whatever tree has been adjoined to resolve a feature conflict was also derived from the same UVG-DL. (Of course, this doesn't mean that the set of languages generated by the syntactic V-TAG is the same as that generated by the lexical UVG-DL from which it is derived – the intermediate constitution of the V-TAG increases the generative power.) For an example of how a UVG-DL derivation turns into a V-TAG tree, see Figures 5.18. 5.19, 5.20, and 5.21, starting on page 164.

What about root and frontier nodes? In the case of root nodes of initial trees, the top feature structure is supplied by the start symbol of the UVG-DL (which is also equipped with a feature structure, of course). In the case of substitution nodes, they only have one feature structure. (Terminal nodes of course do not have any feature structures.) We are left with root and foot nodes of auxiliary trees: these have only one feature structure, contrary to the way that FTAG is defined. We could define an FTAG-variant in which root- and footnodes of auxiliary trees have only one feature structure. It could be shown that the resulting system is weakly equivalent (and strongly equivalent up to the introduction of certain additional nodes) to FTAG. Alternatively, we could assume that the missing feature structure is simply unspecified for all values, and thus unifies with all feature structures it finds. Crucially, it turns out that these "missing" feature structures are not relevant for the linguistic model we will develop.

5.3.6 A Note on Morpho-Syntax

We would like to address briefly the issue of the morphology-syntax interface. One of the arguments for head movement in the syntax is the correspondence between the sequence of morphemes

in inflected forms and the C-selection hierarchy of corresponding functional projections, postulated on independent grounds. This has been called the "mirror principle" (Baker, 1988). This finds a straightforward explanation if we assume that heads move from head position to head position in the syntax, acquiring morphemes that are present at those positions. At first glance, it appears that such an explanation is not available in the approach that we have presented here, since the heads are adjoined fully inflected, and no actual movement takes place. We will argue that we can obtain the same results, by being a bit more precise about the first part of the lexical derivation, which we will call the "morphological derivation". We will assume that when a lexical set is chosen, it is not marked with any features. During the morphological derivation, we freely assign features to the head of that set, except that these features are added incrementally in such a way that the Head Feature Principle is never violated. If we assume that each feature either added to the complement or to the head corresponds to a morpheme added to the head, then we can easily see that we can predict the mirror principle. In fact, we can avoid some of the empirical problems discussed by Joseph and Smirniotopoulos (1993) for Modern Greek. In Modern Greek, a single morpheme can occasionally represent two or more morphological categories, but need not. This is puzzling under a movement analysis, but can be derived if we assume that features are added not one at a time, but in accordance with morphemes. So if a morpheme corresponds to two features, then they can be added at the same time. This approach avoids the problems of the notion of "feature-checking" of Chomsky (1992), which loses the explanation for the mirror principle.[19] For more details, we refer to (Rambow, 1994).

5.3.7 Summary: Principles and Parameters

In this section, we briefly summarize the principles and parameters that we have proposed. We characterize them in an informal manner. We start with principles.

- The **Head Feature Principle** (page 141) specifies that heads can only contribute features that are not yet present in the projection (thus, the same feature cannot be contributed twice), and forces all features that the features on the head C-dominate to be already present below (thus, there can be no features "missing" from a projection).
- The **Agreement Satisfaction Principle** (page 147) requires that all agreement relations be satisfied.
- The V-TAG-version of the **Condition on Elementary Tree Minimality** (page 149) specifies that every lexical set contain exactly one lexical head, and that there is a bijection between the θ–roles the head assigns and the frontier nonterminal nodes of the set.
- The **lexical set structure principle** (page 149) specifies that lexical entries consist of a projection tree (which dominates an empty category), and trees for the head and each argument of the head.
- The **Lexical Interpretation Principle** (page 151) requires that the semantics of each lexical item must be fully specified *before* it enters into the syntactic derivation. This

[19] Of course, it can be independently stipulated that the morphological features are organized hierarchically in a way to mirror the C-selection hierarchy in the syntax. The explanatory force of such an approach is diluted.

means that heads cannot move out of their extended projections (except perhaps in lexically controlled cases of incorporation), that all semantic features must be fully specified prior to the syntactic derivation, and that non-lexical heads (auxiliaries, complementizers, and empty heads) must be adjoined during the lexical derivation.

We now turn to parameters (some of which may in fact turn out to be principles).

- The **Categorial Feature** parameter (page 141) specifies the set of categorial features.
- The **C-selection** parameter (page 141) specifies the C-selection hierarchy formed by the categorial features.
- The **Head Feature Parameter** (page 143) is a list of available heads, along with directionality information.
- The **Sentential Head Feature Parameter** (page 143) specifies which categorial features must be present in root nodes of root clauses.
- The **Barrier Parameter** (page 150) specifies which nodes are marked with the integrity constraint and therefore act as (absolute) barriers for extraction of any sort.

5.4 A V-Grammar for German (Fragment)

In this section, we will present a fragment of a V-grammar for German. We start out by discussing the issue of clause union in Section 5.4.1. Section 5.4.2 presents parameters that constrain the lexicon, and gives some examples of lexical entries. Section 5.4.3 shows how monoclausal structures are derived from the lexical entries. We then turn to multiclausal structures, starting with embedded infinitival clauses. Non-finite embedded clauses are discussed in Section 5.4.4, extraposition in Section 5.4.5, long-distance scrambling scrambling and the "Third Construction" in Section 5.4.6, and clausal scrambling in Section 5.4.7. We then turn to embedded finite clauses in Section 5.4.8, and discuss long-distance topicalization in Section 5.4.9. We show how our analysis accounts for empirical differences between scrambling and topicalization in Section 5.4.10, and finish with a brief remark on coordination (Section 5.4.11).

5.4.1 Clause Union and Verb Complex Formation

An issue specific to Germanic syntax must be discussed at this point, since it bears on the proper definition of the lexicon. Evers (1975) proposes that sentences with recursively embedded clauses in Continental West Germanic languages (CWG) differ significantly in their syntactic analysis from their non-center-embedded counterparts in CWG and in other languages such as English and French. In CWG constructions in which verbs are adjacent to each other, he proposes a two-pronged process for matrix verbs that allow for "coherent" constructions:

1. Embedded verbs move up to their governing verbs and form a single morphological unit through incorporation ("verb cluster formation"). The θ–grids of the two verbs are merged.

2. The process of verb raising dissolves the clause boundary of the embedded clause ("clause pruning").

This analysis has become widely accepted, to the point that it is often assumed in the literature without mention, let alone discussion. As an instance in point, Müller and Sternefeld (1993) simply state that German has no long-distance scrambling, while acknowledging examples of the kind given in Section 2.2.1, page 13, which involve movement of an argument to the left of an argument of a different verb. Since they assume clause pruning has occurred, such examples are simply instances of local scrambling for them. No further discussion is given, and in fact Evers (1975) is not even cited. However, Evers's proposal has not gone uncontested. Kroch and Santorini (1991) argue against both verb cluster formation and clause pruning. We will briefly review their arguments here, since the issue bears on the linguistic analysis that will be presented in this chapter.

Kroch and Santorini (1991) (henceforth KS) present separate arguments against clause pruning and against verb cluster formation. We will review them in turn. Both arguments hinge on the observation that the proposed processes represent additions to the theory of grammar which are unmotivated from other languages, and that therefore analyses that avoid these mechanisms should be preferred.

First, we summarize KS's arguments against clause pruning.

1. KS argue that facts relating to the scope of negation in center-embedded structures are misrepresented by Evers (1975) and subsequent authors (notably Haegeman and van Riemsdijk (1986)), and that the actual facts can be derived by an analysis that does not require clause pruning but that relies on the independently motivated process of raising at LF.

2. Clitic climbing (the presence of clitics from an embedded clause in a matrix clause) occurs in both Dutch and German, but in Dutch it occurs only out of clauses with bare infinitives, and in German it also occurs out of extraposed clauses (which cannot have undergone verb cluster formation or clause pruning). Therefore, KS conclude that clitic climbing facts are not explained by the clause pruning hypothesis.[20]

3. The clause pruning hypothesis violates the projection principle.

Clause pruning is conceptually independent of verb cluster formation. For example, den Besten and Edmondson (1983) present an analysis based on verb cluster formation, but which does not include clause pruning. KS's arguments against verb cluster formation are as follows:

1. Arguments for verb cluster formation based on nominalization fail to take into account that in German, (putative) verb clusters that include *to*-infinitives cannot nominalize. For bare infinitives, the nominal forms of verb clusters can be derived by compounding the nominalizations of the individual verbs as well as by nominalizing a verb cluster.

[20] We note in passing that if clitic climbing (or inter-clausal scrambling) facts are taken as evidence for clause pruning, then the conclusion that German has no long-distance clitic climbing (or scrambling) is merely a vacuous restatement of a previously made stipulation.

2. The negation facts previously mentioned can be adduced as evidence for verb cluster formation even in the absence of clause pruning, since a matrix negator preceding an embedded verb is strange, but if the verb sequence is interpreted as a cluster, then there is no other place for the negator. However, KS provide evidence that a matrix negator may be separated from the verb cluster by an argument, whose incorporation into the verb cluster is implausible.

3. The fact that gapping can affect verb sequences does not show that they form a single morphological constituent, since gapping need not affect constituents (as evidenced by both English and German). Indeed, when considered in detail, the gapping facts cut against the verb cluster hypothesis.

4. Similarly, coordination facts are inconclusive, because coordination of non-constituents is possible.

5. For certain sequences of verbs in German (main-modal-auxiliary), the finite verb exceptionally precedes the others, as shown in (14).[21]

   (14) ...daß Karl hat singen können
   ...that Karl has sing_{INF} can_{INF}
   ...that Karl has been able to sing

   While this can be derived by stipulating permutations within the verb cluster, such stipulations are *ad hoc* and, more importantly, there are other orders among these three elements in German dialects which cannot be derived by any reasonable permutations.

KS therefore reject a "morphological" analysis of verb raising and present a "syntactic" analysis, in which neither clause pruning nor verb cluster formation occurs. Their analysis is based on simple TAG. It should be noted that if simple TAG is chosen as the meta-language for linguistic description, it is crucial that neither clause pruning nor verb complex formation occur. This is because once the elementary trees of the TAG have been derived (by whatever means) their shape cannot be altered during the formal derivation. In particular, lexical items from different trees cannot incorporate, and argument positions from one tree cannot become argument positions of a different tree. Thus, if clause pruning and/or verb cluster formation were to be implemented in a simple TAG framework, they would have to occur prior to the TAG derivation, i.e., as lexical processes. While this is possible (and perhaps even linguistically appealing), this would mean that the grammar would be infinite, and thus not be a TAG. The formal advantages of using TAG would be lost.

In the discussion we will present in this chapter, we will adopt KS's analysis and reject both the clause pruning and verb cluster hypotheses. However, as we will argue in Section 5.4.4, this choice is not dictated by the requirements of the formal system we are using, as it is in the case of simple TAG. In fact, the analysis we will present will be interpretable as implementing in some way both rejected hypotheses of Evers's, thus reflecting his underlying intuition and showing that the

[21] This word order coincides with the exceptional use of the infinitive of the modal instead of the expected participle (*Infinitivus pro Participio* or IPP-effect).

radical difference between his morphological analysis and a purely syntactic analysis (as presented by KS) is mainly an artifact of the underlying metalanguage (be it the GB framework or simple TAG). Furthermore, we will see echoes of both the clause pruning and the verb cluster formation hypotheses in the processing model we will derive from the purely syntactic analysis presented here (see Chapter 6).

5.4.2 The German Lexicon

In this section, we specify language-specific parameters for German, and give some examples of lexicon entries.

We start out with the list of categorial features for German. As mentioned in Section 5.3.2, this may in fact be a universal list not subject to parametrization.

(15) a. **Categorial features for German** (verbal):

- Semantic operator categorial features: C, T-fin, T-inf, Asp, V
- Case agreement categorial features: AgrS
- Semantic interpretation categorial features: AgrW

b. **C-selection hierarchy for German**:
C > (AgrW) > AgrS > T-fin > Asp > V,
T-inf > Asp > V

Some words of explanation are in order. First, observe that we have split T into two categorial features, depending on whether it is finite or not. It is not immediately clear whether, instead, T should not be considered a single categorial feature which is augmented with a secondary binary feature, [fin: ±]; we will assume that finite and non-finite T are simply two distinct categorial features. In the diagrams, in order to avoid clutter, we simply write T; [T: -] will mean [T-fin: -, T-inf: -]; [T: fin] will be shorthand for [T-fin: +, T-inf: -]; [T: inf] will be shorthand for [T-fin: -, T-inf: +]. Observe that [T-fin: +, T-inf: +] is impossible, since the two categorial features are mutually exclusive as defined by the C-selection hierarchy. Note furthermore that we cannot have any of the features C, AgrW, AgrS, or T-fin when we have the feature T-inf, since the former do not C-dominate the latter.[22]

Second, we use a feature Asp that we will assume is associated with infinitival morphology, such as for infinitives and past participles. Semantically, the feature will be related to an aspectual operator. In the case of finite (lexical) verbs, there is no overt aspectual morphology in German; for the sake of uniformity, we will assume that for finite lexical verbs, aspect and tense coincide morphologically (and perhaps semantically in a single operator), so that for such verbs the features [+Asp,+T-fin] always co-occur. We assume that the selection of aspectual forms by particular auxiliaries is handled through a feature-passing mechanism and omit the details. Finally, we do not address the issue of multiple auxiliaries, as in English *They will have been eating.* We could

[22] Infinitival questions such as *Was tun?* "What do we do?", lit. 'what to-do?', suggest that AgrW may also C-select for T-inf. We leave this possibility aside.

either say that *eating* introduces [+Asp], and *will* introduces [+T-fin], with the other auxiliaries introducing no categorial features at all. Or we could propose that there are in fact different Asp heads (for ±future, ±perfective, ±progressive) which are arranged hierarchically. We leave this issue for further research.

We now specify the inventory of German heads. The main issue is that of the verb-second phenomenon and its complementary distribution with overt complementizers,[23] as documented in Section 2.1. We will follow the standard account that has been developed in GB theory by Koster (1975), Thiersch (1978), den Besten (1983), and others. Under the transformational analysis, the verb first raises to head-final I and acquires tense features, and then raises further to head-initial C. An element (argument or adjunct) must obligatorily move into SPEC(CP) if C_0 is occupied by a verb. If C_0 if filled with a complementizer, a verb cannot raise to it, and the complementary distribution of the complementizer and the V2 effect follows. In this case, SPEC(CP) must (in Standard German) remain empty, though in Bavarian and certain regional varieties of Standard German, certain elements may appear. While this account contains a large number of stipulations, it has predictive power compared to the mere description of the facts. For example, if we assume that separable particles remain in their base-generated position, their distribution follows immediately, and since it is the verb, not the MF elements that change position, their behavior with respect to scope of negation and adverbs is correctly predicted to be unaffected by the position of the verb.

We will adopt the intuition behind the GB analysis of V2, namely that the verb in the V2 construction is in the same structural configuration as the complementizer. Of course, we will define this structural configuration in terms of categorial features. As explained in Section 5.3.3, page 144, we do not assume that the obligatory presence of a constituent in (what in traditional analyses is) SPEC(CP) is linked to the C_0 head; rather, we will assume that it is triggered by the obligatory presence of a [+AgrW] feature in German sentences, that an independent (empty) [+AgrW] head is not licensed below a complementizer [+C] head, and that there is a verbal [+AgrW, +C] head in the grammar. Translated to the parameters we have just defined in Section 5.3.2, we obtain the following.[24]

(16) **German Sentential Head Feature Parameter**:

German sentences must have features [+AgrW, +C].

Recall that these features can only be present if we also have [+V, +Asp, +T-fin, +AgrS] in the same projection.

Now let us specify the inventory of heads in German. In the following table, the column labeled "what" is just given for the reader's orientation and has no status in the theory. Recall also that the features of the complement of a head and the resulting features of the headed structure can be deduced from the features on the head, given the C-Selection hierarchy. We represent them here

[23] As von Stechow and Sternefeld (1988) point out, there is in fact no "complementary distribution". It would be correct to speak of the complementary distribution of the V2 syntax and the verb-final syntax (as opposed to their being in free variation), but this does not imply identity of the complementizer and the verb in second position. We retain the sloppy terminology.

[24] We ignore sentences of the form *Daß du auch immer zu spät kommen mußt!*, lit. "That you must always be late (is deplorable)".

(and in subsequent head tables) only for convenience. Strictly speaking, the actual parameter consists only of a list of pairs, representing the features on the head and the directionality of the head. For the sake of legibility, we only represent the semantic operator categorial features in the following table. The full representation (including the AgrS and AgrW features) is shown in Figure 5.15.

(17) German verbal heads (only semantic operator heads shown):

What	Features on Head	Features of Complement	Resulting Features of Structure	Dir of Head
participles and bare infinitives	+Asp	+V	+V, +Asp	final
zu-infinitives	+Asp, +T-inf	+V	+V, +Asp, +T-inf	final
tensed verbs	+Asp, +T-fin	+V	+V, +Asp, +T-fin	final
tensed auxiliaries	+T-fin	+V, +Asp	+V, +Asp, +T-fin	final
tensed verbs	+Asp, +T-fin, +C	+V	+V, +Asp, +T-fin, +C	initial
tensed auxiliaries	+T-fin, +C	+V, +Asp	+V, +Asp, +T-fin, +C	initial
tensed verbs and auxiliaries	+C	+V, +Asp, +T-fin	+V, +Asp, +T-fin, +C	initial
empty	+Asp, +T-fin	+V	+V, +Asp, +T-fin	—
empty	+T-fin	+V, +Asp	+V, +Asp, +T-fin	—

We then can add [+AgrS] freely to tensed heads, as long as the C-selection hierarchy is not violated, and we must add the [+AgrW] head to any head that also has [+C]. We can attempt to relate the obligatory coincidence of [+AgrW] and [+C] in verbal heads to requirements of the illocutionary operator associated with C; ultimately, this stipulation just corresponds to the observation that German shows the V2 effect, which to date no theory has succeeded in reducing to other facts of German in an interesting way. Note that auxiliaries and full verbs look alike in this table of heads. They are differentiated with respect to their ability to assign θ–roles to arguments, which is expressed in this framework by the fact that full verbs are grouped with subcategorization structures and a projection tree into lexical sets, while auxiliaries are not. There are also untensed auxiliaries, which do not contribute any features, and are therefore not heads by our definition.

Complementizers (such as *daß* 'that' and *ob* 'whether') are [+C] heads but not [+AgrW] heads. The head is shown in Figure 5.16

(18) German complementizer head:

What	Features on Head	Features of Complement	Resulting Features of Structure	Dir of Head
participles and bare infinitives	+Asp	+V	+V, +Asp	final
zu-infinitives	+Asp, +T-inf	+V	+V, +Asp, +T-inf	final
tensed verbs	+Asp, +T-fin	+V	+V, +Asp, +T-fin	final
tensed verbs	+Asp, +T-fin, +AgrS	+V	+V, +Asp, +T-fin, +AgrS	final
tensed auxiliaries	+T-fin	+V, +Asp	+V, +Asp, +T-fin	final
tensed auxiliaries	+T-fin, +AgrS	+V, +Asp	+V, +Asp, +T-fin, +AgrS	final
tensed verbs	+Asp, +T-fin, +AgrS +AgrW, +C	+V	+V, +Asp, +T-fin +AgrS, +AgrW, +C	init
tensed auxiliaries	+T-fin, +AgrS +AgrW, +C	+V, +Asp	+V, +Asp, +T-fin +AgrS, +AgrW, +C	init
tensed verbs and auxiliaries	+AgrS, +AgrW, +C	+V, +Asp, +T-fin	+V, +Asp, +T-fin +AgrS, +AgrW, +C	init
tensed verbs and auxiliaries	+AgrW,+C	+V, +Asp, +T-fin, +AgrS	+V, +Asp, +T-fin +AgrS, +AgrW, +C	init
empty	+Asp, +T-fin	+V	+V, +Asp, +T-fin	—
empty	+T-fin	+V, +Asp	+V, +Asp, +T-fin	—
empty	+AgrS	+V, +Asp, +T-fin	+V, +Asp, +T-fin, +AgrS	—

Figure 5.15: German Heads

What	Features on Head	Features of Complement	Resulting Features of Structure	Dir of Head
complementizer	+C	+V, +Asp, +T-fin +AgrS, –AgrW, –C –T-inf	+V, +Asp, +T-fin +AgrS, –AgrW, +C –T-inf	initial

We now consider adjuncts. We distinguish two cases: [+wh] adjuncts – adjuncts explicitly marked for their [+wh] status – and "regular" adjuncts. (Recall that if we assume that topicalization is brought about by adjunction to an empty [+wh] operator, then the adjuncts adjoined to it also become [+wh] adjuncts, even if they are not inherently so.) The [+wh] adjuncts must enter into spec-head agreement with a [+AgrW] head. As for the second type, we observe that adjunction of adverbs cannot occur to CP (in terms of the standard GB analysis), no matter what the type of the adverb, and no matter whether C_0 is occupied an overt complementizer or by the finite

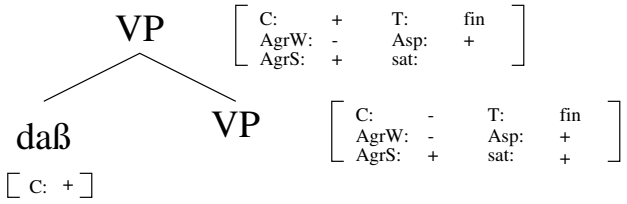


Figure 5.16: Elementary lexical tree for complementizer head (*daß*, 'that')

verb:

(19) a. * Ich glaube, morgen/vielleicht/schnell daß Hans kommen wird
I think, tomorrow/maybe/quickly that Hans come will

Intended meaning: I think that Hans will come tomorrow/maybe/quickly

b. * Hans morgen/vielleicht/schnell wird kommen
Hans tomorrow/maybe/quickly will come

Intended meaning: Hans will come tomorrow/maybe/quickly

Within the MF, however, adverbs can occur in any position, as shown in (8), page 12. In the framework under elaboration here, we obtain the following two types of adverbs for German:

(20) German adjuncts:

What	Example	Features Required Below	Dir of Head
[+wh] Adverbs	*gestern, vielleicht, wann*	+AgrW, –sat	left
[-wh] Adverbs	*gestern, vielleicht*	–C	left

Finally, we need to specify the barriers for German. Recall that the same barriers apply to both the lexical and the syntactic derivations.

(21) **German barriers:**

VP [+C, +sat]

Since a [+C] head does not license spec-head agreement, the two features in the barrier must be seen as two independent stipulations (i.e., we cannot say that a saturated C-head is a barrier, since such a notion is nonsensical). We may want to attempt to derive them from semantic notions – however, any attempt in the transformational literature to limit movement has never included such an attempt. In GB-like terminology, we can say that CP will act as a barrier to extraction. As we will see, this stipulation, together with the [+C] requirement on German root clauses, derives the V2 effect. Furthermore, it predicts the islandhood of relative clauses and other adjuncts, and of finite embedded clauses. Topics and *wh*-words can escape from the latter, but not from the former, by adjunction of the matrix clause, as we will see in Section 5.4.6.

Thus, Webelhuth (1989)'s observation that scrambling and *wh*-movement are subject to many of the same island effects is derived from the same stipulation, namely (21) above. The fact that *wh*-movement, but not scrambling, can escape from certain islands is derived from the different formal treatment of the two movement types. We return to these differences is Section 5.4.10. For simplicity, we will also assume that DPs are barriers for extraction, and defer a discussion of picture-noun phrases to a later study.

5.4.3 Monoclausal Structures

Let us consider the derivation of monoclausal structures. In the derivation of monoclausal structures, all members of a lexical set must be adjoined to the projection from the set. The sentential head feature parameter determines the features of the start symbol, which in turn sets the feature requirements for the root node. In the case of German, we have fixed these features as [+AgrW,+C,+sat], which forces root sentences to have a V2 construction.

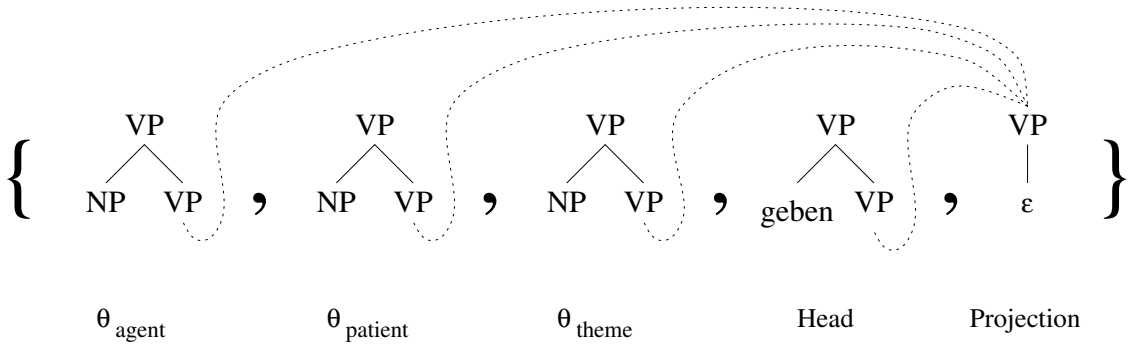


Figure 5.17: Elementary lexical set for *geben* 'to give'

As an example, consider the set for a simple transitive verb such as *geben* 'to give', shown in Figure 5.17. Recall that, formally, lexical entries are vectors of a UVG-DL, i.e., vectors of context-free string-rewriting rules[25] connected by dominance links. We discuss a derivation for a V2 sentence, (4), repeated here as (22).

(22) Der Lehrer [the teacher]$_{\text{NOM}}$ hat das Buch has [the book]$_{\text{ACC}}$ den Kindern [the children]$_{\text{DAT}}$ gegeben given

The teacher has given the book to the the children

The derivation is presented in four steps. First, we assign features to the lexical entry shown in Figure 5.17. Recall that features can be assigned freely to heads as long as the C-selection hierarchy is observed. The feature assignment we choose is shown in Figure 5.18. [+V] is as always supplied by the projection. We have assigned [+Asp] to the lexical head, which is an infinitival form (the past participle). Note that in the feature structures associated with the nodes, we do not indicate [+V] since that is already expressed by the node label (VP), but technically

[25] We will represent context-free string-rewriting rules by trees of height one, and occasionally refer to them as trees.

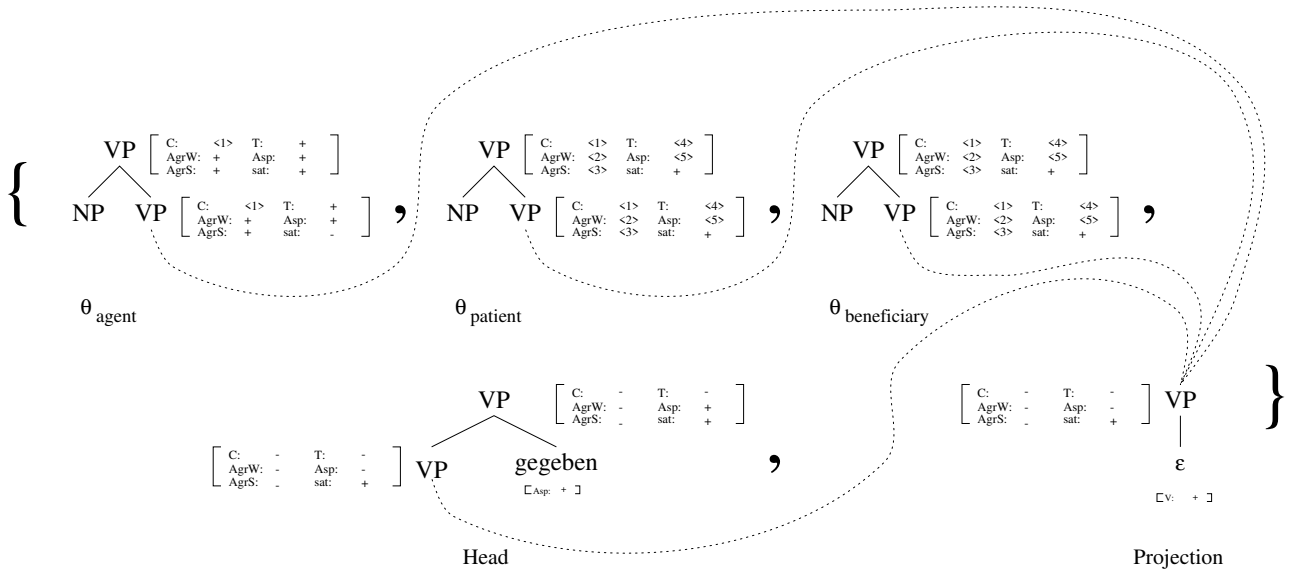


Figure 5.18: Elementary lexical set for *geben* 'to give' with features

[+V] is present at all nodes labeled VP. In the next step, we simply rearrange the elements of the featurized lexical entry, and add an auxiliary [+T-fin,+AgrS,+AgrW,+C] head and the start symbol. The rearrangement suggests a tree, in such a way that the dominance restrictions are obeyed (Figure 5.19). If we connect the pairs of nodes by dotted lines, we have a "quasi-tree" in the sense of Vijay-Shanker (1992). Next, we have simply combined such pairs of nodes to obtain a tree in a TAG with feature structures (Figure 5.20). Each node has a pair of feature structures, as defined in FTAG. This tree is the derivation tree of the derivation in Figure 5.19. Finally, we have unified the pairs of feature structures, and obtained a derived FTAG tree (Figure 5.21). Since all feature structures unify (and assuming we have substituted nominal arguments), we have a complete tree, and the derivation has terminated successfully.

We can also represent the derived sentence in a more succinct matter by using sub- and superscripts. Subscripts indicate the features on the heads (i.e., the features the subscripted head contributes, *not* the cumulative features). Superscript features indicate requirements (spec-head agreement or adjunct requirements). Sentence (22) then can be represented as follows:

(23) [Der Lehrer]$^{[+\mathrm{AgrW}]}$ hat$_{[+\mathrm{T\text{-}fin},\ +\mathrm{AgrS},\ +\mathrm{AgrW},\ +\mathrm{C}]}$ das Buch den Kindern gegeben$_{[+\mathrm{Asp}]}$

As a variant, consider an example in which the *Vorfeld* is occupied by an adverb, *gestern* 'yesterday'. Since we need to derive a sentence with the [+C] feature, we must choose a head that has that feature. On the other hand, the grammatical subject must be in spec-head agreement with a [+AgrS] head. We therefore choose a [+AgrW, +C] head for the auxiliary, which licenses an empty [+T-fin,+AgrS] head. As for the adverb in the *Vorfeld*, we must choose a [+wh] adverb or use the topicalization operator. We get the following derivation:

(24) Gestern$^{[+\mathrm{AgrW},+\mathrm{C}]}$ hat$_{[+\mathrm{AgrW},+\mathrm{C}]}$ der Lehrer $e_{[+\mathrm{T\text{-}fin},+\mathrm{AgrS}]}$ das Buch den Kindern gegeben$_{[+\mathrm{Asp}]}$

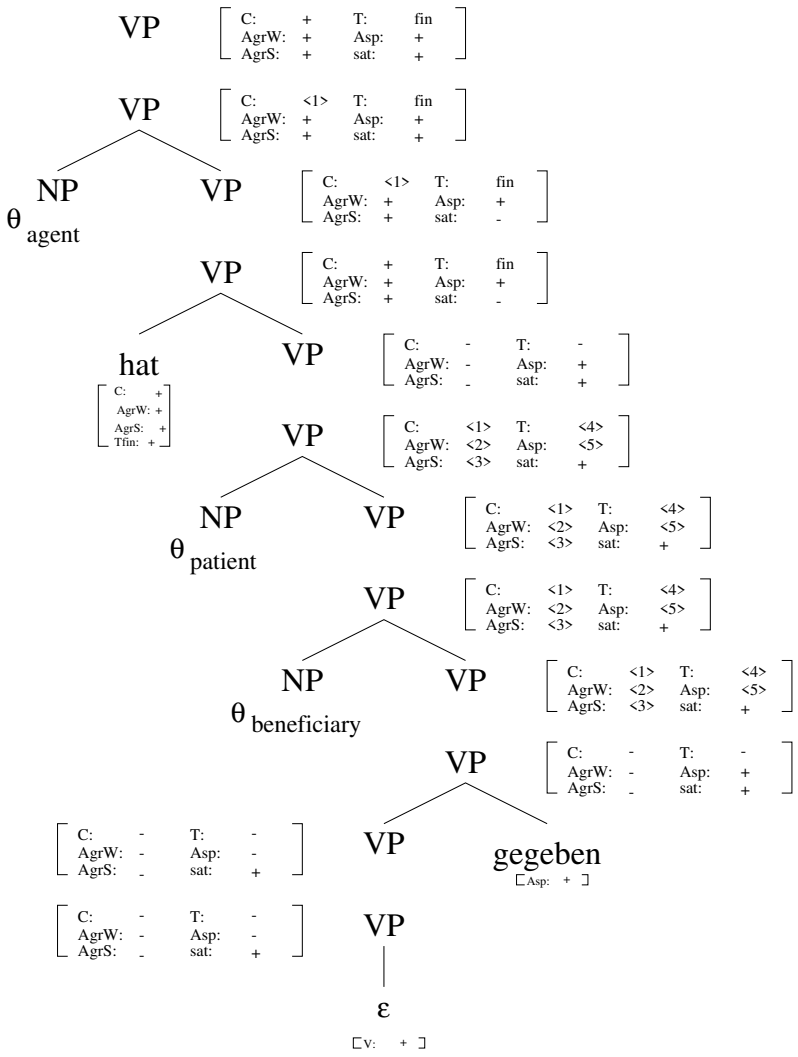


Figure 5.19: Derivation of Sentence (4)

We will now argue that the system can derive all of the possible variations of this construction, and none of the impossible ones.

First, let us discuss *wh*-movement. (These remarks will be equally applicable to topicalization by use of the operator.) Clearly, any [+wh] argument can enter into spec-head agreement with the [+AgrW, +C] head. In the case of the subject, this head must be the [+AgrS, +AgrW, +C] or [+T-fin, +AgrS, +AgrW, +C] head in order to check case under spec-head agreement with AgrS. In the case of adjuncts, we proceed as above in (24).

Now consider scrambling. Observe that there is no restriction on the placement of arguments or adjuncts in the *Mittelfeld*, except that the agent must be in spec-head agreement with the [+AgrS] head. Since all of the [+AgrS, –C] heads are head-final, and since both arguments and adjuncts can adjoin to the saturated [+AgrS] head, the requirement on spec-head agreement for the subject does not affect word order in the *Mittelfeld*. (A similar remark would apply to all arguments if

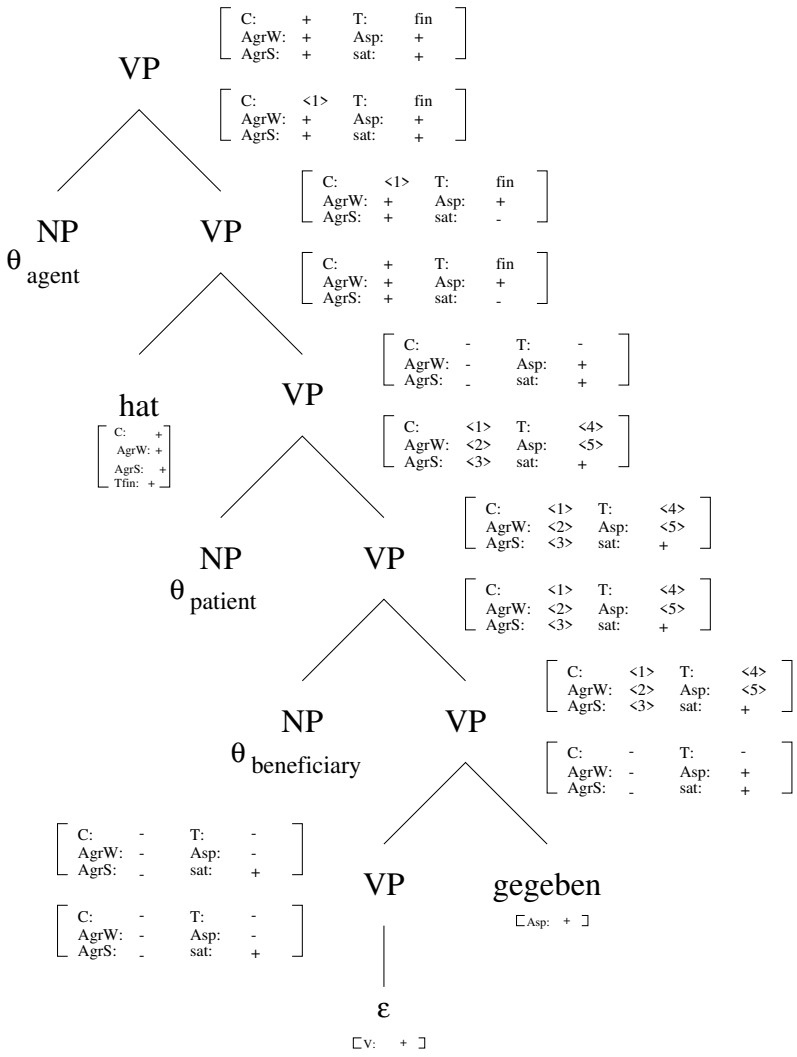


Figure 5.20: Derived syntactic tree of Sentence (4)

we implemented spec-head agreement with the appropriate AgrX heads.) We conclude that we can derive all possible word orders.

We now show that we cannot derive illicit word orders. First observe that no (nominal) arguments or adjuncts can ever appear behind a [+Asp], [+T-fin], or [+T-inf] head-final head. The only remaining problem is a possible violation of the V2 effect. Once we adjoin a [+wh] argument or adjunct to a [+AgrW, +C] head, the feature structure on the top of the adjoined element will have, through unification, the values [+C, +sat]. Therefore, this node will be a barrier, and no argument can move beyond it. (We will see in Section 5.4.9 how long-distance *wh*-movement out of the *Mittelfeld* is handled.) In the case of adjuncts, the *Mittelfeld* variety will fail to adjoin because of its [–C] requirement, and the [+wh] variety cannot adjoin since it requires a [+AgrW, –sat] node. Finally, we cannot adjoin another [+AgrW] head without violating the C-selection hierarchy. Thus, the V2 effect is derived. We conclude that we can derive exactly the correct

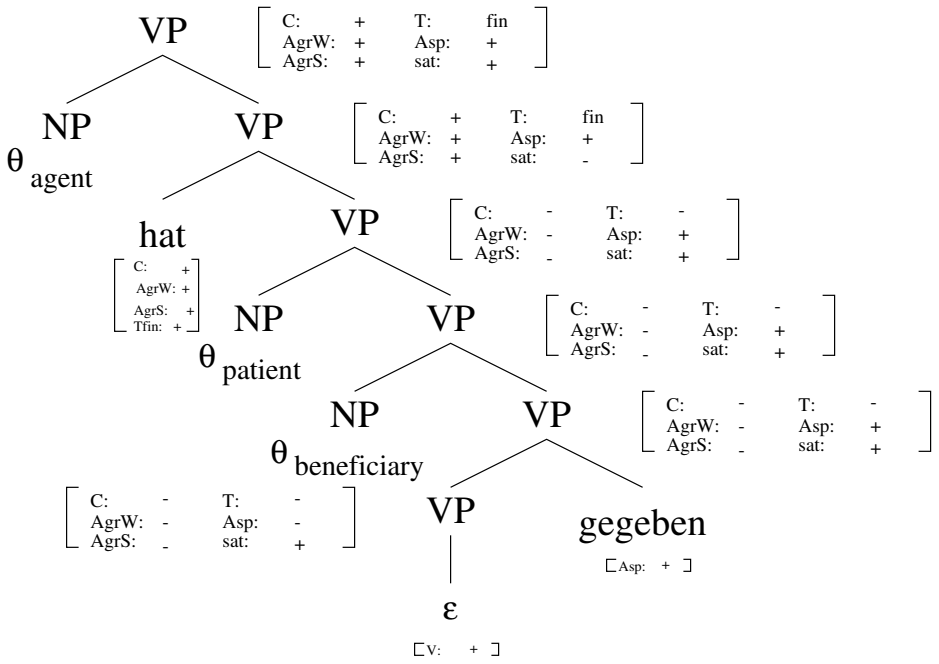


Figure 5.21: Final derived tree for Sentence (4)

range of word order variations in monoclausal structures.

5.4.4 Non-finite Embedded Clauses

In discussing multi-clausal structures in the proposed framework, two steps are necessary (see Section 5.3, page 138):

1. The elementary clausal structures must be derived from the lexicon (lexical derivation in UVG-DL-Δ).
2. The complete sentence must be derived by combining the elementary clausal structures, which are sets of trees (syntactic derivation in V-TAG-Δ).

Both of these derivation steps must be legal derivations as defined by the respective formalisms, but frequently the result of the lexical derivation yields structures that are "unstable" in the sense that clashing feature structures force further adjunctions.

We will begin by discussing subcategorization for non-finite clauses. (We consider embedded finite clauses in Section 5.4.8.) Consider a simple sentence with an embedded infinitival clause:

(25) Moritz hat dem Schneider die Brücke zu reparieren versprochen
Moritz has [the tailor]$_{\text{DAT}}$ the bridge to repair promised

Moritz has promised the tailor to repair the bridge

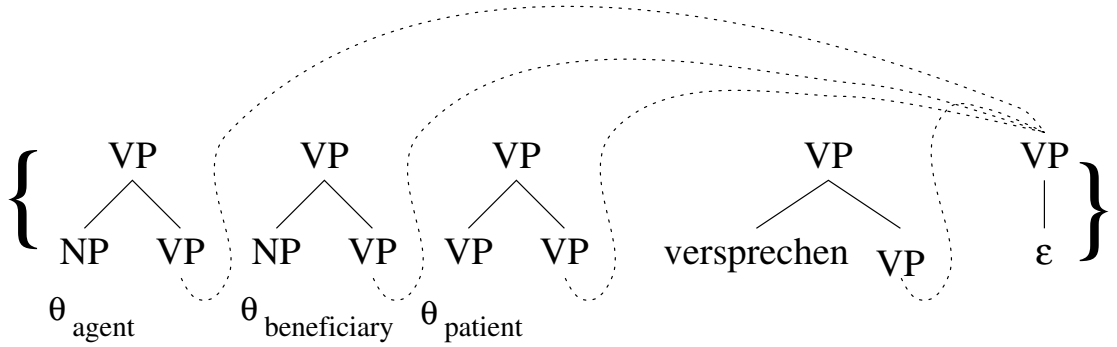


Figure 5.22: Lexical set for *versprechen* 'to promise'

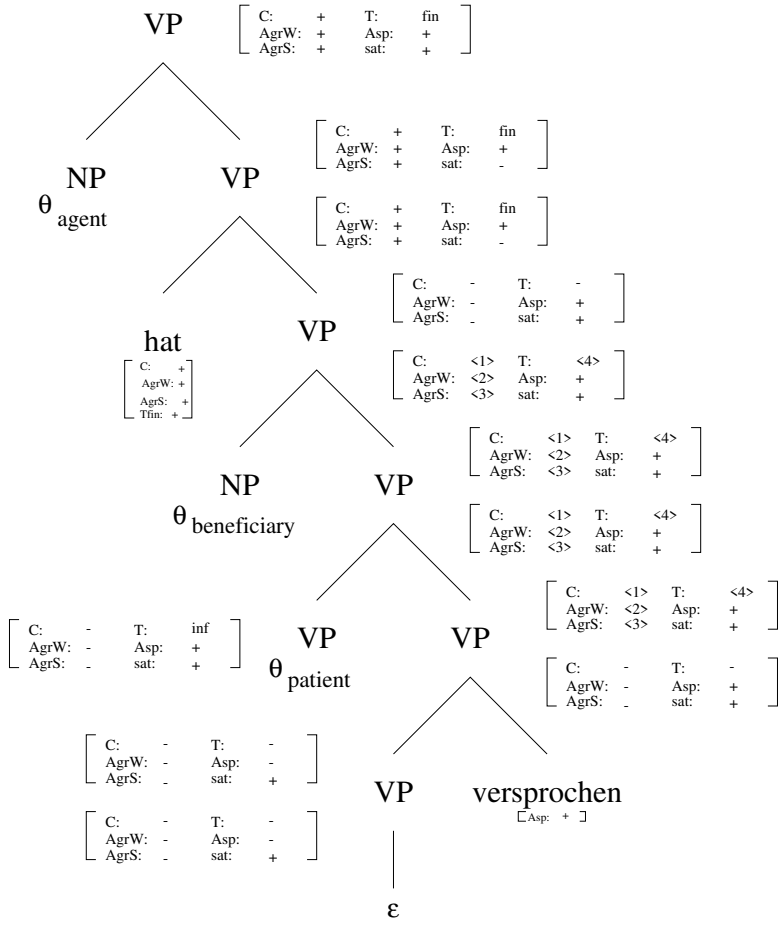


Figure 5.23: Syntactic tree for *versprechen* 'to promise'

We will develop a derivation for this sentence. As an example of a lexical entry of a verb that subcategorizes for a clause, consider Figure 5.22. We have not yet assigned features to the head, but we see that one of the frontier nodes on a subcategorization structure is not an NP, but a VP. We will indicate the subcategorization requirement on this node. We can derive the structure

shown in Figure 5.23[26] along the lines discussed in Section 5.4.3. We have assigned the lexical head the features [+Asp], and have also adjoined an auxiliary head with features [+T-fin, +AgrS, +AgrW, +S]. The VP node on the frontier has the features [+V, +Asp, +T-inf], indicating subcategorization for a *zu*-infinitive. (These subcategorizations must be lexically licensed.) We have not included the start symbol of the lexical grammar (as can be seen from the fact that the root node only has one feature structure), so that this is an auxiliary tree.

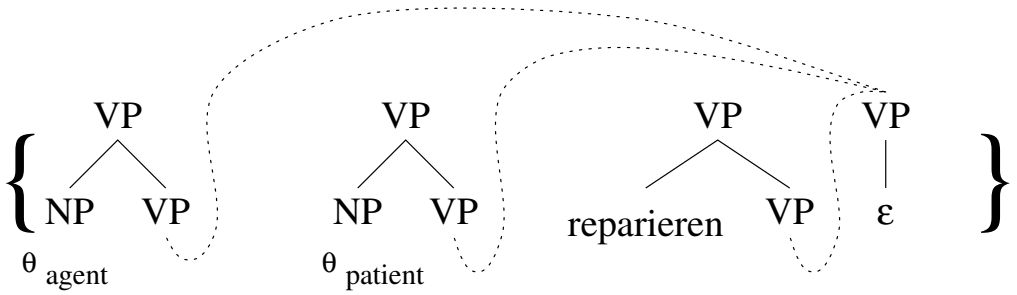


Figure 5.24: Lexical set for *reparieren* 'to repair'

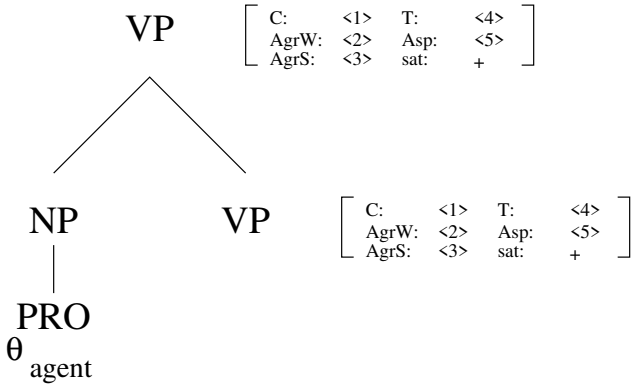


Figure 5.25: PRO argument tree

Let us now turn to the derivation of the embedded clause. We start with the lexical tree set shown in Figure 5.24. Since we want to combine it with a matrix clause subcategorizing for an infinitival, we cannot use a [+T-fin] or [+C] head. This means that we cannot adjoin the subject, since it requires spec-head agreement with a [+AgrS] head, which must be adjoined above the [+T-fin] head. By the projection principle (which, recall, is a consequence of the formalism, not actually a principle), we cannot not adjoin the subject; instead, we use a special PRO-feature on the subject which forces realization of the subject as an empty category and which voids its requirement for spec-head agreement with [+AgrS].[27] A PRO-tree is shown in Figure 5.25; the feature [+sat] on the lower VP indicates that it is not seeking to enter (and in fact cannot enter) into spec-head agreement. The fully derived tree for the embedded clause is shown in Figure 5.26.

[26] In this and subsequent figures, we omit the "[V: +]" feature from the origin of the projection.

[27] Watanabe (1993) has suggested that PRO must agree with a special PRO-agreement head. We could implement that suggestion in this framework by making the PRO-agreement head C-select for a [+T-inf] head.

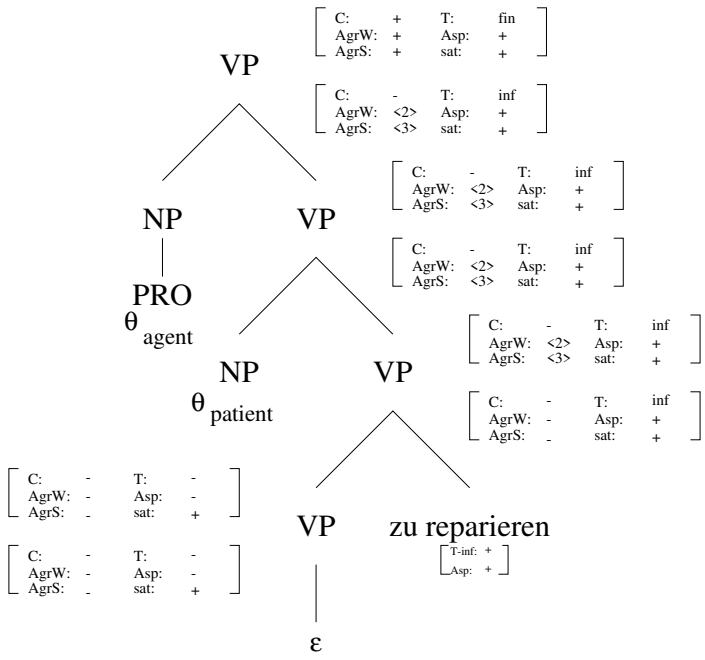


Figure 5.26: Syntactic tree for *reparieren* 'to repair'

We now syntactically derive the biclausal structure by adjoining the matrix clause into the infinitival clause. In principle, we could perform the adjunction at any of the VP nodes in the embedded clause. However, the root node of the matrix clause bears the [+C] feature, which prevents adjunction at any node except the root node of the *reparieren* tree. The result is shown in Figure 5.27.

We will make two observations about this derivation. First, it should be noted that from a formal point of view, this syntactic derivation is indistinguishable from a derivation in which the matrix clause is assigned the start symbol during the lexical derivation, and not the embedded clause. Then the only possible syntactic derivation is a substitution of the embedded clause into the frontier VP node of the matrix clause. The resulting derived trees are identical under both derivations.

Second, in our example, both lexical derivations (of the matrix and embedded clause) assemble all phrase-structure fragments of the lexical set into a single tree. Formally speaking, this means that the grammar used for the syntactic derivation is a simple TAGs. However, we have argued in Chapter 3 that simple TAGs are not sufficient to handle long-distance scrambling. We can therefore also use lexical derivations in which we have adjoined the heads and the clausal arguments into the verbal projection, but not the nominal arguments. This means that the result of the lexical derivation, the syntactic structure, is not a tree, but a set of trees equipped with dominance links. The dominance links are of course exactly those inherited from the lexical representation. In the syntactic derivation, the definition of V-TAG does not impose any order on adjunctions. We can therefore first adjoin the (argumnetless) *versprechen* tree into the (argumnetless) *reparieren* tree. The result is shown in Figure 5.28 (we have omitted the matrix auxiliary and the feature structures for clarity). The result bears a striking resemblance to the "clause union" analysis of

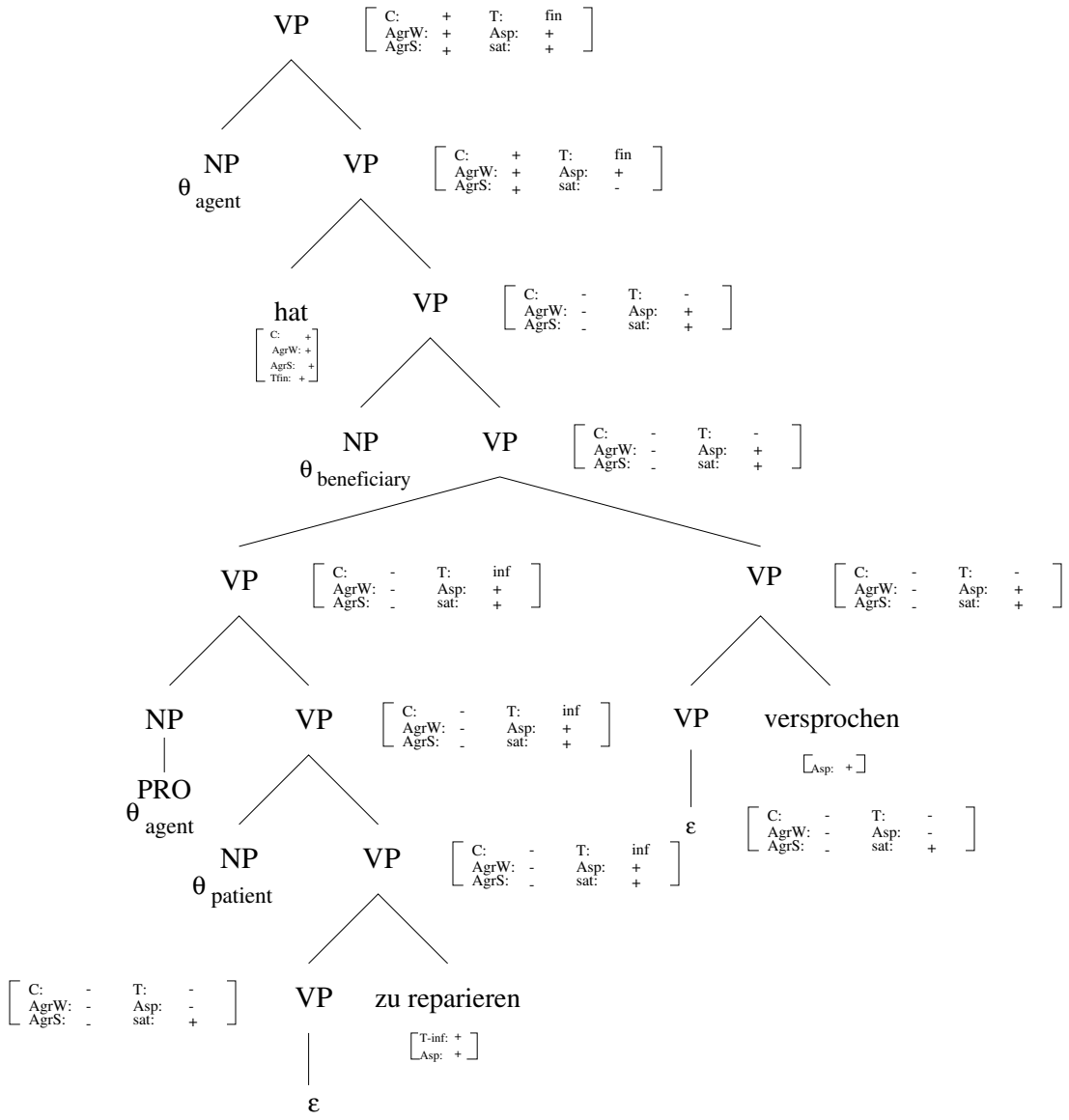


Figure 5.27: Derived sentence with center-embedded infinitival

Evers (1975) and subsequent work, which we rejected in favor of the "syntactic"[28] analysis of Kroch and Santorini (1991) (Section 5.4.1):

- The two verbs clearly form a constituent. If all nominal arguments are adjoined above the matrix verb (as is possible, though not obligatory), then this constituent is preserved in the subsequent derivation steps. Furthermore, the θ–grids of the two verbs have been merged. We can therefore interpret the set in Figure 5.28 as verb complex.

[28] We use the term "syntactic" in quotes to use it in the sense of Kroch and Santorini (1991), namely (roughly) "using syntactic derivational machinery" and opposed to "morphological". It should not be confused with the notion of "syntactic derivation" which has been given a specific meaning in the framework being elaborated here.

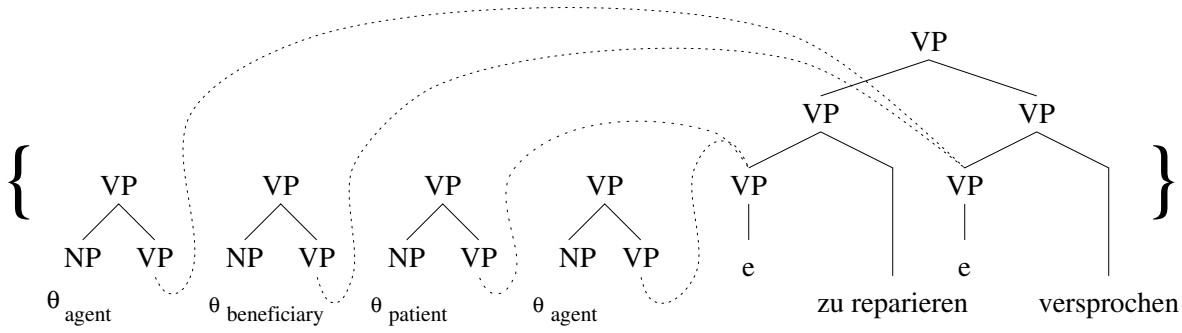


Figure 5.28: The "clause union" analysis in V-TAG

- Clearly, there is no intervening VP[+C, +sat] and hence no barrier, and the embedded nominal argument can freely mingle with the matrix arguments. Thus, we can interpret the representation in Figure 5.28 as representing a state in which clause pruning has occurred.

Of course, Figure 5.28 and the associated derivation does not implement the analysis of Evers (1975) faithfully. In particular, the "verb complex" is not dominated by a lexical category (in fact, there are none in this system), and it does not in any way appear to have undergone a special morphological process. And, as we have mentioned, the clauses were never "pruned", since this is a non-transformational system. However, we would like to claim that this derivation captures the underlying intuition of Evers's analysis, which has proved compelling to most native speaker linguists, even if Evers's arguments for it have not necessarily. What this analysis suggests is that the difference between Evers's "morphological" analysis and Kroch & Santorini's "syntactic" analysis is really an artifact of the systems that these authors used to express them. Evers, working in a transformational framework whose formal origins are in context-free grammars, could not adequately represent the multiclausal, scrambled *Mittelfeld* and the predicate-argument relations that hold in it except by saying that it is essentially like a monoclausal *Mittelfeld*. Kroch & Santorini, on the other hand, could not possibly implement a "morphological" analysis in the simple TAG framework that they chose, since the elementary structures are trees, which cannot be modified in the formal derivation. However, if we choose to assemble phrase-structure incrementally, we see that both the clause-union analysis (and its underlying intuition) and the "syntactic" analysis are an effect of the derivation, not expressible explicitly in the competence grammar on their own.

Further levels of embeddings are achieved by additional adjunctions at the root node. For example, consider the lexical entry for *versuchen* 'to try' and the tree lexically derived from it shown in Figure 5.29. If we first adjoin the *versuchen* 'to try' tree to the root clause of the *reparieren* 'to repair' tree, and subsequently adjoin the *versprechen* tree to the root node of this derived tree (which is in effect the root node of the original *versprechen* tree), then we obtain three levels of embedding. Clearly, the adjunction of clauses can be iterated.

We discuss subcategorization for finite clauses in Section 5.4.8, page 179.

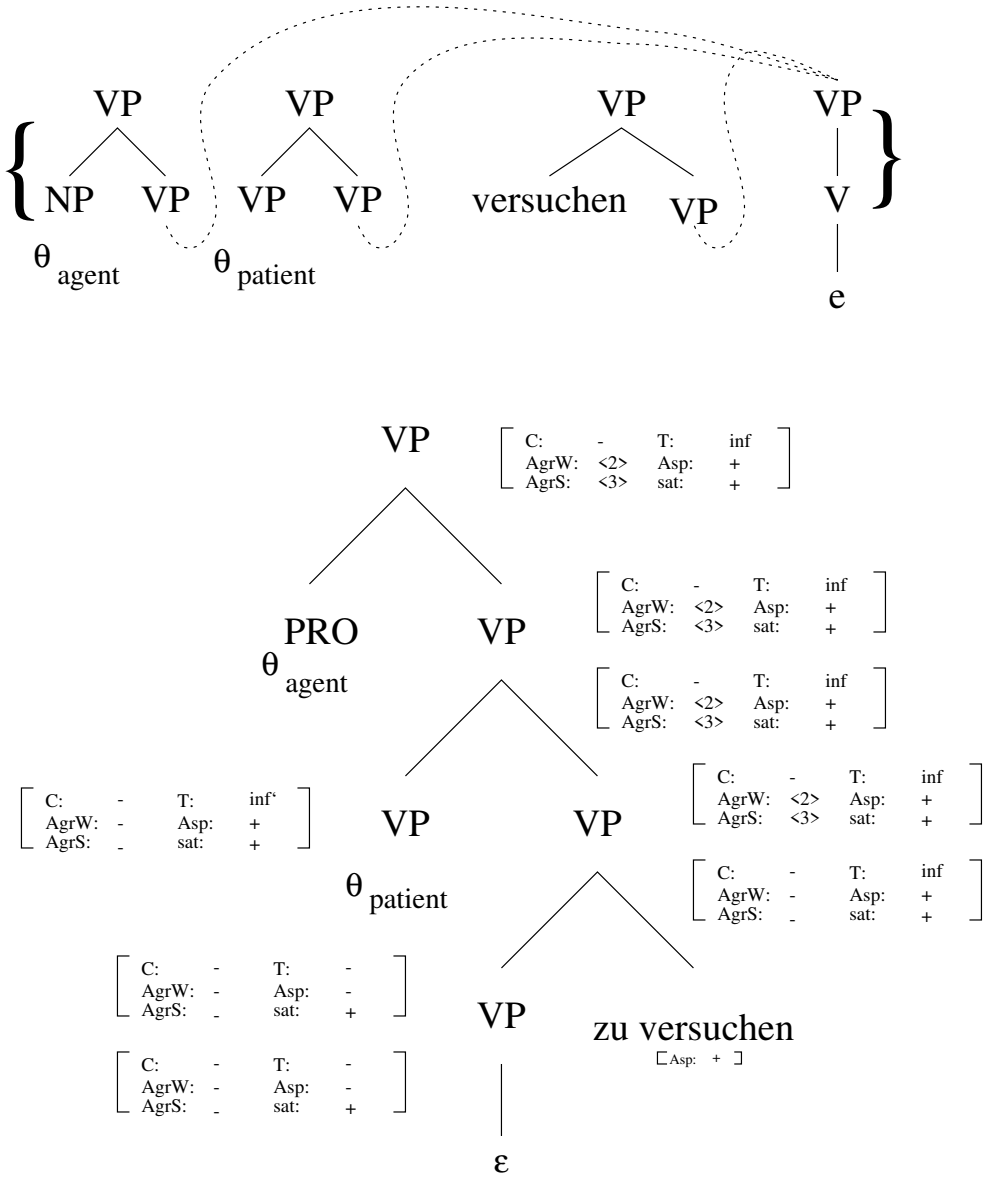


Figure 5.29: Lexical tree set and derived tree for *versuchen* 'to try'

5.4.5 Extraposition

As discussed in Section 2.1.3, embedded clauses (but not nominal arguments) can be generated in the *Mittelfeld*, or they can appear behind the finite verb ("extraposition"). They cannot appear between a non-finite verb and a finite verb, as shown in (25):

(26) * Weil ich versprochen, das Fahrrad zu reparieren, habe
because I promised the bike to repair have

Intended meaning: Because I have promised to repair the bike

On the other hand, constructions such as (23), page 19, show that it is possible to scramble out of extraposed clauses into the matrix clause. We conclude that extraposed clauses must be adjoined above the [+T] head (be it [+fin] or [-fin]), but that they need not adjoin to the [+C] head. We will express extraposition by allowing clausal subcategorization trees in lexical sets of the sort in Figure 5.30 on the left to change directionality, but only if the feature [+T] is added to the lower VP spine node, as shown in Figure 5.30 on the right.

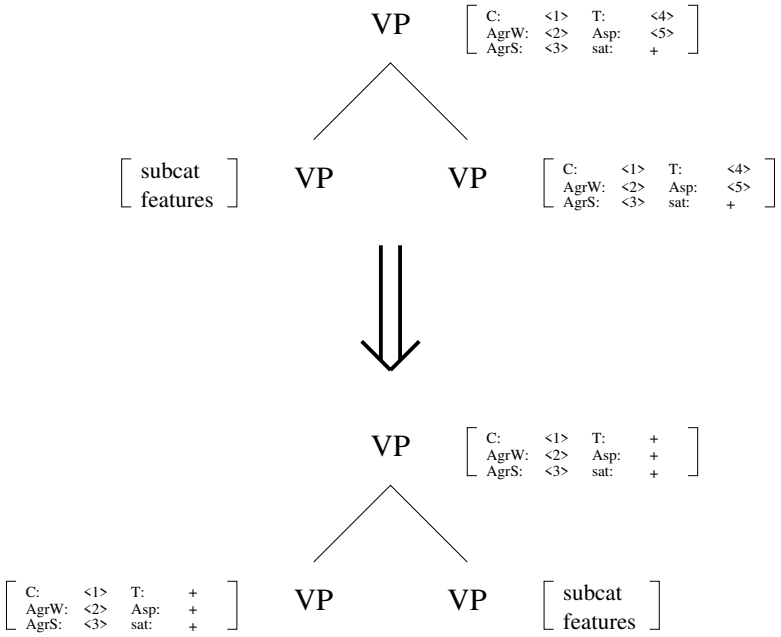


Figure 5.30: Extraposition in elementary lexical sets

Figure 5.31 shows the derived tree for the extraposed version of (25):

(27) Moritz hat Max versprochen, die Brücke zu reparieren

5.4.6 Long-Distance Scrambling and the Third Construction

In this section, we will show how to derive the range of sentences that involve both by long-distance scrambling and extraposition ("Third Construction") presented in Figure 2.1, page 18. In the previous two sections, we have discussed subcategorization for clauses and extraposition. We have also seen that we need not adjoin all arguments during the lexical derivation. This gives us the machinery to implement some of the preliminary analyses sketched in Section 2.2. Figure 5.32 shows a syntactic tree set derived from the lexical set for *versprechen* 'to promise' in Figure 5.22 (except that we have omitted the optional beneficiary θ–role), and Figure 5.33 shows a syntactic tree set derived from the lexical set for *reparieren* 'to repair' in Figure 5.24. In both of these tree sets, the overt nominal arguments have not been adjoined into the projection. (There is no point in scrambling empty arguments.) The syntactic tree set for *versprechen* 'to promise' contains two auxiliary trees; the nominal argument is an auxiliary tree, and since we

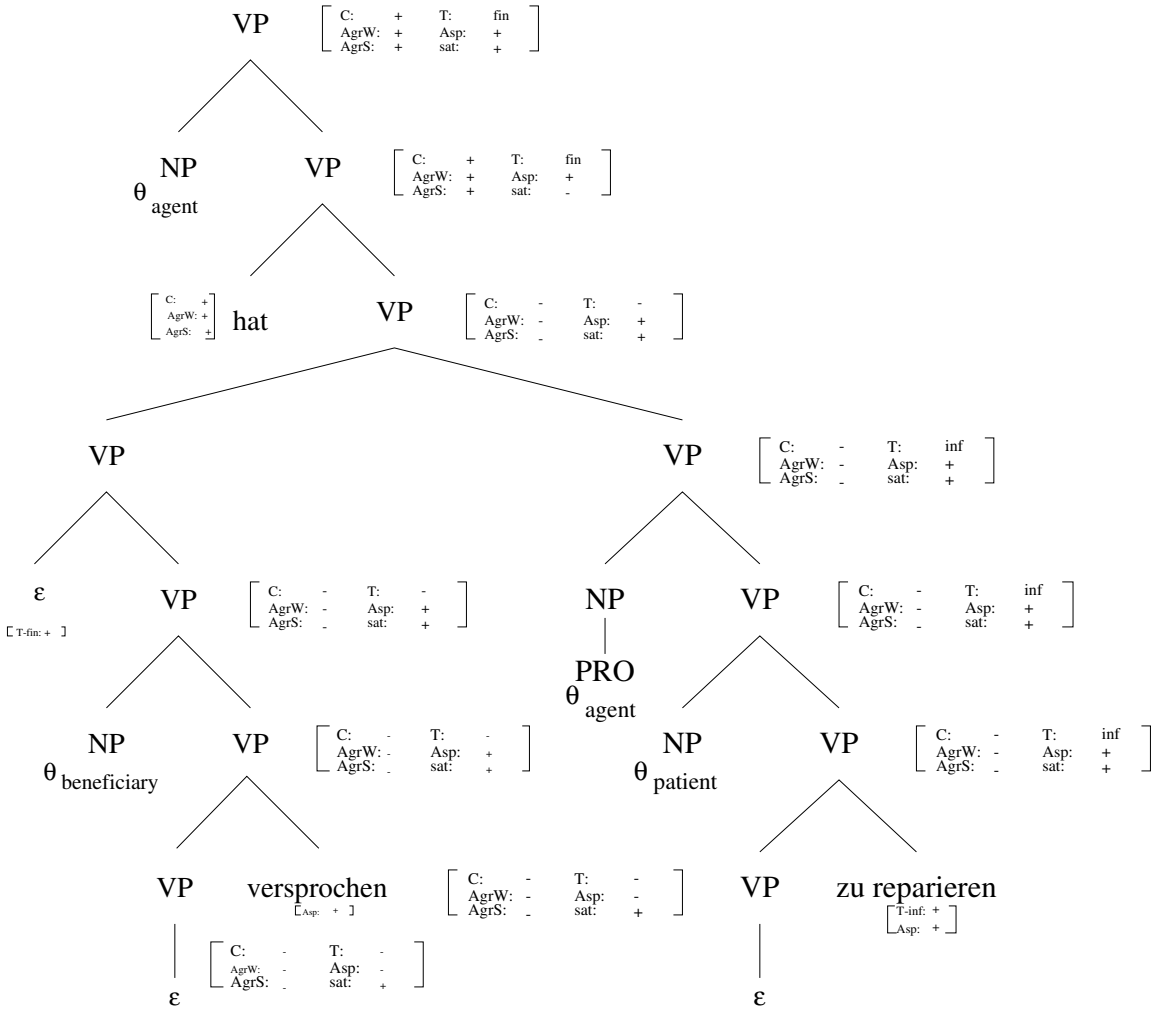


Figure 5.31: Derived sentence with extraposed infinitival

adjoin the clausal subcategorization structure into the projection, creating a footnode, the main verbal tree is also an auxiliary. Note that we have adjoined the clausal subcategorization in a configuration for center-embedding, not extraposition. The tree for *reparieren* 'to repair' contains an auxiliary tree for the nominal argument and the verbal projection, which is an initial tree. Finally, for the intermediate *versuchen* 'to promise' clause in our examples, observe that it has no overt nominal argument, and that therefore we can simply use the derived syntactic tree at the bottom of Figure 5.29.

Let us first look at scrambling without extraposition. Since there are only two nominal arguments, there is only one possibility of nominal scrambling in a center-embedded construction, giving us the order $N_3N_1V_3V_2V_1$ (sentence (ii) of Figure 2.1). The syntactic derivation proceeds as in the center-embedded case without scrambling (sentence (i)): by adjunction of the *versuchen* tree into the root node of the *reparieren* projection tree, and by subsequently adjoining the *versprechen* clause into the root node of the tree that results form the previous adjunction. We then are left

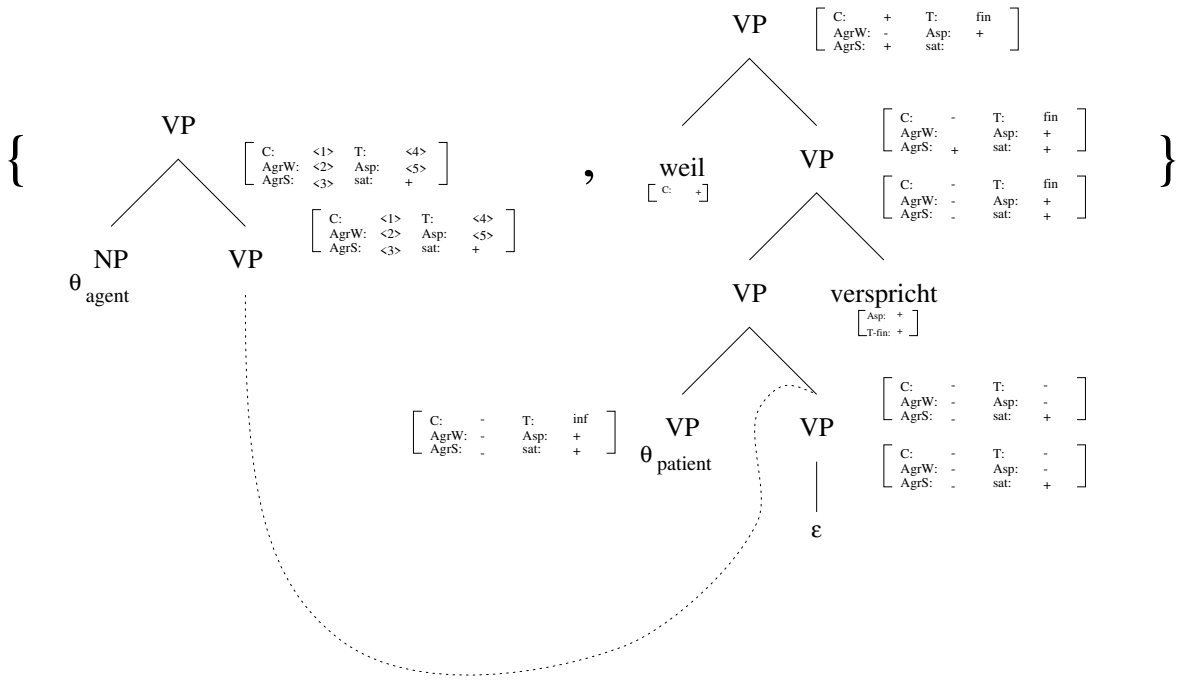


Figure 5.32: Syntactic tree set for *versprechen* 'to promise'

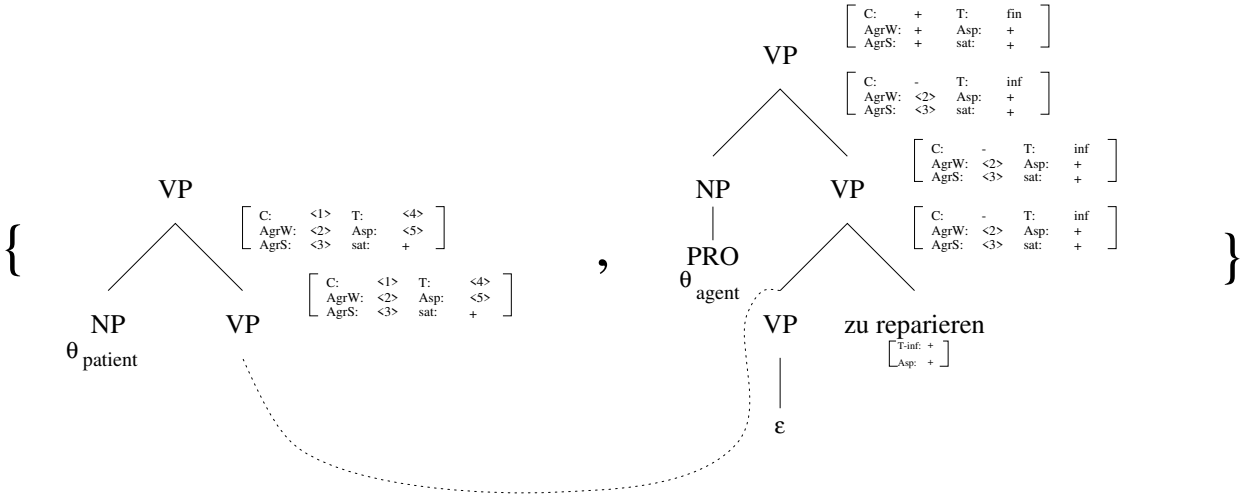


Figure 5.33: Syntactic tree set for *reparieren* 'to repair'

with the two arguments, the matrix θ_{agent} and the embedded θ_{patient}. Because of the dominance links, we must adjoin the matrix θ_{agent} into its own clause, but we can adjoin the most deeply embedded θ_{patient} into any of the three clauses, and in particular, we can adjoin it above the matrix θ_{agent}, yielding the word order we are aiming for. (We will discuss in Section 5.4.8 why we cannot adjoin either argument to the complementizer of the matrix clause.)

We can also extrapose one or both of the clauses, and then we still have the option of scrambling or

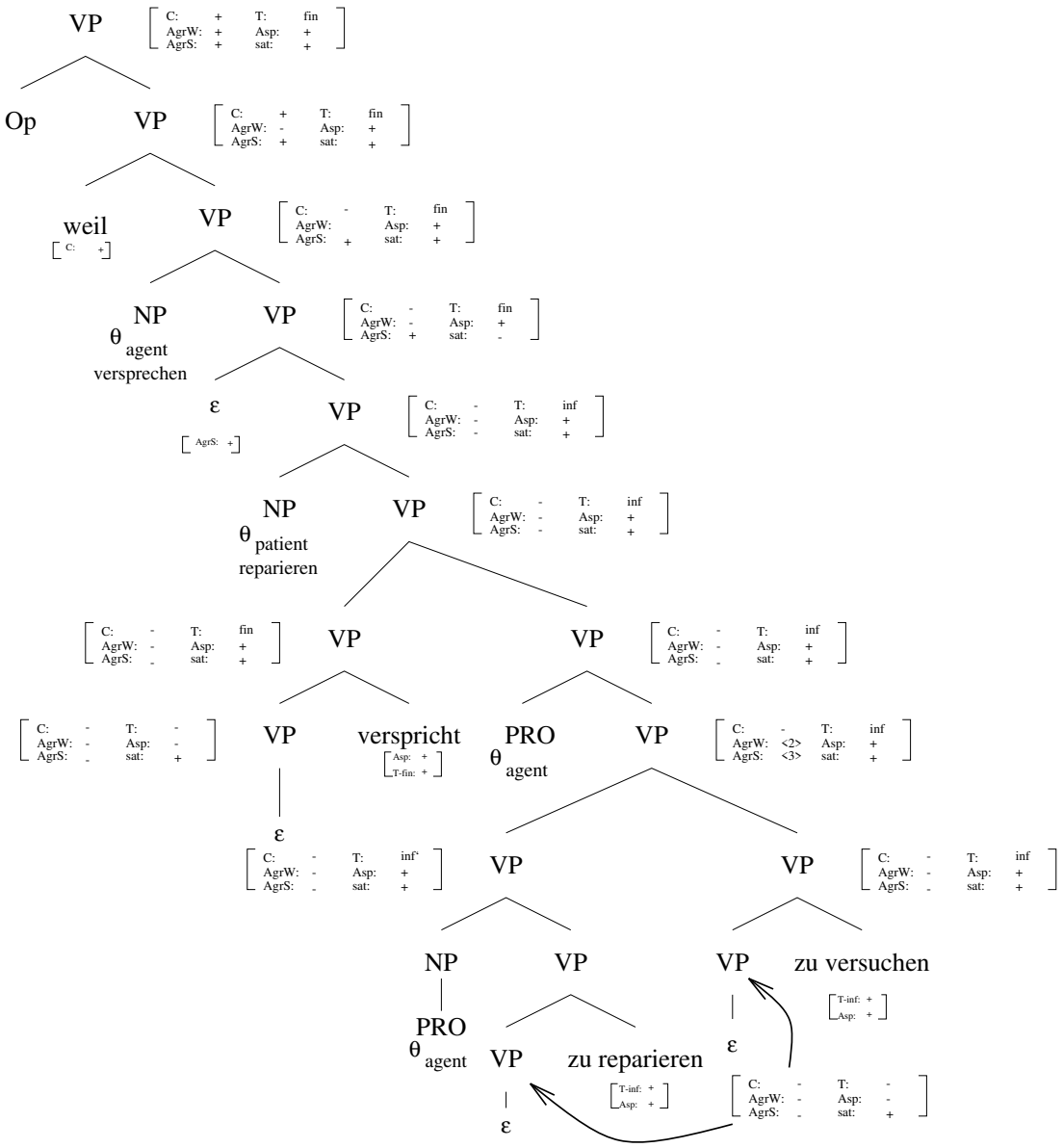


Figure 5.34: Derived tree for sentence (xviii)

not, since extraposition will not alter the structural configurations in a way to restrict scrambling. Observe that if we have extraposed the most deeply embedded clause (*reparieren*) behind its governing verb (*versuchen*), then scrambling the embedded θ_{patient} into the *versuchen* clause no longer is string-vacuous. Overall, using just extraposition and scrambling, we can derive 12 of the 30 sentences in Figure 2.1, namely (i) to (v) and (xiii) to (xix). Figure 5.34 shows, as a representative example, the derived tree for (xviii), with the order $\mathsf{N}_1\mathsf{N}_3\mathsf{V}_1\mathsf{V}_3\mathsf{V}_2$.

5.4.7 Clausal Scrambling

We have now provided the tools to derive 12 of the 24 licit sentences shown in Figure 2.1. The remaining 12 licit sentences involve clausal scrambling, which is the subject of this section. In the case of clausal scrambling, we will make a distinction between local and long-distance scrambling (of the clause). In the case of local scrambling of clauses, we can simply derive a different syntactic tree set by placing the clausal subcategorization marker higher in the derived auxiliary tree. A syntactic tree set for *versprechen* 'to promise' in which its embedded clause is locally scrambled past the matrix subject is shown in Figure 5.35. We can use it just the same way as we use the equivalent tree set for a non-scrambled center-embedded clause. This will give us orders such as $\mathsf{N}_3\mathsf{V}_2\mathsf{V}_3\mathsf{N}_1\mathsf{V}_1$, which is sentence (xii), and whose derived tree is shown in Figure 5.36. We omit the features on the interior nodes, since they are very similar to those in Figure 5.34.

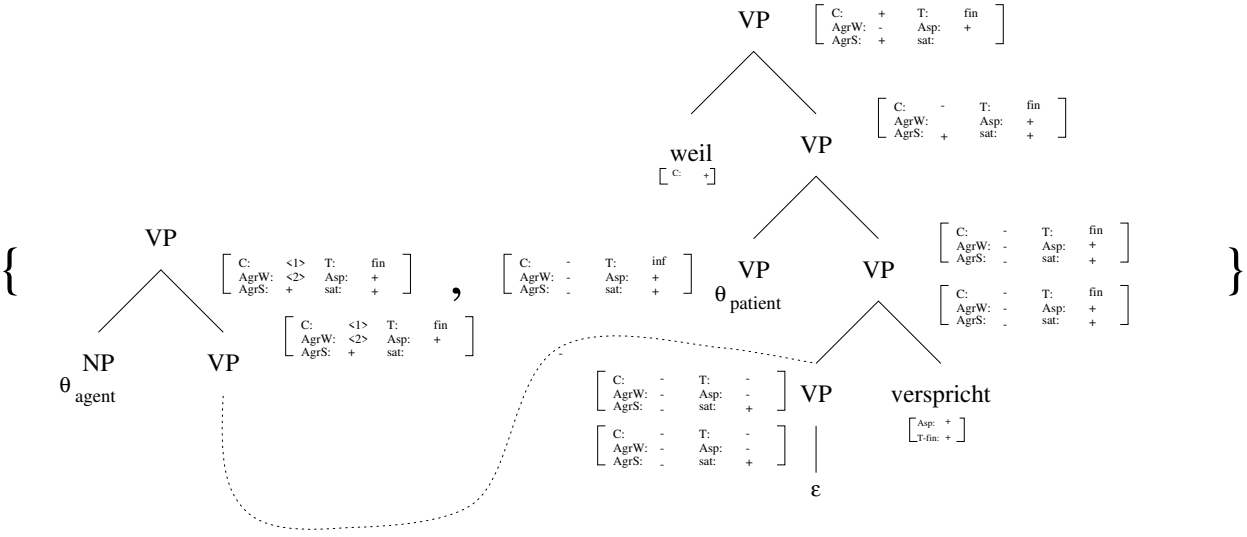


Figure 5.35: Syntactic tree set for *versprechen* 'to promise' with local clausal scrambling

The addition of local scrambling lets us derive sentences (vii), (viii), and (xii). We have not yet accounted for nine of the 24 licit sentences; for these, we will require long-distance scrambling. Recall that in Section 5.4.6, we pointed out that instead of adjoining the matrix clause into the embedded clause, we can also substitute the embedded clause into the matrix clause. The only difference is that in the former approach, we assign the start symbol to the embedded clause, while in the latter approach, we assign it to the matrix clause. Either approach is perfectly compatible with the definition of the lexical derivation, and while the two syntactic derivations differ, their result is indistinguishable. In order to allow clausal long-distance scrambling, we will follow the second approach: we will assume that embedded clauses are substituted into the clausal subcategorization structures of their matrix clause. Furthermore, we will assume that the clausal subcategorization structure is not adjoined into the matrix clause's projection tree during the lexical derivation. Instead, it remains separated from the projection, though of course it is connected to it by a dominance link. Thus, clausal long-distance scrambling works in the same way as nominal long-distance scrambling does.[29] Observe that we have not made any special

[29] This analysis of clausal scrambling is identical to that proposed by Lee (1991). See also (Rambow and Lee,

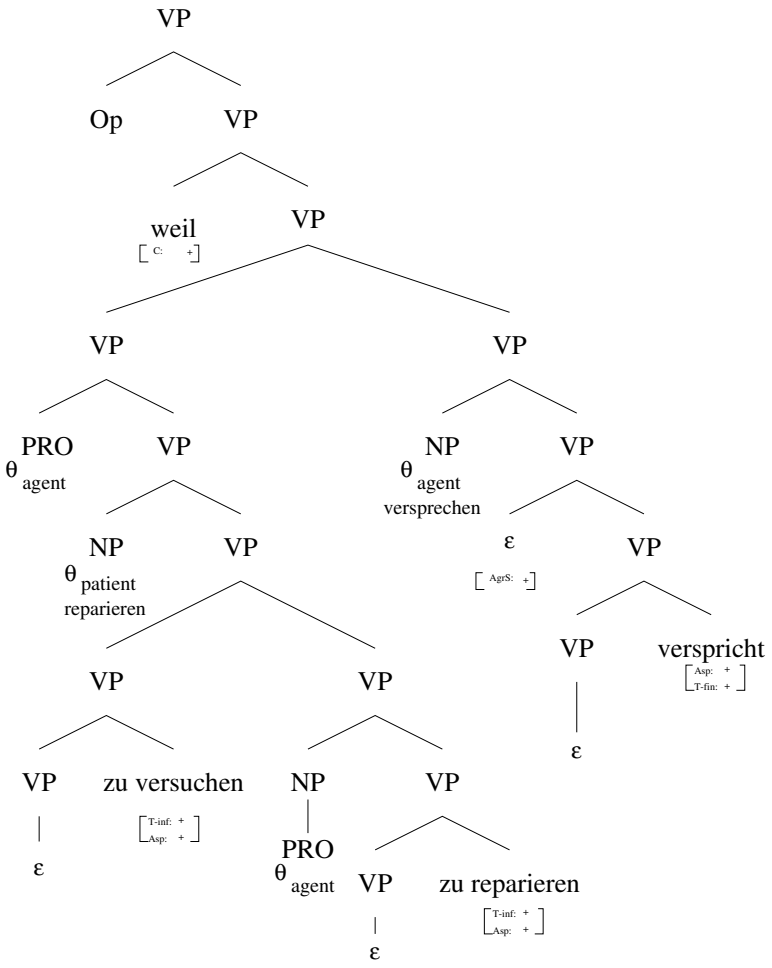


Figure 5.36: Derived tree for sentence (xii)

stipulations for it, we are just exercising the options provided by the lexical derivation step. We show a syntactic tree set for *versuchen* 'to try' in which the clausal subcategorization structure has not been adjoined into the projection in Figure 5.37; furthermore, we have chosen extraposition.

As an example, we will step through the derivation of sentence (xxvi) from Figure 2.1, which in abstract is $\mathsf{N}_3\mathsf{N}_1\mathsf{V}_2\mathsf{V}_1\mathsf{V}_3$. We have a choice between substituting the embedded *versuchen* clause into the matrix *versprechen* 'to promise' clause, or adjoining the matrix clause into the embedded clause. Let us chose the substitution approach. We start with Figure 5.32, page 175. We substitute into the VP node on the frontier of the projection *versprechen* tree the projection *versuchen* tree from Figure 5.37. We then adjoin the clausal subcategorization structure of the *versuchen* set into the *versprechen* tree, and then substitute into it (to the right of the spine) the *reparieren* projection tree from Figure 5.33. Finally, we adjoin the two nominal arguments into the matrix *versprechen* clause in the appropriate order, and we are done. The derived tree is shown in Figure 5.38.

1994).

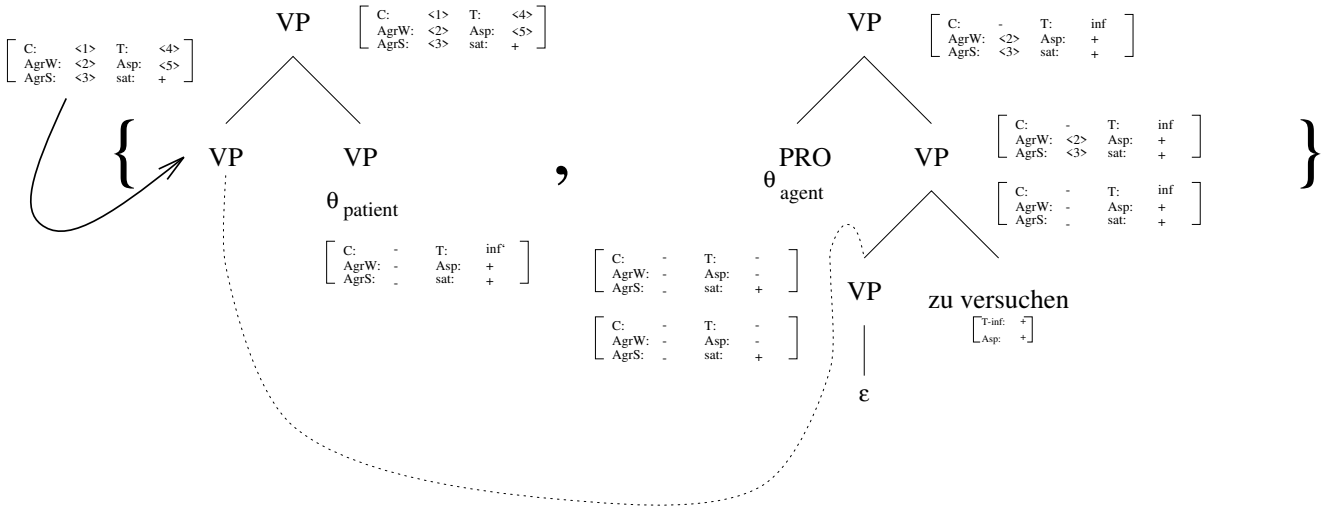


Figure 5.37: Syntactic tree set for *versuchen* 'to try' with auxiliary clausal subcategorization structure

By long-distance scrambling clauses, we can derive the remaining licit sentences: (vi), (xx) to (xxvi), (xxviii), and (xxx).

To conclude this section, we will briefly show how the dominance links preclude the derivation of one of the sentences marked '*' in Figure 2.1. Take as an example sentence (xxiv), represented as $\mathsf{N}_1\mathsf{V}_2\mathsf{N}_3\mathsf{V}_1\mathsf{V}_3$. Since V_3 follows V_1, it must be extraposed, and since V_2 precedes V_1, it must have first been scrambled long-distance. We are therefore in a similar situation as with the preceding case, Sentence (xxvi). Since V_3 has long-distance scrambled, it must be substituted into a clausal subcategorization structure which is linked by a dominance link to the VP node in the the projection of the *versuchen* clause V_2, and therefore the projection of the V_3 clause must be adjoined above the projection of the V_2 clause (*versuchen*). We can assume that N_1 is adjoined as far up as possible. The dilemma is now clearly visible in Figure 5.39. The bold lines indicate the spine along which we may adjoin N_3: it can be seen that the desired word order – with N_3 between V_2 and V_1 – is impossible. (Observe that this is exactly the case which we suspect is difficult to rule out in a FO-TAG; see Section 3.3, page 43.)

5.4.8 Finite Embedded Clauses

After exhaustively examining the word order possibilities in embedded infinitival clauses, we now turn to finite clauses. Subcategorization for a finite clause looks just like subcategorization for a non-finite clause, except that the subcategorization requirements on the frontier VP node are different. We will assume that they are [+C, +sat], representing the fact that the semantic contribution of the [+C] head is required (i.e., a determination of modality or illocutionary force), and that furthermore no unsatisfied spec-head agreement may be present. We can now adjoin this tree into the root node of a V2 tree (which can function as a matrix clause on its own) such as the one given in Figure 5.21, page 166, resulting in an embedded V2 sentence:

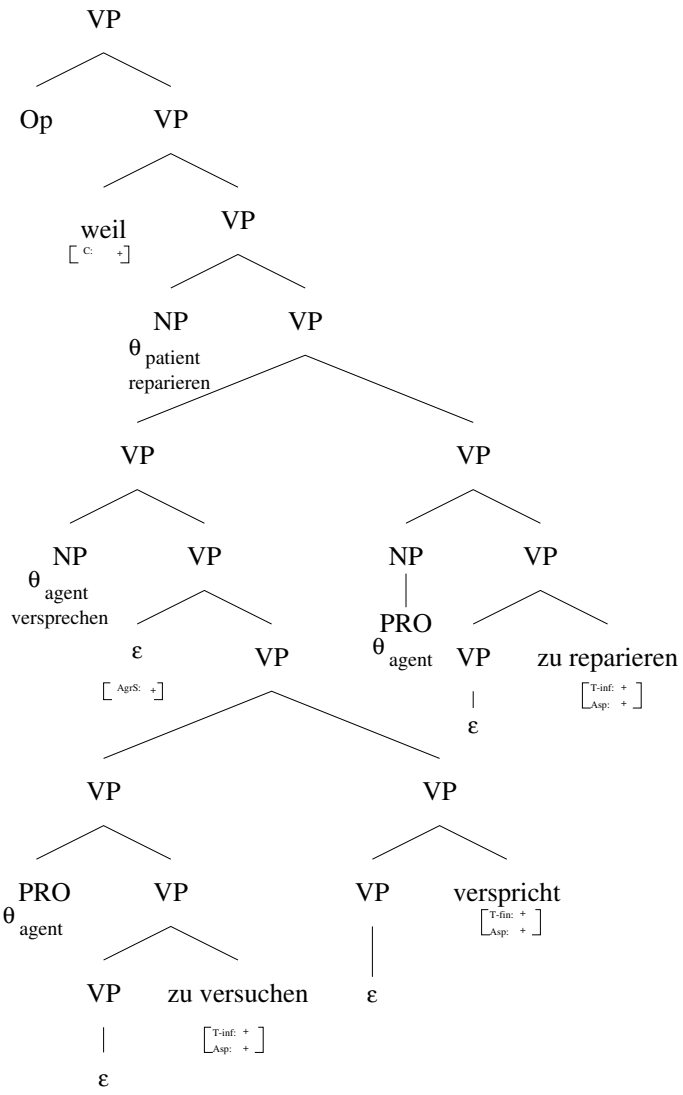


Figure 5.38: Derived tree for sentence (xxvi)

(28) Ich behaupte, der Lehrer hat die Bücher den Kindern gegeben
I claim the teacher has the books the children given

I claim that the teacher has given the books to the children

Instead of using a V2 structure for the embedded clause, we can also use an overt complementizer. The syntactic tree used is shown in Figure 5.40. We again adjoin the matrix clause at the root node, and obtain the derived tree in Figure 5.41. Observe that the AgrW feature is not present in the latter approach.

(29) Ich behaupte, daß der Lehrer die Bücher den Kindern gegeben hat
I claim that the teacher the books the children given has

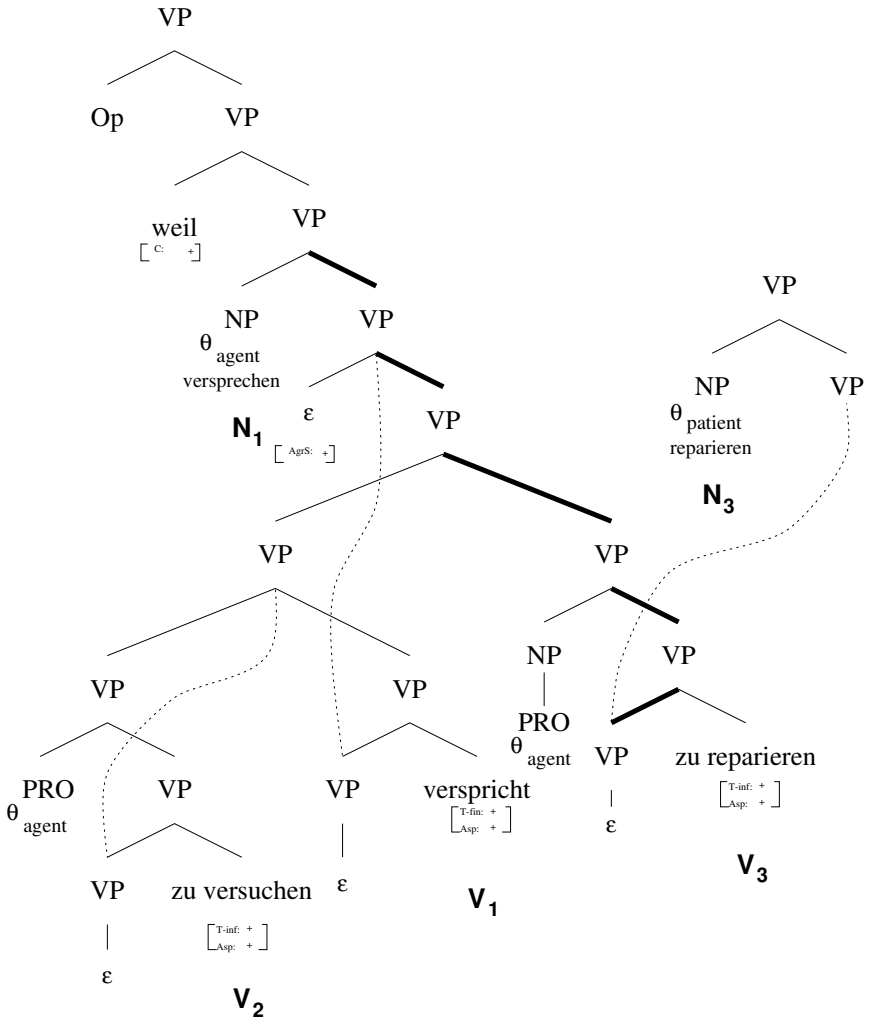


Figure 5.39: There can be no derived tree for sentence (xxiv)

I claim that the teacher has given the books to the children

We observe that in either case, the root node of the embedded clause is marked with the integrity constraint (Δ), since it carries the features [+C,+sat]. In the case of the embedded *daß* clause, this combination of features is only acquired during the syntactic derivation through adjunction of the matrix clause, which contributes the [+sat] feature. The integrity constraint in both cases prevents embedded arguments from scrambling into the matrix clause. Of course, the embedded arguments are free to scramble within the *Mittelfeld* of the embedded clause, as are any matrix arguments or adjuncts scramble within the *Mittelfeld* of the matrix clause.

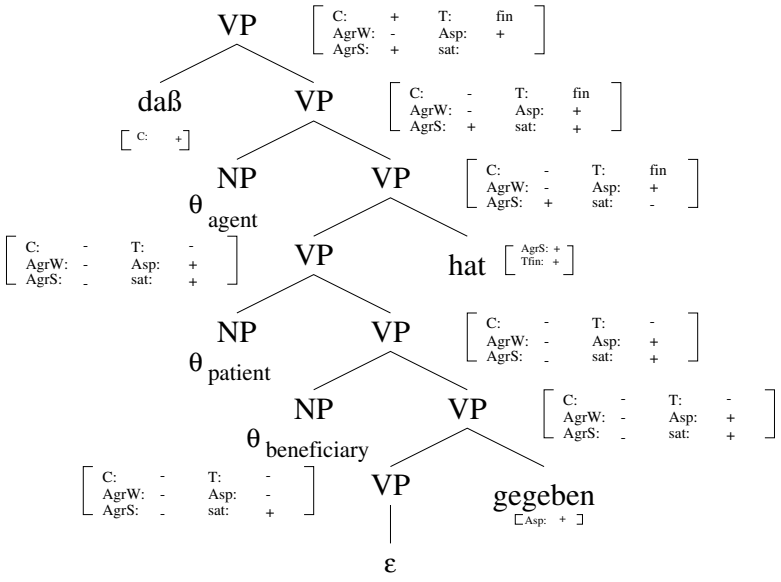


Figure 5.40: Finite embedded clause with overt complementizer

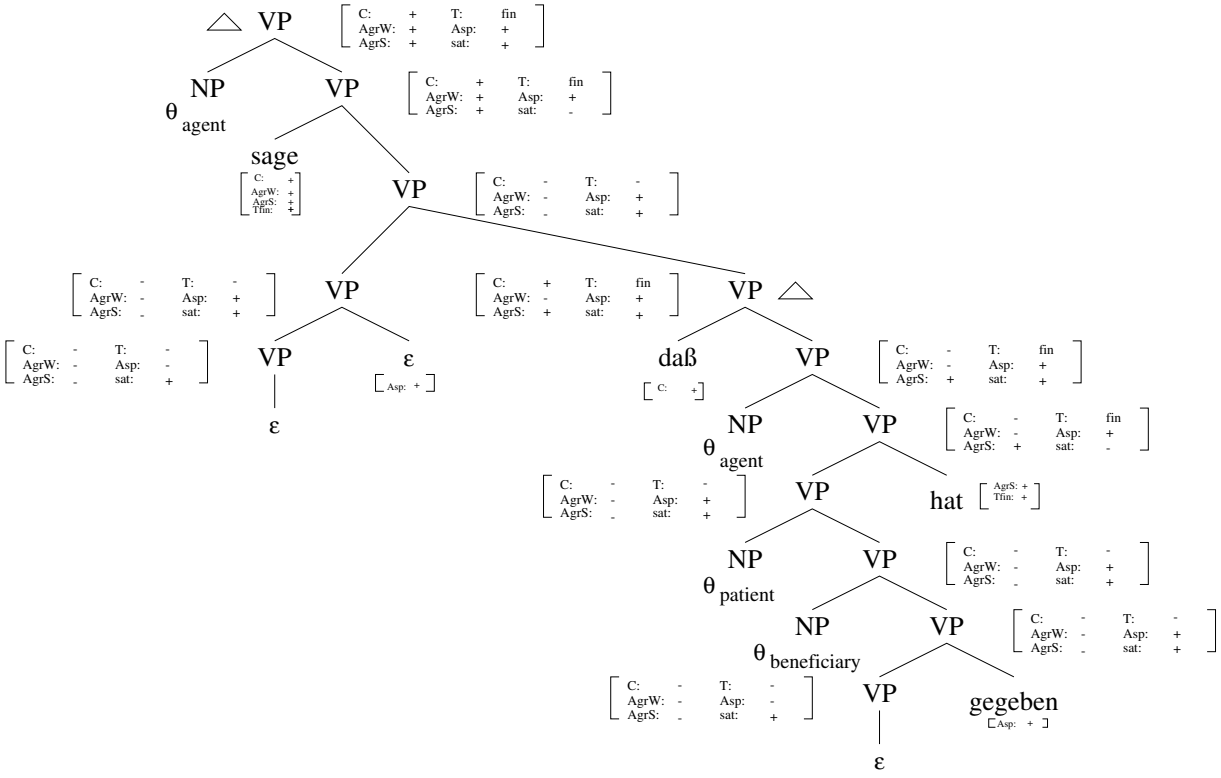


Figure 5.41: Derived sentence with embedded *daß* clause

5.4.9 Long-Distance Topicalization

As noted in Section 2.3.2, page 24, constituents can long-distance topicalize out of embedded clauses. This is possible without restrictions out of infinitival clauses. In certain dialects, long-

distance topicalization is also possible out of embedded *daß* clauses if the matrix verb is a bridge verb, perhaps subject to certain pragmatic or semantic conditions. Finally, long-distance topicalization is possible out of embedded V2 clauses, though the analysis is unclear, as discussed in Section 2.3.3, page 25. We will discuss the first two cases in turn, leaving the third aside.

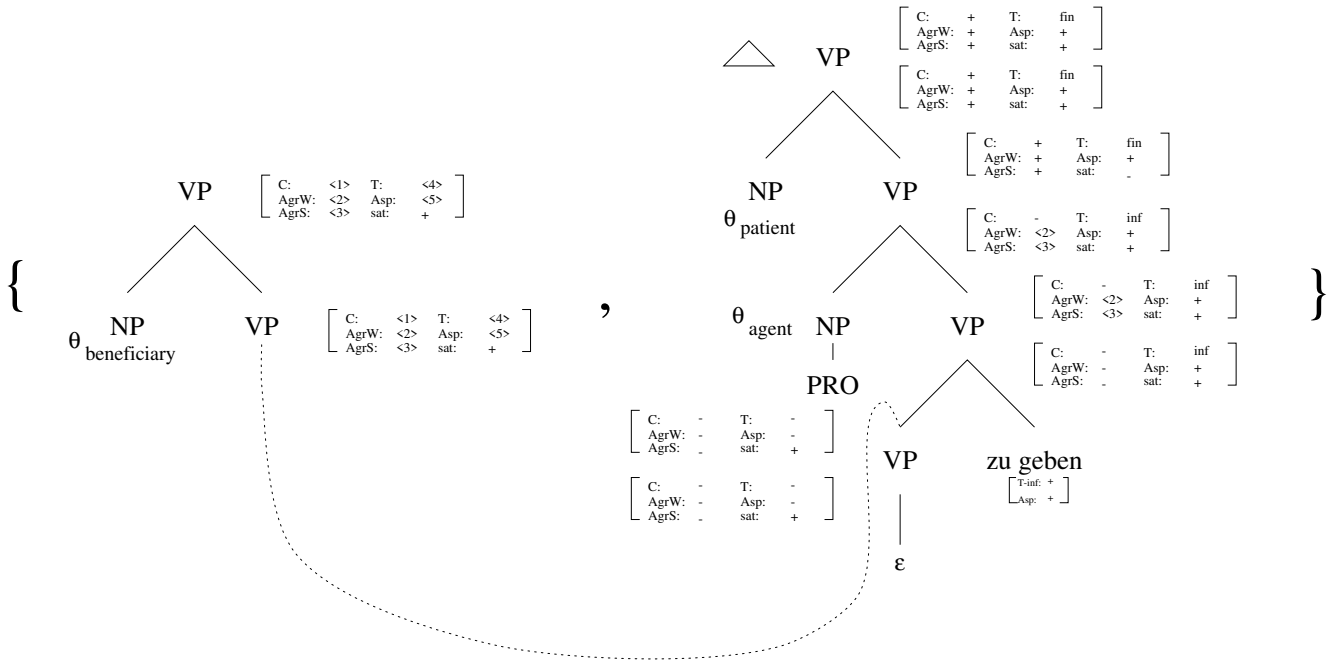


Figure 5.42: Syntactic tree set for long-distance topicalization

Let us start with long-distance topicalization out of embedded infinitival clauses, as shown in (30a) below (without long-distance scrambling) and in (35), page 24 (with long-distance scrambling), repeated here as (30b). Recall that sentence (30b) motivated the non-simultaneity in the definition of the V-TAG formalism.

(30) a. [Dieses Buch]$_i$ hat der Lehrer [PRO den Kindern t$_i$ zu geben] versucht
[this book]$_{\text{ACC}}$ has [the teacher]$_{\text{NOM}}$ [the children]$_{\text{DAT}}$ to give tried

This book the teacher has tried to give the children

b. [Dieses Buch]$_i$ hat [den Kindern]$_j$ bisher noch niemand [PRO t$_j$ t$_i$ zu geben]
[this book]$_{\text{ACC}}$ has [the children]$_{\text{DAT}}$ so far yet [no-one]$_{\text{NOM}}$ to give
versucht
tried

So far, no-one has tried to give this book to the children

The analyses of both types of long-distance topicalization have in common that the matrix clause is adjoined into a clause in which the element that will end up in the *Vorfeld* has already been fronted. Pursuant to our assumption that all semantic categorial features in the projection must be specified by the lexical derivation (lexical interpretation principle, page 151), this fronted

element must be marked [+C, +AgrW]. The syntactic tree for the embedded clause therefore must be as shown in Figure 5.42.

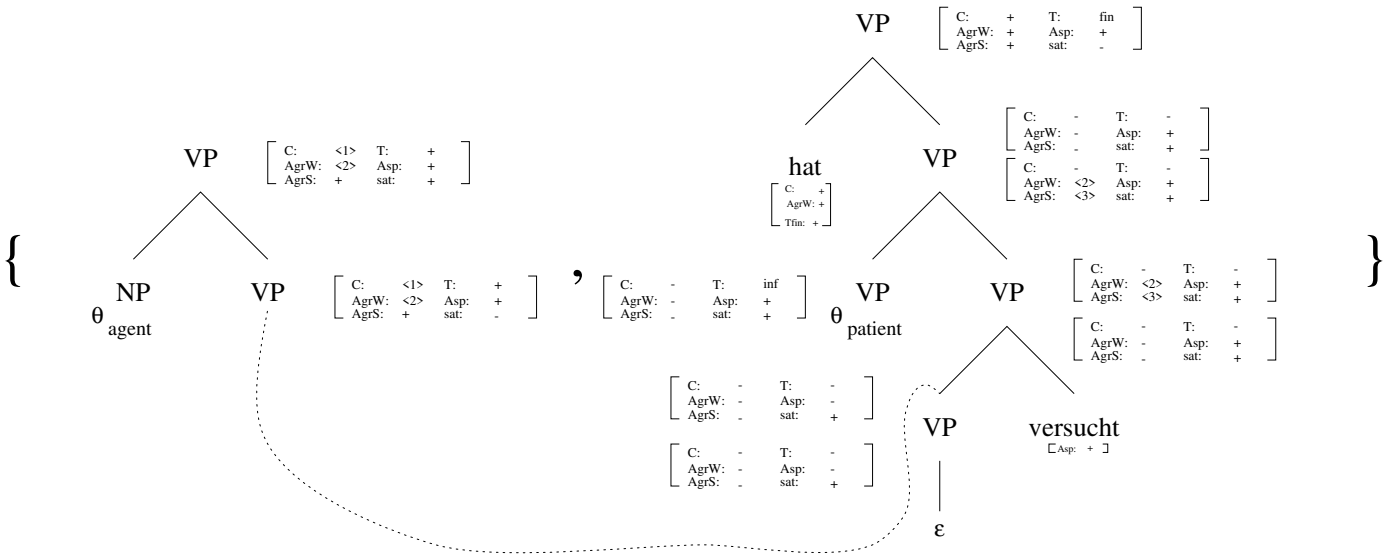


Figure 5.43: Syntactic tree set for matrix clause with long-distance topicalization

We observe that the root node has bottom features [+C, +sat], and therefore has the integrity constraint, preventing us from adjoining the $\theta_{\text{beneficiary}}$ to the root node of the projection tree. Furthermore, we observe that there is a feature clash on the second VP from the top: its top feature structure is specified [-sat] (from the [+Wh] θ_{patient}), while its bottom structure has [+sat] (the default for arguments). The structure that must be adjoined must have top features [+AgrW, +C, -sat], i.e., it must be an elementary tree lacking a topic, but with a verb in the V2 position. Furthermore, the tree must subcategorize for an infinitival clausal argument. Such a tree is essentially identical to the matrix clause shown in Figure 5.23, page 167, except that θ_{agent} has not been adjoined above the finite verb (auxiliary), but below. The tree is shown in Figure 5.43. When we adjoin the matrix tree at the VP node with the feature clash of the embedded tree, we resolve the feature clash. Two nominal arguments then still need to be adjoined. Since there are no integrity constraints between the non-finite verbs and the auxiliary, these can be adjoined anywhere in the MF, and we can choose either the order given in (30a), or the order in (30b). Figure 5.44 shows the embedded $\theta_{\text{beneficiary}}$ adjoined above the matrix θ_{agent} (we have chosen a matrix clause that extraposes its embedded clause, for reasons of representational ease). Note that this derivation crucially relies on non-simultaneous adjunction of the two trees in the embedded *geben* syntactic tree set.

Clearly, we can generate all the licit sentences, since there is no restriction on which argument is adjoined at the top of the syntactic *geben* tree, and we can also adjoin an adjunct there. Furthermore, all possible orders below the matrix finite verb in second position can be derived along the lines discussed above in Section 5.4.6. In order to see that we do not overgenerate, observe that the matrix clause must be adjoined at the second VP node in the embedded tree because of the feature clash there. (We cannot adjoin two "matrix" clauses because that would violate the θ–criterion.) Once adjoined, no element can be adjoined at the root of the sentence,

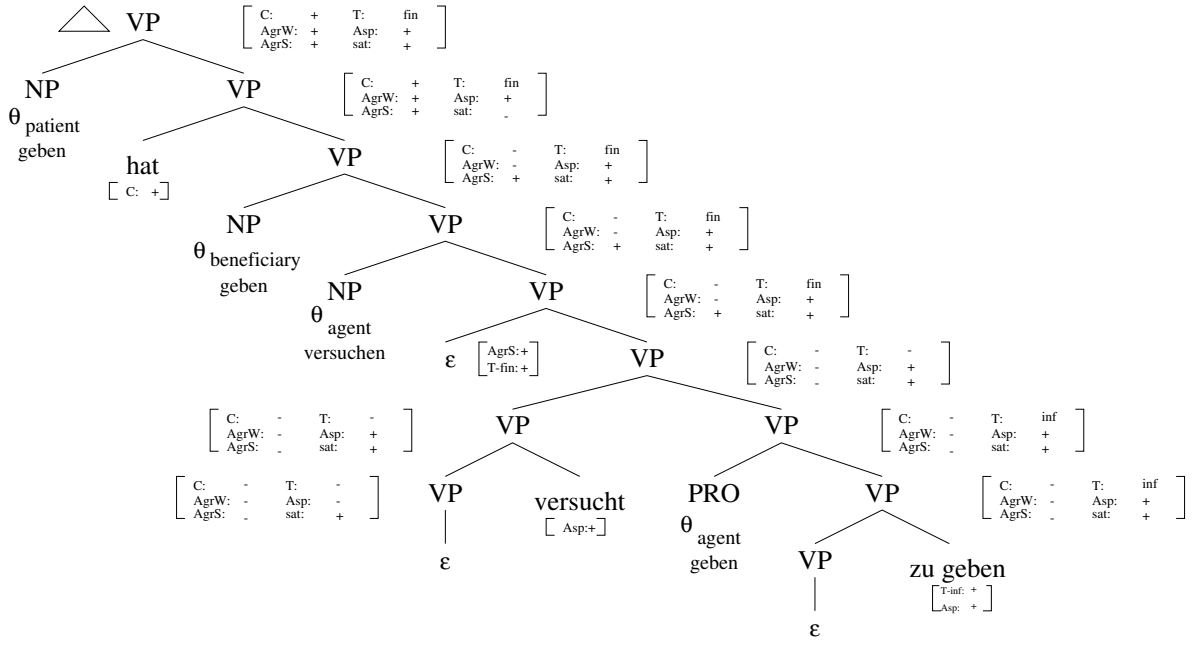


Figure 5.44: Derived sentence with center-embedded infinitival clause and long-distance scrambling

because of the integrity constraint, and none can be adjoined at the matrix [+C] head, since it would disrupt the required [+AgrW] spec-head agreement. We conclude that we can derive just the right set of structures.

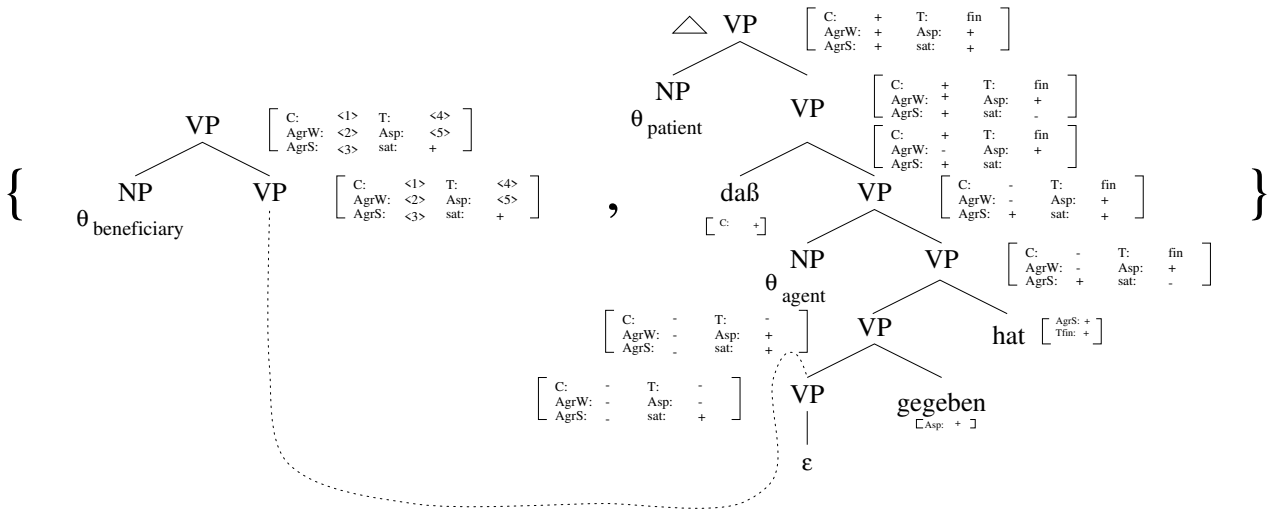


Figure 5.45: Syntactic tree for embedded clause with long-distance scrambling

Now let us turn to extraction out of *daß* clauses. Consider (36), page 24, repeated here as (31):

(31) Was glaubt Peter daß der Lehrer den Kindern gegeben hat?
what$_{\text{ACC}}$ thinks Peter that [the teacher]$_{\text{NOM}}$ [the children]$_{\text{DAT}}$ given has

What does Peter think that the teacher gave the children?

We will derive this sentence as follows. The initial tree includes a finite verb as a [+V-fin] head, an overt complementizer, and an argument (or adjunct) adjoined to the complementizer. Recall that since the complementizer does not participate in spec-head agreement, its [sat] feature is unsatisfied, and hence the VP node immediately dominating the complementizer does not have an integrity constraint. The element adjoined there, however, is marked [+C,+sat], and therefore does. The tree is shown in Figure 5.45. For the sake of readability, we have collapsed most feature structures except the one at the VP node immediately dominating *daß*. We see that there is a feature clash since *daß* does not contribute [+AgrW], while the topicalized element requires it. This forces us to adjoin a matrix clause. It is the essentially same structure that we adjoined in the previous case (Figure 5.43), except that the verb *glauben* 'to believe' takes a clausal complement marked [+V, +T-fin, +C, +sat], indicating subcategorization for a finite clause. Note that this feature combination implies the integrity constraint. The tree is shown in Figure 5.46. the result of the adjunction is shown in Figure 5.47.

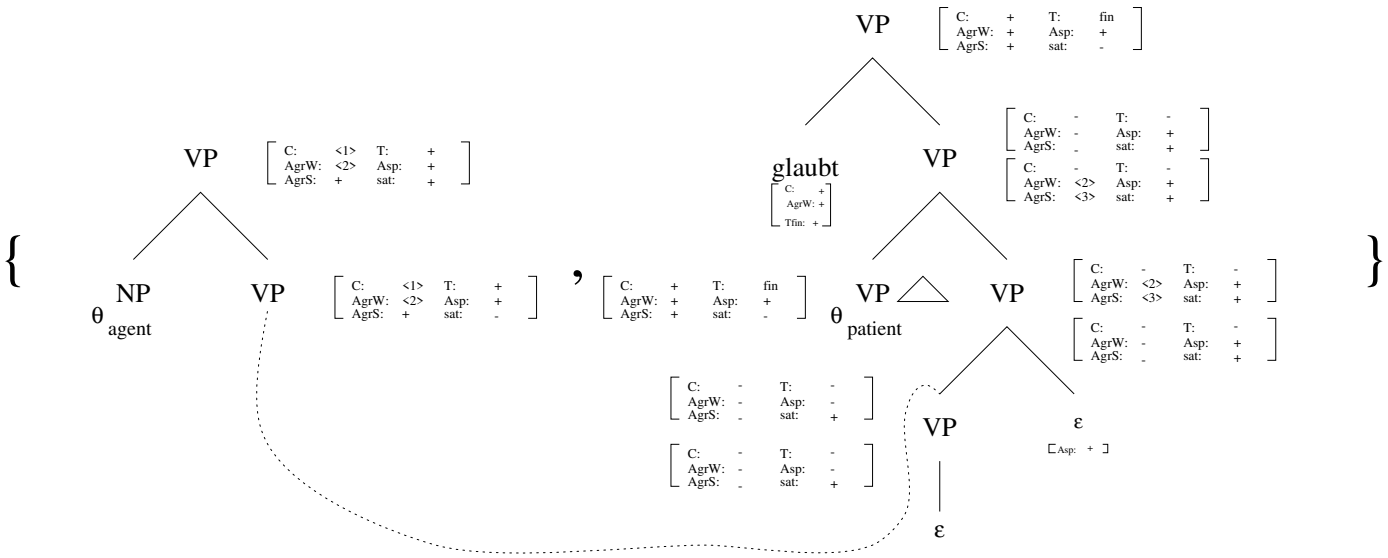


Figure 5.46: Syntactic tree set for matrix clause with long-distance topicalization

We will now argue that we can derive exactly the correct structures. Clearly, all licit structures can be derived. To see that no illicit structures can be derived, first observe that during the lexical derivation we cannot adjoin two elements (arguments or adjuncts) above the complementizer, since the lower one introduces an integrity constraint. During the syntactic derivation, we cannot adjoin an argument above the complementizer, either before of after adjoining the matrix clause, since the newly adjoined argument would introduce the [+sat] feature, and thus add the integrity constraint to the VP immediately dominating *daß*, making the adjoined argument illicit. Adjuncts cannot appear in that position during the syntactic derivation since adjuncts marked for [+C] also require [+AgrW], which the complementizer cannot provide. Thus, we conclude that exactly

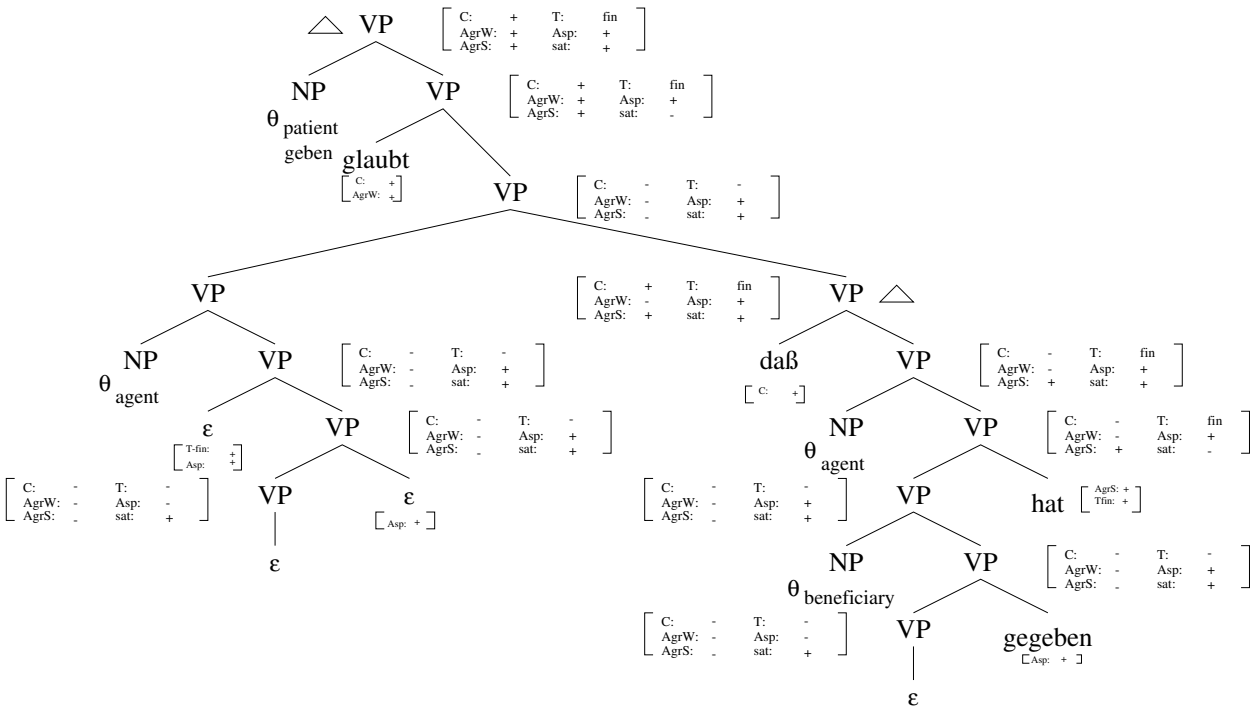


Figure 5.47: Derived sentence with extraposed finite clause *daß*-clause and long-distance topicalization

one element can topicalize, and no element can appear adjoined to the complementizer. The integrity constraint also prevents scrambling into the matrix clause, as we have seen previously. We conclude that the right set of structures is generated.

5.4.10 Topicalization and Scrambling: Two Distinct Types of Movement

In Section 2.4, page 30, we discussed several differences between scrambling on the one hand and topicalization and *wh*-movement on the other. Let us examine how these differences are accounted for in the proposed framework.

Let us start with differences relating to the freedom of movement. Since scrambling is implemented by multi-component adjunction, we predict it to be iterable, and that landing sites are not necessarily clause-peripheral. Topicalization, which is implemented by adjunction of the matrix clause, can only move a single string (though from the formal system we have no way of restricting this to be a single constituent), and the "landing site" must be clause-peripheral. Furthermore, the availability of a single adjunction site above a complementizer (or a verb in V2 position), implemented by the barrier on [+C,+sat] VPs, predicts that only a single constituent can topicalize. Furthermore, the same barrier prevents scrambling from leaving CPs, while the adjunction of the matrix clause provides an "escape hatch", which is of course similar to traditional analyses of successive-cyclic movement, but does not depend on a mechanism that alters the barrierhood of nodes dynamically. We conclude that we have adequately addressed the differences relating to

freedom of movement, either entirely through the choice of formal means, or through a single stipulation of barriers.

Let us now turn to the linguistic differences relating to binding facts. We do not propose to elaborate a theory of binding for V-TAG here. However, we can predict the observed differences if we correlate reconstruction effects with the lexical derivation phase, at the end of which the semantic properties of the derived lexical structure must be fixed. This stipulation precludes long-distance scrambled elements from showing reconstruction effects. If we furthermore assume that only arguments that participate in the semantic interpretation[30] of a clause *may* be adjoined during the lexical derivation (as opposed to *must* be adjoined, as we assumed in Section 5.3.5), then the differences follow.

We conclude by observing that this analysis avoids a problem that movement-based analyses encounter: why can scrambling not make use of the same escape hatch that long-distance topicalization uses, such as adjunction to the lower CP? For example, Müller and Sternefeld (1993) propose a "Principle of Unambiguous Binding", which restricts the positions in chains to be of the same type (except for the foot). In our analysis, the two types of movement cannot be "mixed", since after the lexical derivation, each argument is either topicalized or scrambled.

5.4.11 Coordination in German

In this section, we briefly review the argument put forward by Heycock and Kroch (1993) (HK, henceforth) in favor of multi-functional heads and projections. Their analysis is based on the so-called "SLF"-coordination in German, first discussed by Höhle (1983). In SLF-coordinations, the shared constituent is in SPEC(IP) in the first conjunct, and the second conjunct is a C′, as evidenced by the verb in clause-initial position. This is shown in (32), which is HK's (1b).

(32) Das Gepäck ließ er fallen und rannte zum Ausgang
[the baggage]$_{\text{ACC}}$ let he drop and ran to-the$_{\text{DAT}}$ exit

He dropped the baggage and ran to the exit

This construction is problematical if the uncontroversial assumptions are adopted that only like constituents can coordinate, and that the shared element must be outside the first constituent. HK's solution is based on a "dynamic" theory of licensing, in which licensing is not determined by specific structural positions, but by the relationship (spec-head agreement or government) between two elements in phrase structure, independent of their specific position. They then propose a "Principle of Minimal Satisfaction", according to which only the minimum structure needed for satisfying licensing relations in which a head participates is projected. This principle forces the deletion of the I-projection when the subject moves into SPEC(CP) and the finite verb into C_0, because the spec-head agreement of the I-projection is copied in the C-projection system, and therefore redundant. When the I-projection deletes, the C-projection takes on the functions of the I-projection, and is labeled C/I.

[30] By "semantic interpretation" we mean issues beyond the propositional content, of course.

The SLF coordination facts are then easily explained: the second conjunct is not only a C′, but also an I′, and therefore can coordinate with the I′ from the first conjunct. This leaves us with violations of the Across-The-Board constraint on extraction from coordinate structures. HK show that argument extraction into a specifier position is possible even in non-SLF coordination in German, thus suggesting that the ATB constraint does not apply unconditionally.

In the adjoined-head approach, we can immediately adopt HK's analysis. We simply need to propose a [+C, +I] head for German (or [+C, +T-fin], in our notation). Clearly, this head can only be used if the subject has topicalized, since otherwise the subject can never receive nominative case. If we then assume that heads can coordinate just in case they share a categorial feature, then the facts follow immediately, in a manner that essentially duplicates HK's analysis.

While the arguments that HK give for a projection that functions both as an I-projection and as a C-projection are compelling, the implementation of the "merged" projection in their system has some technical (or, rather, "aesthetic") difficulties. First, the notion of structure deletion may be criticized on conceptual grounds as an unwarranted complication of the machinery of the theory. Second, while HK label the "merged" projection "CP/IP", it is not entirely clear what the status of this label is in the theory. Is it merely a mnemonic to signify that the C-projection is now "exhibiting the properties" of the I-projection, or is it a different projection? If it is a new projection, then the deletion of the I-projection must be accompanied by a transformation of the C-projection, which seems strange. While the intuition is quite clear, the implementation of the intuition in the traditional GB framework is not. We would like to suggest that a precise formulation of the content of HK's analysis would necessarily lead to the implementation of the content of functional projections as features, as is done in the adjoined-head approach.

HK discuss several other issues that their approach handles. Not surprisingly, similar accounts can be given in the adjoined-head approach. Subject questions in English are discussed in (Rambow, 1993, Section 3), while constraints on the topic position in Yiddish are analyzed in Section 5.5.1.

5.5 Parametric Variation

In this section, we briefly touch on two Germanic languages closely related to German, Yiddish and Dutch, and suggest how the syntactic differences can be derived within the framework we are elaborating. For a discussion of English (and French), see (Rambow, 1994).

5.5.1 Yiddish

Yiddish is an SVO language which shows the V2 phenomenon. However, unlike German, Yiddish is also V2 in embedded clauses ("symmetrical V2 language"), as exemplified by (32).[31]

(33) a. Moische bedauert az zayn bukh hob ikh geleyent
Moische regrets that his book have I read

Moische regrets that I read his book

[31] (33) is taken from (Heycock and Santorini, 1992).

b. Ikh veys nit farvos in tsimer iz di ku geshtanen
I know not why in room is the cow stood

I don't know why the cow stood in the room

It has been proposed that this is because in Yiddish, the verb only moves to I_0, not to COMP as in German (Diesing, 1988; Santorini, 1989).[32] In the framework that we have proposed in this chapter, we can no longer refer to structurally defined positions, such as I_0 or SPEC(IP). Instead, we must analyze this phenomenon in terms of the semantically and morphologically justified set of functional categorial features. The obvious approach to pursue is to claim that in Yiddish (and other symmetrical V2 languages), the [+C] and [+AgrW] heads need not coincide, while in German (and other asymmetrical V2 languages) they must. (Recall that the [+AgrW] head either licenses a [+wh] constituent, or establishes the topic position through the adjunction of an empty [+wh] operator.) Then the complementizer *az* 'that' (or a *wh*-word introducing a subordinate clause) contributes only [+C] in Yiddish. The [+AgrW] feature is introduced by a separate head (the finite verb), as is the [+AgrS] which assigns Case to the subject. (If the subject immediately follows the complementizer, then there is a [+AgrS, +AgrW] head.)

Now consider a matrix *wh*-question, and a long-distance *wh*-question:

(34) a. Vemem zoln mir gebn ot di bikher?
whom$_{\text{DAT}}$ should we give *prt* the books

To whom should we give *these* books?

b. Vemem hot er nit gevolt zoln mir gebn ot di bikher?
whom$_{\text{DAT}}$ has he not wanted should we give *prt* the books

To whom did he not want us to give *these* books?

Let us assume that in questions, both matrix and embedded, verbal heads must have the [+C] feature.[33] Then in (34a), the auxiliary *zoln* 'should' is a [+AgrW, +C] head, while an empty [+AgrS] head assigns Case to the subject. For the case of long-distance questions, observe that (34b) corresponds to the German cases that we discussed in Section 2.3.3. There, we argued, contra Grewendorf (1988), that it is not obvious that the "matrix clauses" are not actually parentheticals. Presumably, the same question arises for Yiddish as well. Since examining this question goes far beyond the scope of this discussion, we will for the sake of argument assume that (34b) represents a true case of embedding, and that there is an empty [+wh] operator just in front of the auxiliary *zoln*. Then it is clear that our analysis of matrix questions transfers to long-distance questions straightforwardly.

[32] Heycock and Santorini (1992) present convincing evidence against a CP-recursion analysis.

[33] This assumption corresponds to the proposal in transformational frameworks that in questions, the verbal head raises to C_0. While this is not uncontroversial for the case of matrix questions in Yiddish (see (Diesing, 1990)), it is for the case of long-distance questions. As Heycock and Santorini (1992) argue, assuming that there is no difference between matrix and long-distance questions is desirable since it assimilates Yiddish questions to those of other symmetrical V2 languages such as Old English and Old French, for which there is independent evidence that the verb raises to C_0, and it lets us present a uniform analysis of the phenomenon under discussion here. Diesing, on the other hand, must present different analyses for the matrix and long-distance question cases.

This analysis makes an interesting prediction. Since, in order to license the [+wh] element to the left of the auxiliary, the verbal head must be [+AgrW], and since the Head Feature Principle prohibits two different heads from introducing the same feature in the same projection, no additional topic can be licensed below (to the right) of the auxiliary, and the auxiliary must immediately be followed by the subject. This prediction is borne out:

(35) a. * Vemem zoln ot di bikher mir gebn?
whom$_{\text{DAT}}$ should *prt* the books we give

Intended reading: To whom should we give *these* books?

b. * Vemem hot er nit gevolt zoln ot di bikher mir gebn?
whom$_{\text{DAT}}$ has he not wanted should *prt* the books we give

Intended reading: To whom did he not want us to give *these* books?

These facts follow from a straightforward analysis of Yiddish if we assume that heads can be multifunctional. These facts, however, are significantly more difficult to explain under a traditional transformational approach, in which phrase structure is given *a priori* and can only be changed by movement. The problem is that under a traditional analysis, after the verb has raised to C_0, there is still the I-projection below it, and an explanation needs to be found why SPEC(IP) is not available for *ot di bikher* in (35a) and (35b) above. Heycock and Santorini (1992) base their analysis on a general (unnamed) principle (their (10)):

(36) Nonthematic positions are licensed only at S-Structure

Since SPEC(IP) is a nonthematic position, it can only be licensed if the verbal head is in I_0 *at S-Structure*, which is not the case in (36a) and (36b). The subject remains in its base-generated VP-internal position, and receives Case from the I_0 head under government. A rather complex story is then needed to explain why English subjects cannot receive Case under government but need to raise to SPEC(IP). Heycock and Santorini's analysis is in fact rather similar to the one proposed here. Both rely crucially on the notion that nonthematic positions can only be licensed "once". In the V-grammar framework, this is a consequence of the Head Feature Principle: each categorial feature can only be introduced once, and each type of (nonthematic) licensing (*wh*-element, nominative Case, etc.) is associated with one feature. In Heycock and Santorini's approach, the additional principle given in (36) is needed. As a consequence of both analyses, nonthematic licensing and licensing through θ–assignment are handled through orthogonal mechanisms: in V-grammar, θ–role assignment (and no other kind of licensing) is handled through the grouping into sets, and in Heycock and Santorini's approach, θ–role assignment (and no other form of licensing) is handled at D-Structure. Interestingly, Heycock and Santorini conclude that principle (36) implies that specifiers should be adjoined (rather than substituted into existing phrase structure positions), since the the positions cannot be present at D-Structure.[34] This of course represents half of the proposal embodied by V-grammar. The other half, that the heads themselves are also adjoined, is what permits us to simplify Heycock and Santorini's proposal, since the possibility of multifunctional heads allows us to retain Case-assignment under spec-head

[34]This point is discussed in more detail in (Heycock, 1991).

agreement even for Yiddish, dispensing with the need for Case-assignment under Government and with the need to account for the theory-internal difference between English and Yiddish that Heycock and Santorini encounter.

The analysis we have proposed also addresses an unrelated problem with the IP-analysis of symmetrical V2 languages (Diesing, 1988; Santorini, 1989). Since the verb must raise to I_0, but since the subject need not raise to SPEC(IP), it is unclear how a morphosyntactic account can be found that is in accord with the mirror principle (Baker, 1985). Since the agreement morphology is outside the tense morphology on Yiddish verbs, it is unclear how agreement can happen after the verb has raised, but the subject has not. While Case assignment under Government may be a plausible, if cumbersome, option, agreement under Government is implausible.

5.5.2 Dutch

We now briefly hint at how the analysis of Dutch verb-raising proposed in the simple TAG framework (Kroch and Joshi, 1985; Kroch and Santorini, 1991) can be adapted to the V-grammar approach proposed here. The crucial element will be that we adopt the simple TAG notion of tree-internal raising, and therefore favor a "syntactic" analysis in the sense of Kroch and Santorini (1991) over a morphological analysis that actually involves verb raising into the projection of another head and subsequent incorporation. We will assume that verb raising is a lexical process, i.e., it happens to the lexical tree set after features have been assigned, and it only involves a single head. Since the elementary trees containing the heads now no longer contain any argument positions, the verb just raises one "level". The effect of verb raising is shown in Figure 5.48. Crucially, the feature structure associated with the root node of the head auxiliary tree before raising (Φ_2) remains associated with the *intermediate* VP node after raising. The new root node is syntactically "inert": no adjunctions can take place there, it contains no feature information.

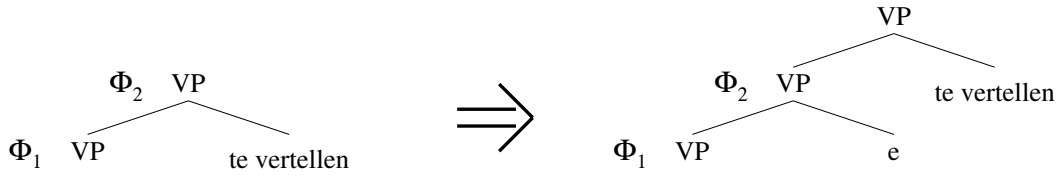


Figure 5.48: The Verb Raising operation

We now discuss derivation for Dutch sentence such as (37).

(37) omdat Jan Marie leugens verbood te vertellen
that Jan Marie lies forbade to tell

that Jan forbade Marie to tell lies

We will assume that both verbal heads undergo verb raising. After adjoining the arguments, we obtain the two trees shown in Figure 5.49. When we adjoin the matrix clause *verbood* tree into the embedded clause *te vertellen* tree at the VP node just below the root, we obtain the tree shown in Figure 5.50. (We omit the features, and also do not discuss the difference in scrambling behavior for Dutch and German.) Further examples of Dutch derivations can be found in Section 6.2.1.

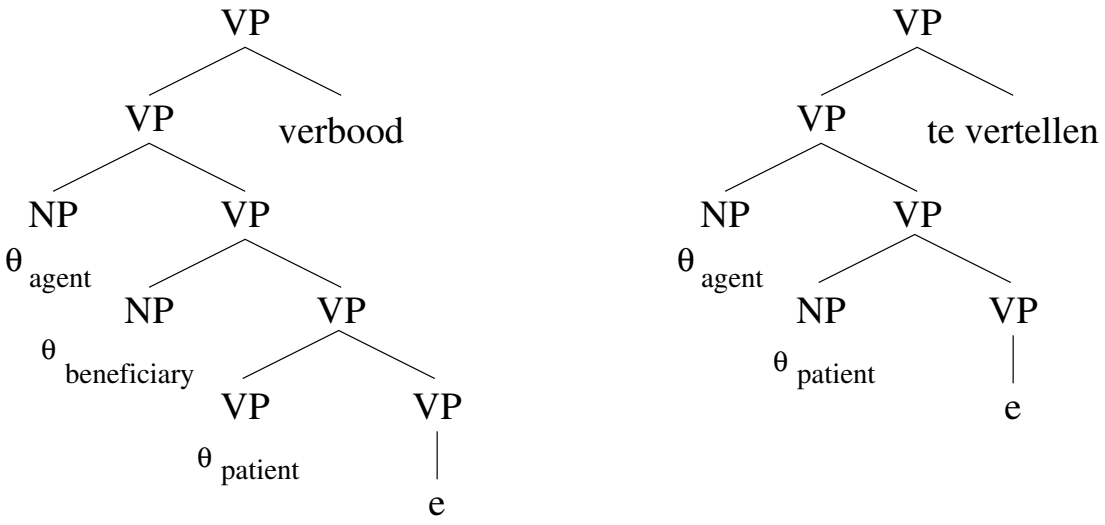


Figure 5.49: Derived Dutch clausal trees

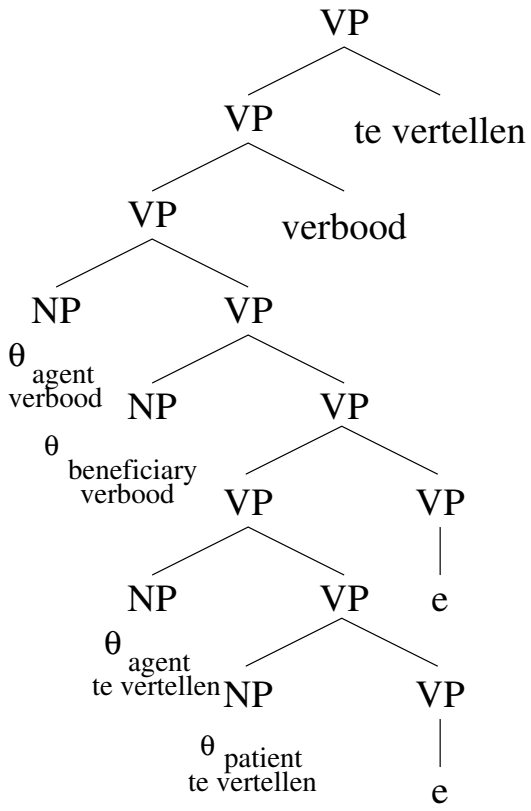


Figure 5.50: Final Dutch tree

Note that as in the case of German, we need not adjoin any nominal arguments to the projection during the lexical derivation. If we then start out the syntactic derivation by adjoining the matrix clause projection into the embedded projection, we get the structure shown in Figure 5.51, which,

as its German counterpart (Figure 5.28, page 171), appears to capture some of the intuition behind the "clause union" analysis of Evers (1975).

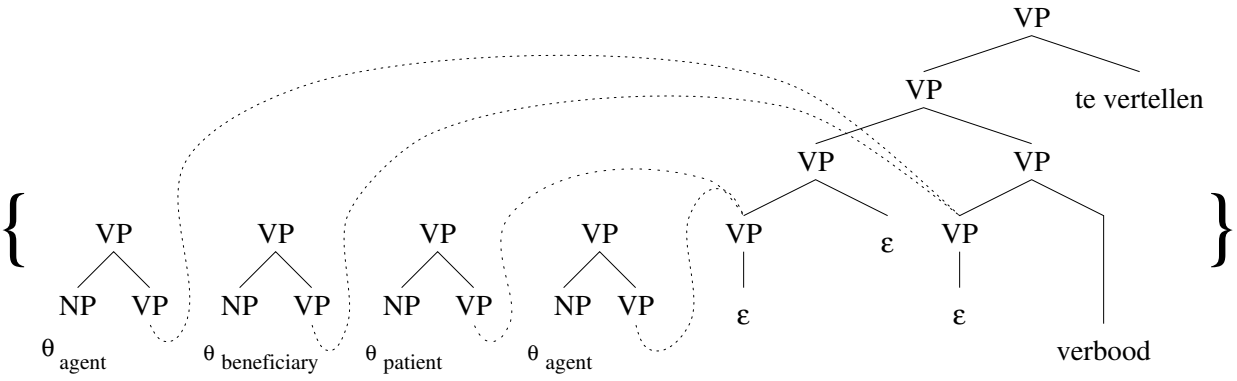


Figure 5.51: The "clause union" analysis in V-TAG for Dutch

5.6 Comparison to Other Approaches

In this section, we compare the linguistic approach outlined in this chapter to several other linguistic proposals. We start with two TAG-based linguistic theories. Frank's proposal represents the most complete exposition of a linguistic theory based on simple TAGs to date (Section 5.6.1). Hegarty uses the notion of incremental generation of phrase structure and thus his proposal bears many similarities to ours; we review it in Section 5.6.2. We then turn to non-TAG related accounts of free word-order phenomena, and discuss Karttunen in Section 5.6.3 and Uszkoreit's proposal for German in Section 5.6.4.

5.6.1 Frank (1992)

In the previous two sections, we have already seen how the proposal made in this thesis relates to the theory developed by Frank (1992). The two approaches share a key objective, namely the use of mathematically constrained formalisms for the expression of syntactic competence. Many similarities are directly or indirectly related to this underlying common approach. However, we have seen in Section 5.2.1 that Frank uses a non-formal derivational step to derive the syntactic structures, while we propose using a formal step. Frank explicitly argues against such an approach, which he flatteringly calls "a radical alternative" (Frank, 1992, Section 3.6). He advances three reasons against such an approach. Let us briefly discuss them in turn.

Frank first addresses the facts concerning the order of adverbs, verbs, and auxiliaries in English and French discussed by Emonds (1978) and subsequently by Pollock (1989). Frank argues that the possible analyses that compose the phrase structures incrementally cannot account for the facts properly. However, Frank has not considered all possible analyses. In Rambow (1994, Section 3), we address exactly this problem, and give an analysis in terms of V-grammar. Frank also claims that such an approach would mean that we have heads which are in separate trees

from their arguments. While this is undesirable in a pure TAG approach, we have extended the formalism to allow for sets of trees, so that this fact poses no theoretical problems.

Frank then turns to other cases of verb movement, in particular, VSO structures in Arabic. They do not appear to present a particular problem, if we suppose that Infl is split into T and AgrS heads, and if we suppose that the verb carries agreement morphology only if it is c-commanded by the AgrS head. We will, however, not work out the details. Frank also mentions Germanic I-to-C head movement as problematic for the "radical alternative". This issue has been discussed extensively in Section 5.4.

Finally, Frank cites the problem of floating quantifiers, which we address below in response to the analysis in (Hegarty, 1993b). This problem seems to be the most compelling of those mentioned by Frank: the problem for the V-grammar approach is not the mobility of heads, but the issue of traces for arguments. However, as we will argue, what is crucial in these cases is in fact not the argument trace, but the adjacent head.

In conclusion, we see that the introduction of a new formalism makes it possible to give a reinterpretation to the transformation notion of head movement (as "head mobility"), and thus allows us to reinterpret much recent work in GB theory (which relies increasingly on moving heads) in a non-transformational framework.

5.6.2 Hegarty (1993)

Hegarty (1993a; 1993b) (independently) pursues the same goals as this thesis: to eliminate a separate notion of a tree-sized elementary structure that is derived by a mechanism completely different and independent from that of TAG (be it a CFG with transformations, or some more linguistic notion of X′-schema and move-α). For essentially the same reasons (namely, the definition of the TAG formalism and the notion of adjunction), Hegarty is also led to assuming that the functional projections are not categorially distinct from their associated lexical category, but are realized as features. In both approaches, phrase structure is extended by adjunction in response to the need for specific functional projections, a need which is expressed as a feature mismatch.

However, there are certain differences between the two approaches, three of which are major and give the analyses their "flavor". The most salient is that while the adjoined-head approach uses multi-component TAG sets as the lexical representation, Hegarty uses trees. As argued in Section 5.1, the set-based approach allows for a language-independent, parametrized "pure" lexical representation, while the tree-based approach must include a theory of how θ-roles are mapped onto phrase structure. The problem becomes particularly obvious when considering ditransitive verbs. Under the adjoined-head approach, English and German lexical entries for ditransitive verbs look alike. Under the tree-based approach, a theory of double objects must be available for English, which, as the GB literature shows, is not at all a simple matter.

The second difference concerns the heads. In the adjoined-head approach, heads are adjoined during the derivation, and the effect of what is analyzed as head movement in standard GB analyses is achieved by the (optional) adjunction of empty heads. Hegarty (1993b, p.19), on the other hand, proposes to implement French head movement within the elementary verbal projection (i.e., the lexical representation). Presumably, this approach would extend to other

instances of head movement such as German verb-second. This appears to add a transformational step between the lexical representation and the adjunction-based derivation, thus making the structure of the linguistic system more complex. In the adjoined-head approach, the only possible derivation is (TAG-defined) adjunction. Furthermore, in Hegarty's analysis, the verb does not end up in the I projection, since the I-projection is adjoined separately (just below the moved verb). In the adjoined-head approach, the verb (as a head) carries the features it has in the standard GB analysis, i.e., it coincides with the I^0 node (or, in the case of German V2, the C^0 node). As a consequence of the different treatment of heads, the adjoined-head approach is fully lexicalized (except for the empty heads, which, however, are specifically licensed by lexical items), while Hegarty's approach is merely "featurized" (Hegarty, 1993b, p.25), meaning that every tree introduces new features, but not necessarily a new lexical item. It is not clear whether this difference is relevant, e.g. for parsing or processing models.

The third major difference is that Hegarty does not allow for the possibility of merging functional projections (at least not as described in (Hegarty, 1993b)), since the adjoined functional projections actually contain nodes labeled I or C (though they project to VP nodes). Thus, the analysis of English proposed in (Rambow, 1994), according to which tensed lexical verbs in English introduce categorial features [+V,+T-fin], is not available to Hegarty, and a formulation of the analysis of Heycock and Kroch (1993) of German conjunction, given his assumptions, would have to mirror their notion of projection deletion explicitly.

There are some smaller differences worth mentioning:

- Hegarty proposes to force adjunction of the I-projection through a [+NOM] feature (p.8). In the V-grammar approach outlined here, adjunction of the AgrS and T features is required by the language-specific Sentential Head Feature Parameter, which supposedly is motivated by the semantic function of the [+AgrS] head. The case-assigning/checking function of the [+AgrS] head acts as a constraint on word order, not as a forcing factor in adjunction.

- Hegarty's system leaves traces for subjects within the VP. These traces are supplied by the adjoined I projection, and are motivated by quantifier stranding facts in English and French. Under the adjoined-head approach, no traces are left, which on the face of it would pose a problem for the analysis of stranded quantifiers. While this issue is not discussed in this thesis, it seems possible to link stranded quantifiers to specific heads, as does Hegarty's (57). This approach (as opposed to an approach in which the quantifiers are linked to independently established underlying positions for arguments) is also supported by the fact that in the presence of multiple auxiliaries in English, the quantifier can occur immediately to the right of any of them (Hegarty, 1993b, fn.11).

- Hegarty assumes that the [+C] feature, called [+assert] and given the same semantic interpretation, is obligatory in English. It is not clear whether there is empirical evidence for or against such a feature in English.

- Hegarty uses a [±root] feature (introduced by adjunction of the complementizer) in order to insure that clauses with complementizers receive matrix clauses. The V-grammar approach uses the independently motivated feature AgrW for much the same purpose.

- The irrealis construction is analyzed differently. Hegarty (1993b, p.29) uses a feature [+IR], while we propose to use an empty [+I] head (see Section 5.3.2, page 143). It is not clear whether these are mutually exclusive, or just compatible approaches to different aspects of the problem.

5.6.3 Karttunen's "Radical Lexicalism"

Karttunen (1989) suggests a categorial analysis of free word order (in Finnish) which differs in important respects from that given by Hoffman (1992) and discussed in Section 4.7.1. While Hoffman follows the traditional categorial approach and uses complex categories to express subcategorization requirements of (semantic) functors, Karttunen uses complex categories to express *dependence* in the sense of dependency theory (Tesnière, 1959; Mel'čuk, 1988). Therefore, arguments and adjuncts of a verb all have the same category, namely that of a (non-directional) functor from V to V, with V being a primitive category. The only composition operation is function application. In order to verify subcategorization, Karttunen embeds the categorial system in a unification-based framework, and assigns syntactic functions to noun phrases based on their case. These syntactic functions unify with subcategorization requirements of the verb when the noun phrase functor is applied to the verb.

Local word-order variation (scrambling) is easily expressed because neither the order of function application nor the order of unifying syntactic functions need be fixed. If word order needs to be constrained, as in the case of certain auxiliaries in Finnish which must occur before the verb, directionality parameters can be set in the definition of the functor. The case is more complex for long-distance scrambling. As we have seen Section 2.2.1, page 13, Karttunen also shows that Finnish allows for rather free long-distance word order variation. The problem here is that arguments of an embedded verb that have scrambled past the matrix verb *must* apply to the matrix subject, since function composition is not available, but at the same time, *cannot* apply to the subject since the relevant grammatical function of the matrix verb is filled by an element from the matrix clause. Karttunen seriously considers two related mechanisms. In the first, the category of the long-distance scrambled argument is changed, so that it is not (for example) the object, but the object of the embedded clause of the verb to which it will apply. In a sense this corresponds to type raising in a traditional categorial framework, except that the it is not the actual category that is affected (the noun phrase remains a functor from V to V). The disadvantage of this approach is that for arguments moved beyond two clause boundaries, the type changing operation must be invoked again, requiring mechanisms in an implementation that control the type changing operation. While this is possible to implement, the resulting system is not very elegant. Instead, Karttunen proposes to adopt a mechanism originally proposed for LFG by Kaplan and Zaenen (1989), *functional uncertainty*. Under this approach, noun phrases are not assigned a simple function, but rather a set of functions. Kaplan and Zaenen argue that allowable sets can be represented by regular expressions. For example, in the case of scrambling, we could represent the set of allowable functions of a scrambled noun phrase as "the object (of the embedded clause)*", where '*' is the usual Kleene star, meaning that the string enclosed in parentheses may occur n times, $n \geq 0$.[35] This mechanism allows for the unconstrained scrambling

[35] Of course, both Kaplan and Zaenen (1989) and Karttunen (1989) actually use a formal notation, whose details

that Finnish (and German) allow.

Karttunen's approach, motivated by essentially the same linguistic phenomenon (the "double unboundedness" of scrambling – see page 15), resembles the approach proposed here in crucial respects. Most importantly, subcategorization information and categorial information[36] is handled separately. In both approaches, there is only one verbal category (V or VP), and arguments and adjuncts combine with this verbal category without modifying the category label (through function application in Karttunen's system, through adjunction here). Karttunen proposes a separate set of features to check functional information, while in the approach outlined in this chapter, subcategorization information is represented by the grouping into sets. Furthermore, the structural relation between arguments and their governing verb is left "uncertain": in Karttunen's approach, through the use of LFG-style functional uncertainty, in this approach through the use of the dominance links, which leave unspecified the distance over which they will be stretched in a derivation. (Note that the notion of dominance is also included in functional uncertainty, since the regular expression express "downward" paths in trees.) The crucial difference in the treatment of long-distance dependencies is that in Karttunen's system (and functional uncertainty in general), the *licit* paths are specified, while in the V-TAG approach outlined here, the *illicit* paths are specified (through the use of the integrity constraint). It is not immediately clear whether these two devices give us exactly the same expressive power (it seems that the functional uncertainty approach is more expressive), but it is clear that they can be used to express the linguistically relevant constraints on movement (islandhood of adjuncts, and of overt complementizer-headed clauses in German). In summary, the two linguistic proposals in question, Karttunen's "radical lexicalism" and V-grammar, show significant similarities. This is not surprising, given that both analyses are motivated by the same type of data.

Since Karttunen's proposal is embedded in a unification-based framework, it is not *a priori* formally constrained. It may be possible to define a reduced version of full typed feature structures which would be powerful enough to express Karttunen's proposal, but which would be formally constrained (see for example the proposal to constrain HPSG in Section 4.7.3, page 126). However, it is not immediately clear how to express functional uncertainty so that the resulting unification-based formalism can be shown to be formally restricted. We leave to future research a closer investigation of the formal matters involved.

5.6.4 Uszkoreit's GPSG-Based Approach

Uszkoreit (1987) presents an important fragment of German grammar in the Generalized Phrase Structure Theory (GPSG) framework of Gazdar et al. (1985). GPSG is based on the claim that natural language is context-free.[37] Various extensions to CFGs, including LP rules that operate on sister nodes and metarules (appropriately constrained) extend the expressive power of CFGs, but not their weak generative capacity. The case of embedded infinitival clauses, which give rise to long-distance scrambling in German, are not part of the data set covered by the grammar.

need not interest us here.

[36] In the case of the approach proposed here the "categorial information" includes the categorial features.

[37] Uszkoreit does not commit himself to the context-freeness of natural language, and acknowledges the result of Shieber (1985).

Since it is exactly this case which, we have argued in Section 3.4.2, surpasses the derivational generative capacity of any LCFRS, including that of a CFG, the fact that Uszkoreit's grammar is expressed in a context-free formalism is not at odds with any claim made in this thesis. However, the results about the formal power required for scrambling suggest that the grammar could not be easily adapted in order to handle long-distance scrambling.[38]

Uszkoreit's grammar integrates different types of constraints on word order in the *Mittelfeld*; in this respect, it appears to be unique among grammars of German. By using newly defined disjunctive LP rules, different types of factors (pronominalization, case, pragmatic status) constrain word order, without any one factor necessarily determining it. This approach is based on the work of Lenerz (1977) and Höhle (1982), who show that different pragmatic contexts license different word orders, but certain word orders are licensed in more contexts than others. It is this latter point that Höhle takes as justification for the claim that, as is standardly assumed, the word order SUBJ-IOBJ-OBJ is in fact the default order. In the system presented in this chapter, there is no way of encoding this information. Even the additional use of AgrX heads for the two objects will not immediately allow us to encode the notion of default word order, since having AgrS C-select for, say, AgrO will of course prevent scrambling past the subject, an undesired effect. An option may be to introduce something like disjunctive LP rules into the C-selection hierarchy. This issue is closely related to the issue of processing; we return to it briefly in Section 6.5, but it will require further study.

[38] Uszkoreit mentions the fact that GSPG does not appear to provide a way of gradually increasing the formal power without immediately gaining full Turing power (p.44).

Chapter 6

A Processing Account

The processing model that we present in this chapter is based on (Joshi, 1990).[1] Joshi (1990) proposes to model human sentence processing with a formal automaton as defined in theoretical computer science, namely the embedded pushdown automaton (EPDA). The EPDA is equivalent to tree adjoining grammar (TAG), in the sense that for every TAG there is an EPDA that accepts exactly the set of strings that the TAG generates, and for every EPDA, there is a TAG that generates exactly the set of strings that the EPDA accepts. In this chapter, we extend the account of Joshi (1990) in two ways. First, we use the bottom-up variant of the EPDA, called BEPDA, which we introduced in Section 4.5, and which is also equivalent to TAG. [2] Second, we use the extension to BEPDA we defined in Section 4.6.4, the {}-BEPDA, which is equivalent to V-TAG. The formal equivalence between automaton and grammar formalism is crucial to our point: TAG has been used for the representation of competence syntax, and we propose that through the formal equivalence we can relate formal models of competence directly to formal models of performance in a motivated manner.

This chapter is structured as follows. We introduce the notion of using a formal automaton in Section 6.1. We recast the analysis of Joshi (1990) in terms of the BEPDA in Section 6.2. Section 6.3 extends this analysis to long-distance scrambling and extraposition. In Section 6.4 we consider long-distance topicalization. We finish with a brief discussion of the main characteristics of this model in Section 6.5.

6.1 Linguistic TAGs and associated BEPDAs

The formal equivalence between TAG and BEPDA means that for every tree adjoining language L, there is at least one BEPDA M that recognizes exactly L; in fact, there is an infinite number of such automata. The existence of *some* automaton that makes psycholinguistically relevant predictions is not of interest unless we know how to choose the right automaton from among

[1] This chapter represents joint work with Aravind Joshi.

[2] We would like to thank Yves Schabes for suggesting the use of the BEPDA (rather than the EPDA) in modeling human sentence processing.

those that are formally equivalent (but not all of which make the right predictions). The key proposal of this chapter is that the "easy method" introduced in Section 4.5.2, page 100 (for TAG) and Section 4.6.4, page 115 (for V-TAG) will derive the right automaton. Specifically, from a linguistically motivated grammar it will derive an automaton that models syntactic processing in a plausible way. This approach shifts the problem of providing a principled account of how to construct models of the syntactic processor (i.e., automata) to the problem of how to construct (competence) grammars – which is of course the object of linguistics. This point is important because it affects the question of the universality of the parser. The processor behaves differently for different languages. If we want to explain this variation in a principles-and-parameter type methodology, we have two options: either we can assume that the processor is parametrized on its own (though the parameter setting may be linked to the setting of the linguistic parameters), or we can deduce the cross-linguistic variation among processors from cross-linguistic variation among competence grammars and the way in which the (universal) processor interacts with the competence grammar. The latter view is adopted by Inoue and Fodor (1993), who term it "as-if-parametrized". Our approach falls into this paradigm as well.

6.2 A Syntactic Automaton

In this section, we turn to linguistic grammars and show how the easy method derives automata. We will start out with simple TAGs. We first give examples of how processing models are derived from competence grammars, and then we show how the automaton can be equipped with a metric to predict processing load.

6.2.1 Dutch Cross-Serial Dependencies and German Nested Dependencies

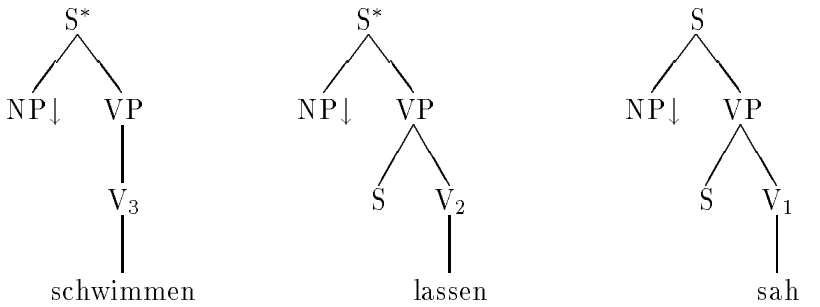


Figure 6.1: The German grammar

Let us consider the grammar for a fragment of German given in Figure 6.1. In this grammar, matrix clauses are adjoined into their subordinate clauses at the root S node. This analysis is motivated by facts about *wh*-extraction out of subordinate clauses as discussed by Kroch (1987; 1989): if we assume that the *wh*-word is included in the same tree as its governing verb (at an S′ node) and adjoin the matrix clause at the S node, then we get subjacency effects "for free".

(Further cross-linguistic evidence for this analysis of clausal embedding comes from Dutch. Kroch and Santorini (1991) give extensive syntactic evidence for the grammar given in Figure 6.3. Again, the correct derivation of the cross-serial dependencies relies on the adjunction of the matrix clause into its subordinate.) We therefore adopt this approach. Note that it is motivated by purely linguistic considerations – no processing issues intervened in the formulation of this grammar. The grammar in Figure 6.1 can generate center-embedded sentences such as the following:

(1) ... daß Peter Maria die Kinder schwimmen lassen sah
... that Peter Maria the children (ACC) swim (inf) let (inf) saw
N_1 N_2 N_3 V_3 V_2 V_1

...that Peter saw Maria let the children swim

German sentence (1) is derived by first substituting the nominal arguments into the NP substitution nodes of trees I, A, and B. Then, auxiliary tree A is adjoined into initial tree I at the root node of I, and auxiliary tree B is adjoined into the root node of the derived tree (which is in fact the root node of auxiliary tree A). The resulting structure, the derived tree, is shown in Figure 6.2.

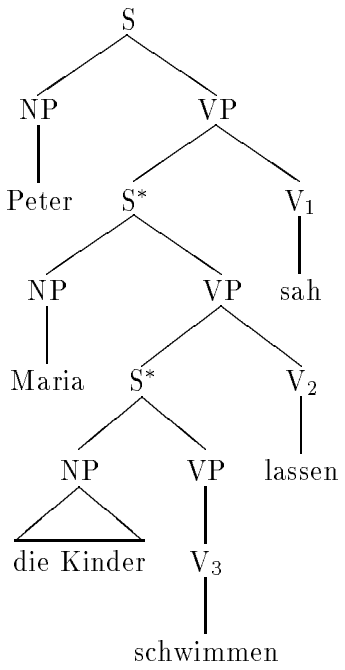


Figure 6.2: German derived tree

We now construct an automaton using the easy method. We start out by "exploding" the trees into context-free rules. For Tree I, we obtain the following context-free rules:

(2) Derived context-free rules for Tree I in Figure 6.1:

$S_I \longrightarrow NP_I\ VP_I$
$VP_I \longrightarrow V_I$
$V_I \longrightarrow$ *schwimmen*

As we have stressed during the discussion of the TAG formalism, the elementary structures of a TAG are trees, and therefore we can associate every elementary structure with a single lexical item ("lexicalization"). Since the automaton mimics the derivation in a TAG, we can interpret the moves of the BEPDA as establishing connections between lexical items. This view of the processor – as manipulating lexical items, and not phrase structure nodes – is somewhat different from the view on processing that arises when one starts from context-free grammars. In order to emphasize the lexical orientation of the BEPDA model, we will follow Joshi (1990) in giving the stack symbols quasi-categorial labels, rather than the standard labels. Suppose we have a clausal tree (anchored on a verb) that has frontier nonterminal nodes $\alpha_1, \ldots, \alpha_n$. Recall that frontier nonterminal nodes are either substitution nodes or the footnode. In either case, the tree can only participate in a terminating derivation if some operation (substitution or adjunction) is performed that adds structure below these frontier nonterminals. Thus, linguistically, frontier nonterminals correspond to arguments. Substitution nodes are nominal (or occasionally clausal) arguments, while footnodes are always clausal arguments. We will associate with the verb and each of the nodes in its projection (V, VP and S in this case) a label of the form $V\{\alpha_1, \ldots, \alpha_k\}$, where the α_i are labels of nonterminal nodes on the frontier (substitution nodes or the footnode) that have not yet been filled. Thus, linguistically, the list represents unfulfilled subcategorization requirements. It is important to note that this notation is entirely equivalent to the notation used in writing the trees themselves: in our sample trees, the VP corresponds to V{N},[3] and the top S node to V{}. If the verb subcategorizes for a clause, we write V{V}. The rules of the BEPDA that we obtain are as follows.[4]

(3) BEPDA rules derived from Tree I in Figure 6.1:

1. **Terminal Shift** *die Kinder* and *schwimmen* onto the pushdown store
2. **Reduce** *schwimmen* to $V_I\{NP\}$
3. **Reduce** [NP , $V_I\{NP\}$ to $V_I\{\}$

The first rule in (3) implements the principle of the easy method to push any scanned input symbols onto the pushdown store. Rule 2 represents the projection of *schwimmen* to the VP and corresponds to the second and third context-free rules from (2). In the third rule, the subject and the VP are combined to form the sentence, corresponding to the first context-free rule of (2). (Recall that the rule means that the stack [NP is removed from immediately below the top stack, and the symbol $V_I\{NP\}$ is popped from the top stack, and replaced by the symbol $V_I\{\}$.) Note that we assume that all NP symbols (both stack symbols and those in the subcategorization set) are marked with case information. Furthermore, we assume that we have a syntax of NPs that

[3]The notation should of course be understood to mean that the verb subcategorizes for a projection of a noun, not a bare noun. A similar remark applies to the notation for clausal subcategorization.

[4]Note that for the sake of readability, we will not split the nodes of the trees into two parts (top and bottom), as is actually required by the easy method. Therefore, we have omitted all **noadjoin** moves.

will allow us to recognize full NPs. Specifically, this syntax will allow us to **reduce** (in several moves) *die Kinder* to NP.

While we have described the initial tree I, we have not yet taken into account the possibility of adjoining. In our example tree, we can adjoin tree A or tree B at the root node. (There are no other possible adjunctions.) We express this by adding the following rule:

(4) BEPDA rules derived from Tree I in Figure 6.1 (ctd):

4. **Adjoin** tree A or B at the root node

Let us look at tree A as an example of an auxiliary tree. The rules derived from tree A are very similar to those derived from tree I, except for a new rule (v), which accounts for the fact that A is an auxiliary tree.

(5) BEPDA rules derived from Tree B in Figure 6.1:

1. **Terminal Shift** *die Kinder* and *schwimmen* onto the pushdown store
2. **Reduce** *sah* to $V_B\{NP\}$
3. **Reduce** [NP , $V_B\{NP\}$ to $V_B\{\}$
4. **Adjoin** tree A or B at the root node
5. **Unadjoin** the root node

The BEPDA is defined as a non-deterministic automaton. This is not appealing as a model of human sentence processing. The most salient instance of non-determinism involves the **adjoin** move. The automaton must nondeterministically guess that a particular tree is adjoined at a given node prior to performing any other operations relating to that tree. We will therefore assume that the automaton **adjoins** only if it already knows that it can perform a subsequent **reduce** move in the adjoined tree. We will combine the two moves into a **adjoin-reduce** move. In our example grammar, the sister nodes to the foot nodes in trees A and B are the verbs. We will therefore assume that the automaton pushes the footnode of tree A or B if it can immediately subsequently **reduce** the verb, which it can predict by scanning the input symbol (i.e., with lookahead one), and we will represent this as an **adjoin-reduce** move of V. Observe that, contrary to the **adjoin** move, the complex **adjoin-reduce** move is specifically licensed by the involvement of an overt lexical item. We will generalize this method of avoiding nondeterminism into the following principle:

(6) **Lexical Determinism Principle:** the syntactic automaton groups a sequence of moves into a single complex move if the sequence of moves must be performed in immediate sequence and if only one of the moves involves an overt lexical item.

The lexical determinism principle is, in a sense, a technical manifestation (at the level of the model we are discussing here) of the determinism hypothesis of Marcus (1980): the non-availability of mechanisms such as backtracking or (pseudo-) parallelism in the model means that moves (such as **adjoin**) that are not immediately warranted by observable evidence cannot be made. Since the available evidence, both from the input string and the pushdown store, is lexical, we can say

that the lexical items must determine the actions of the automaton. We return below to the issue of the orientation of the model towards lexical items as opposed to phrase structure. Observe also that the lexical determinism principle is reminiscent of the concept of closing a state in an LR(0) construction under a certain set of operations.

This still leaves open the question of how to order the rules that can all apply at the same time, given a certain input symbol and push-down store configuration. We observe that the notion of incremental processing means that the syntactic processor performs as much computation as it can on a given input token, rather than wait for the complete sentence before processing initial parts of it. We will therefore assume the following ordering principle on the application of BEPDA rules:

(7) **Ordering Principle for BEPDA rules:**

1. Perform all possible **reduce**, ε-**reduce** and **adjoin-reduce** moves first.
2. Then perform all **unadjoin** moves.
3. **Terminal-shift** a new input item only when no other moves are possible.

This of course does not address the problem that arises when two moves of the same category (e.g., **reduce**) are possible, which represents cases of true syntactic ambiguity such as PP-attachment. Such syntactic ambiguity does not arise in the cases we are interested in in this paper, and the automaton does not immediately make predictions about preferences. We return briefly to the issue of syntactic ambiguity in Section 6.5.

As an example, suppose we want to use the BEPDA to recognize the sentence fragment *die Kinder schwimmen*. The derivation involves no adjunction. The run of the automaton would be as follows:

(8)

Move	Rule Applied	Pushdown Store Configuration
1	**terminal-shift** *die Kinder* (Rule 1)	[*die Kinder*
2	Use NP BEPDA rules to derive	[NP
3	**terminal-shift** *schwimmen* (Rule 1)	[NP [*schwimmen*
4	**reduce** *schwimmen* (Rule 2)	[NP [V_I{NP}
5	**reduce** [NP , V_I{NP} (Rule 3)	[V_I

Recall that root nodes of initial trees are the final stack symbols of the grammar. Therefore, the store after move 5 signals successful recognition of a clausal unit. Since the rules derived from all three trees in the grammar in Figure 6.1 are structurally similar, we can group them together into the following table:

(9) **Automaton for German:**

Rule	Read Head	Top of Stack is	Action
1	N	*anything*	**terminal-shift** N
2	V	*anything*	**terminal-shift** V
3	*anything*	[N [V{N, ...}	**reduce** [N , V{N, ...} to V{...}
4	*anything*	[V [V′{V, ...}	**adjoin-reduce** V, [V′{V, ...} to VV′{ ...}
5	*anything*	[...V{}	**unadjoin** V{}

In the third rule, V{N, ...} has an unfulfilled subcategorization requirement for a noun (and perhaps other elements, as indicated by the dots, in a notation adapted from the categorial approach of Hoffman (1992)), which is resolved so that we have V{...} after the **reduce** move. In the fourth rule, V′{V, ...} has an unfulfilled subcategorization requirement for a clause (and perhaps other elements). The **adjoin-reduce** adds V′ on top of V, representing the fact that the clausal subcategorization requirement of V′ is absolved by adjoining its tree into the tree of its subordinated clause.

Now let us turn to recognition of German center-embedded sentence (1). In (10), the column "Input" shows the symbol being scanned by the read head.

(10)

Move	Rule	Store	Input Read	Input Scanned
0	0			
1	1	[N_1	N_1	N_1
2	1	[N_1 [N_2	N_2	N_2
3	1	[N_1 [N_2 [N_3	N_3	N_3
4a	2	[N_1 [N_2 [N_3 [V_3{N_3}	V_3	V_3
4b	3	[N_1 [N_2 [V_3{}	—	V_2
5a	2	[N_1 [N_2 [V_3{} [V_2{N_2}	V_2	V_2
5b	4	[N_1 [N_2 [V_3{} V_2{N_2}	—	V_1
5c	3	[N_1 [V_3{} V_2{}	—	V_1
6a	2	[N_1 [V_3{} V_2{} [V_1{N_1}	V_1	V_1
6b	4	[N_1 [V_3{} V_2{} V_1{N_1}	—	—
6c	3	[V_3{} V_2{} V_1{}	—	—
6d	5	[V_3{} V_2{}	—	—
6e	5	[V_3{}	—	—
6f		[S		

In moves one through three, the NPs are recognized and stored in separate stacks in the pushdown store. In move 4a, a verb is read in. Since the case features match, the **reduce** rule derived from tree I applies and the nominal subcategorization requirement of V_3 is fulfilled. Note that we have now recognized tree I to the root node. However, it cannot be removed from the automaton, since a further **adjoin** (or, more precisely, **adjoin-reduce**) may occur at the root node, and in fact does occur in our example. (Put differently, we have really only recognized the bottom version of the root node.) We therefore read in the next input V_2 (move 5a), which can be **adjoin-reduced** with the top stack (move 5b). After move 5c, the top stack contains V_2{} on top of V_3{}, representing the fact that we have recognized V_2's tree A (with its nominal argument substituted) adjoined into V_3's tree I (with its nominal argument substituted). Finally, in moves 6a and 6b, we read in V_1, and **adjoin-reduce** it, and then **reduce** N_1 with the top stack. We can now **unadjoin** V_1{},

followed by $V_2\{\}$. We are now left with the root node of the initial tree I (strictly, speaking, the top version of the root node), and we have successfully finished our recognition.

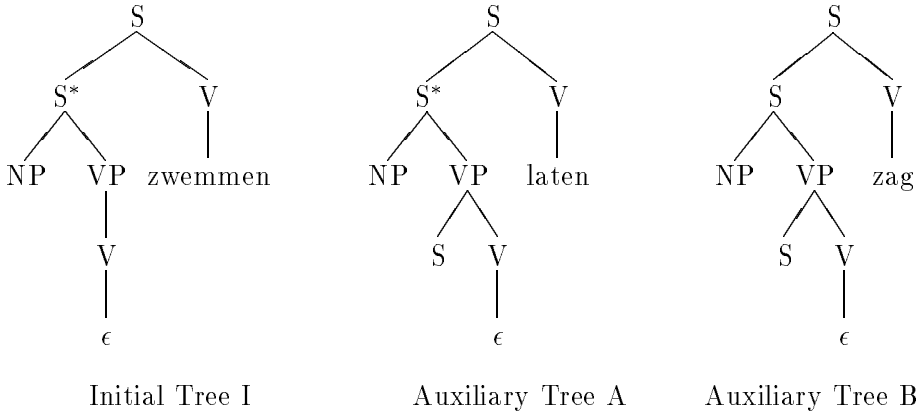


Figure 6.3: The Dutch grammar

We now turn to Dutch. We use the grammar for Dutch given in Kroch and Santorini (1991), repeated in Figure 6.3. Kroch and Santorini give extensive syntactic evidence for their grammar, but do not consider processing issues at all. The Dutch cross-serial dependencies are derived by adjoining Tree B into Tree A at the node marked S^*, and then adjoining the combination into Tree I, again at the node marked S^*. Note that the cross-serial dependencies are a result only of head movement (verb raising) that has occurred locally in each clause, not of multi-clausal ordering rules or verb-complex formation, operations that involve elements from different clauses (and hence trees).

We derive an automaton in an manner analogous to the German case. There is, however, one complication: the empty category that results from head movement of the verb. Since the bottom-up recognition of a tree can only proceed once the empty head has been posited, the automaton must have a rule for hypothesizing empty heads. Clearly, we do not want it to do so non-deterministically, and we must refine our rules for making deterministic BEPDAs derived by the easy method from a TAG. We do so by defining the following two conditions on processing empty heads, or, more precisely, on performing a ε-**reduce** move for empty heads. Both of the following two conditions must be met.

- **Bottom-up condition:** The automaton performs a ε-**reduce** move only if it has recognized its sister subtree and it can immediately subsequently perform a **reduce** operation involving the result of the ε-**reduce** move.
- **Top-down condition:** A ε-**reduce** move can only be made if the input symbol the read head is currently scanning licenses an empty category of the appropriate type.

According to the lexical determinism principle (6), we combine the ε-**reduce** with the **reduce** operation that licenses it into a single move (which we will refer to as a ε-**reduce-reduce** move). In the case of initial tree I, the **reduce** move in question involves an overt noun, and the resulting

complex move complies with the lexical determinism principle. However, we observe that in trees A and B the sister to the empty head is the footnode of the tree, which appears on the automaton only as the result of an **adjoin** move. As in the German case, we will want to combine the **adjoin** move with a **reduce** move, but combining the **adjoin** move with the ε-**reduce** move will still not involve an overt lexical item and therefore does not comply with the lexical determinism principle. We therefore combine these two moves with the subsequent **reduce** move that involves the overt nominal argument, and obtain a complex move, which we will call a **adjoin-ε-reduce-reduce** move.[5]

We are now in a position to present the Dutch automaton (we omit further details of the construction). We will denote the projection from a (verbal) head trace by V^{h-}, while we will use V^{h+} to indicate an overt full lexical head.

(11) **Automaton for Dutch:**

Rule	Read Head	Top of Stack	Action
1	N	*anything*	**terminal-shift** N
2	V^{h+}	*anything*	**terminal-shift** V^{h+}
3	V^{h+}	[N	ε-**reduce-reduce** [N to [V^{h-}{}
4	V^{h+}	[N [$\ldots V^{h-}${}	**adjoin-ε-reduce-reduce** [N, V^{h-}{} to V^{h-}{}V^{h-}{}
5	V^{h+}	V^{h-}	**reduce** V^{h-}{}, [V^{h+} to V^{h+}{}
5	*anything*	[$\ldots V^{h+}${}	**unadjoin** V{}

Recall that all possible **reduce** moves are performed before any possible **terminal-shift** moves.

Consider the following sentence, a simplified version of (2) (see also Figure 3.3, page 38 for a schematic depiction of the dependencies).

(12)

	...	omdat	Piet	Marie	de kinderen	zag	laten	zwemmen
	...	because	Piet	Marie	the children	saw	let	swim
			N_1	N_2	N_3	V_1^{h+}	V_2^{h+}	V_3^{h+}

...because Piet saw Marie let the children swim

Given the input N_1 N_2 N_3 V_1^{h+} V_2^{h+} V_3^{h+} , the automaton executes the following moves:

(13)

[5] Observe that the top-down EPDA has particular problems with Dutch if it is derived from the linguistically motivated grammars, since it must posit *all* the verbs before seeing any evidence for any of them (except perhaps the first one). Such an automaton cannot be made deterministic by any means. See (Joshi, 1990, Section 6) for a discussion.

Move	Rule	Store	Input Read	Input Scanned
0	0			
1	1	$[N_1$	N_1	N_1
2	1	$[N_1$ $[N_2$	N_2	N_2
3a	1	$[N_1$ $[N_2$ $[N_3$	N_3	N_3
3b	3	$[N_1$ $[N_2$ $[V_3^{h-}\{\}$	—	V_1^{h+}
3c	4	$[N_1$ $[V_3^{h-}\{\}$ $V_2^{h-}\{\}$	—	V_1^{h+}
3d	4	$[V_3^{h-}\{\}$ $V_2^{h-}\{\}$ $V_1^{h-}\{\}$	—	V_1^{h+}
4a	2	$[V_3^{h-}\{\}$ $V_2^{h-}\{\}$ $V_1^{h-}\{\}$ $[V_1^{h+}$	V_1^{h+}	V_1^{h+}
4b	5	$[V_3^{h-}\{\}$ $V_2^{h-}\{\}$ $V_1^{h+}\{\}$	—	V_2^{h+}
4c	6	$[V_3^{h-}\{\}$ $V_2^{h-}\{\}$	—	V_2^{h+}
5a	2	$[V_3^{h-}\{\}$ $V_2^{h-}\{\}$ $[V_2^{h+}$	V_2^{h+}	V_2^{h+}
5b	5	$[V_3^{h-}\{\}$ $V_2^{h+}\{\}$	—	V_3^{h+}
5c	6	$[V_3^{h-}\{\}$	—	V_3^{h+}
6a	2	$[V_3^{h-}\{\}$ $[V_3^{h+}$	V_3^{h+}	V_3^{h+}
6b	5	$[V_3^{h+}\{\}$	—	—
6c	6		—	—

First, the three nouns are read into the push-down store on separate stacks. Before move 1, the bottom-up condition for positing a headed null-subtree is not met. Before moves 2 and 3, the bottom-up condition is met, but not the top-down condition: the input head is not scanning a potential licensor. The latter condition is also met after move 3a, so that in moves 3b through 3d the head traces are posited and the complex **adjoin-ε-reduce-reduce** moves are performed. At the end of move 3d, the empty heads with saturated subcategorization requirements are stacked on one stack, representing the fact that tree A is adjoined into tree I, and tree B into tree A, with the adjunctions taking place below the lexical verb but above its trace. Then, in moves 4a to 4c, 5a to 5c, and 6a to 6c, the lexical heads are **terminal-shifted**, **reduced** with the top of the stack, and the completed structures are **unadjoined** off the push-down store.

6.2.2 Measuring Processing Load

Bach et al. (1986) show experimentally that native speakers of German find sentences with nested dependencies more difficult to understand than native speakers of Dutch find equivalent sentences with cross-serial dependencies. In the experiment, native speakers were asked to rate how easy to understand sentences with varying depths of embedding are, on a scale from 1 ("easy") to 9 ("difficult"). The native speakers were presented with sentences with center-embedded constructions (German) or cross-serial constructions (Dutch), as well as with sentences with extraposed constructions in both languages. The extraposed constructions presumably represent no syntactic difficulty for comprehension and therefore their scores reflect the purely propositional complexity. By subtracting the scores for the extraposed constructions from those for the embedded construction, we obtain a measure of the complexity contributed by the syntactic construction alone. The relevant results, taken from (Bach et al., 1986, Table 1), are summarized in (14).[6] The column

[6]The German scores represent the mean of the scores for constructions with inifinitives (the *Inifitiv pro Participio* or IPP-effect) or with a real participle, representing a dialectal variation.

"Absolute Scores" refers to the judgments of the embedded sentences, while the column "Difference Scores" reports the difference between the scores for the embedded constructions and those for extraposed constructions.

(14)

Level of	Absolute Scores		Difference Scores	
Embedding	Dutch	German	Dutch	German
1	1.14	1.17	—	—
2	2.34	2.70	0.23	0.51
3	5.42	5.99	1.36	2.00
4	7.66	7.96	1.72	2.20

Joshi (1990) presents a processing model based on a (top-down) EPDA that predicts these facts. In this section, we will show that the BEPDA model proposed in this paper makes the same predictions, while being derived from linguistically motivated grammars.

We will associate a metric with the run of the automaton. Joshi (1990) proposes two metrics, a simple one that registers the maximum number of items stored during processing of a given input sentence, and a more complex one which also takes into account how long each item spends in the push-down store. We will adopt the second approach, since it is better suited to account for the data we will address subsequently. However, we will modify it slightly. The basic idea is to record for each processing step how many items are stored in the push-down store (each lexical item is given a score of 1), and then sum up the scores for all moves. We can define "processing step" in different ways, giving us different metrics. Joshi (1990) defines a processing step to be the moves of the automaton starting with a **terminal-shift** up to (but excluding) the subsequent **terminal-shift**. Let us simply call such a processing step a "turn". We have indicated the turns of the automaton by giving them the same number, so that in (14) above, moves 3a though 3d are all moves of the third turn and form a single processing step under this definition. We will call the metric that is based on turns "Metric 1".[7] A second way to define a metric is is simply to define each move (simple or complex) of the automaton as a processing step. We will call this metric "Metric 2". Under this interpretation, each of Moves 3a through 3d are separate processing steps. Obviously, Metric 1 will always yield scores less than or equal to those of Metric 2.

For center-embedded sentences, the automaton for German gives us the following scores:[8]

(15) $\mathsf{N}_1\ \mathsf{N}_2\ \mathsf{N}_3\ \mathsf{V}_3\ \mathsf{V}_2\ \mathsf{V}_1$

[7] Joshi (1990) also adopts the convention of not counting elements that are added to and removed from the pushdown store in the same turn. For simplicity, we will not adopt this convention – there do not appear to be any important ramifications.

[8] Runs of the German automaton have been generated using a Lisp implementation of the German {}-BEPDA discussed in Section 6.2.1.

Step	Store	Metric 1 New	Metric 1 Cum	Metric 2 New	Metric 2 Cum
1a	[N_1	1	1	1	1
2a	[N_1 [N_2	2	3	2	3
3a	[N_1 [N_2 [N_3	3	6	3	6
4a	[N_1 [N_2 [N_3 [V_3 {N_3 }	4	10	4	10
4b	[N_1 [N_2 [V_3 {}		10	4	14
5a	[N_1 [N_2 [V_3 {} [V_2 {N_2 V_3 }	5	15	5	19
5b	[N_1 [N_2 [V_3 {} V_2 {N_2 }		15	5	24
5c	[N_1 [V_3 {} V_2 {}		15	5	29
6a	[N_1 [V_3 {} V_2 {} [V_1 {N_1 V_2 }	6	21	6	35
6b	[N_1 [V_3 {} V_2 {} V_1 {N_1 }		21	6	41
6c	[V_3 {} V_2 {} V_1 {}		21	6	47
6d	[V_3 {} V_2 {}		21	4	51
6e	[V_3 {}		21	2	53
6f			21	0	53

String accepted. Scores: 21 by Metric 1, 53 by Metric 2.

"New" refers to the score contributed by that step, while "Cum" is, of course, the cumulative total up to and including that step. Note that a verb with its subcategorization requirements fulfilled contributes a score corresponding to the verb and its nominal arguments. Now consider the Dutch automaton:

(16)

Step	Store	Metric 1 New	Metric 1 Cum	Metric 2 New	Metric 2 Cum
0			0		0
1	[N_1	1	1	1	1
2	[N_1 [N_2	2	3	2	3
3a	[N_1 [N_2 [N_3	3	6	3	6
3b	[N_1 [N_2 [V_3^{h-}{}		6	3	9
3c	[N_1 [V_3^{h-}{} V_2^{h-}{}		6	3	12
3d	[V_3^{h-}{} V_2^{h-}{} V_1^{h-}{}		6	3	15
4a	[V_3^{h-}{} V_2^{h-}{} V_1^{h-}{} [V_1^{h+}	4	10	4	19
4b	[V_3^{h-}{} V_2^{h-}{} V_1^{h+}{}		10	4	23
4c	[V_3^{h-}{} V_2^{h-}{}		10	2	25
5a	[V_3^{h-}{} V_2^{h-}{} [V_2^{h+}	3	13	3	28
5b	[V_3^{h-}{} V_2^{h+}{}		13	3	31
5c	[V_3^{h-}{}		13	1	32
6a	[V_3^{h-}{} [V_3^{h+}	2	15	2	34
6b	[V_3^{h+}{}		15	2	36
6c			15		36

Our model is based on the assumption that the automaton manipulates representations of lexical items. This assumption dictates a convention about scoring that we have made above: empty

heads do not contribute to the score, since they are not associated with any lexical item. The following table summarizes the scores for different depths of embedding.

(17)

Level of	Metric 1		Metric 2	
Embedding	Dutch	German	Dutch	German
1	3	3	5	5
2	8	10	18	23
3	15	21	36	53
4	24	36	60	95

We see that the results of our simulation match the psycholinguistic results obtained by Bach et al. (1986).

6.2.3 German Extraposition and Center-Embedding

Finally, without going into much detail, we give a run of an automaton for a German sentence with extraposition, N_1 V_1 N_2 V_2 N_3 V_3. The grammar is as given above in Figure 6.1, except that in trees A and B, the S node follows, rather than precedes, the V node.[9]

(18) N_1 V_1 N_2 V_2 N_3 V_3

Step	Store	Metric 1		Metric 2	
		New	Cum	New	Cum
1a	[N_1	1	1	1	1
2a	[N_1 [V_1 { N_1 V_2 }	2	3	2	3
2b	[V_1 { V_2 }		3	2	5
2c			3	0	5
3a	[N_2	1	4	1	6
4a	[N_2 [V_2 { N_2 V_3 }	2	6	2	8
4b	[V_2 { V_3 }		6	2	10
4c			6	0	10
5a	[N_3	1	7	1	11
6a	[N_3 [V_3 { N_3 }	2	9	2	13
6b	[V_3 {}		9	2	15
6c			9	0	15

String accepted. Scores: 9 by Metric 1, 15 by Metric 2.

We see that we can remove each clause as it is recognized, since we start with the matrix clause. In extraposed constructions, the scores grow linearly with the number of embedded clauses, while in center-embedded constructions, the scores grow with the square of the number of embeddings. The following table refers to German data:

[9] *Lassen* 'to let' does not allow extraposition. Any verb that takes a *zu*-infinitive does, and can be used in its stead.

(19)

Level of	Metric 1		Metric 2	
Embedding	Extraposed	Center-Emb.	Extraposed	Center-Emb.
1	3	3	5	5
2	6	10	10	23
3	9	21	15	53
4	12	36	20	95

The automaton model predicts strongly that extraposition is preferred over center-embedding, in particular at levels of embedding beyond two. This prediction is confirmed by native-speaker intuition, and we conjecture that psycholinguistic experiments would come to the same conclusions. Furthermore, we conjecture that an investigation of corpora would reveal very few, if any, examples of center-embedding beyond level two.

6.2.4 The "Principle of Partial Interpretation"

Why does the BEPDA automaton model make different predictions for Dutch and German sentences of comparable level of embedding? The main reason is that in the Dutch sentences, clauses are removed from the push-down store as soon as the first verb is read in, while in German, clauses are only removed once the last verb of the sentence has been processed. Bach et al. (1986) interpret their experimental data as suggesting such a behavior by the processor, and propose that this behavior arises because structures can only be removed from the processor once there is a place for them "to attach to" – an embedded clause cannot be removed while its matrix clause is still in the processor. Joshi (1990) proposes to formalize this intuition by defining a restriction on the way that automaton works, called the "Principle of Partial Interpretation" (PPI). The PPI makes the following two stipulations:

1. Structures are only discharged from the automaton when they are a properly integrated predicate-argument structure. More precisely, a clausal structure must contain all of the nominal arguments it subcategorizes for.

2. A structure is discharged only when it is either the root clause or it is the immediately embedded clause of the previously discharged structure.

In our discussion so far, we have not appealed to the PPI. For the types of structures under consideration, we have not needed to do so: the PPI is simply a consequence of the easy method used to derive the automaton from the competence grammar, and independently motivated ways in which the competence grammar is defined. The reason for this is that adjunction in the grammar is simulated in the automaton by recognizing the adjoined tree bottom-up and then removing any trace of it once its root node has been reached (the **unadjoin** move). Thus, the first material to be removed from the automaton corresponds to the last tree adjoined. Substitution, on the other hand, is handled differently: the material corresponding to the substituted tree is not removed from the automaton; in fact, it is treated as if it were part of the tree into which it was substituted. Thus, we see that the first part of the PPI follows from the fact that, in the

competence grammar, we substitute nominal arguments into the trees of their governing verbs. The second part of the PPI follows from the fact that, in the competence grammar, we adjoin matrix clauses into their subordinate clauses. We intend to investigate further whether the PPI is required as an independently stated principle of processing by considering other constructions from other languages.

While the predictions of the PPI and of the BEPDA model coincide in the cases that we have considered so far, there are constructions where they make differing predictions, and which therefore may be crucial in determining their empirical relevance. Consider subject relative clauses in a COMP-final language such as Korean.

(20) i kay-ka ccochko isste-n koyangi-i kkaymwu-n cwui-ka cheese-lul mekessta
this dog_{NOM} was $\text{chasing}_{\text{REL}}$ cat_{NOM} bit_{REL} rat_{NOM} $\text{cheese}_{\text{ACC}}$ ate

The rat that the cat that this dog was chasing bit ate the cheese.

According to the PPI, the relative clauses must remain on the push-down store until the end of the sentence is reached: a relative clause cannot be removed since it attaches to its noun, but the noun cannot be removed until its verb has been reached. Thus a PPI-based processing model would predict such constructions to be difficult to process, like German center-embedded sentences. According to the BEPDA model, however, the relative clauses can be removed from the push-down store once fully recognized, since in the competence grammar relative clauses are adjoined to the nominal tree. The BEPDA model would therefore predict these sentences to be equivalent to German extraposition sentences. Native speaker intuitions suggest that sentences such as (20) are in fact quite easy to process, thus supporting the pure BEPDA model over one in which the PPI is postulated separately. However, the evidence is only (barely) suggestive. We intend to pursue this line of inquiry.

6.3 Extending the Automaton for Long-Distance Effects

6.3.1 Nominal Scrambling and Extraposition

How do we handle long-distance scrambling? In Chapter 3 we argued that TAGs are not powerful enough to derive the full range of scrambled sentences. In Chapter 4 we introduced a multi-component extension of TAG called V-TAG. Recall that in V-TAG, several trees are grouped together into a set. There is no "locality" restriction on where we may adjoin trees from one set, and there is also no requirement that trees from one set be adjoined simultaneously. Furthermore, the trees in a set are connected by *dominance links* (indicated by dotted lines in the figures).

To make clear the connection between a linguistic V-TAG and the derived automaton, we will used a simplified example. Let us consider the tree set for *versprechen* 'to promise', shown in Figure 6.4. We see that the head and the PRO subject have already been adjoined to the projection tree in the lexical derivation. However, the position in the projection of the overt direct object is not specified and it can move away from the verb, while still receiving θ- and case-marking. As explained in Chapter 5, we have chosen to label all nodes in the projection of the verb with 'VP';

the functional information expressed by separate node labels in recent syntactic theories (IP, CP, AgrSP, etc.) are expressed as features (not shown here for simplicity).

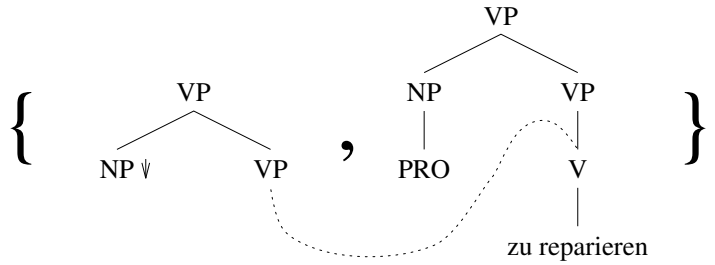


Figure 6.4: Multi-component tree set for *versprechen* 'to promise'

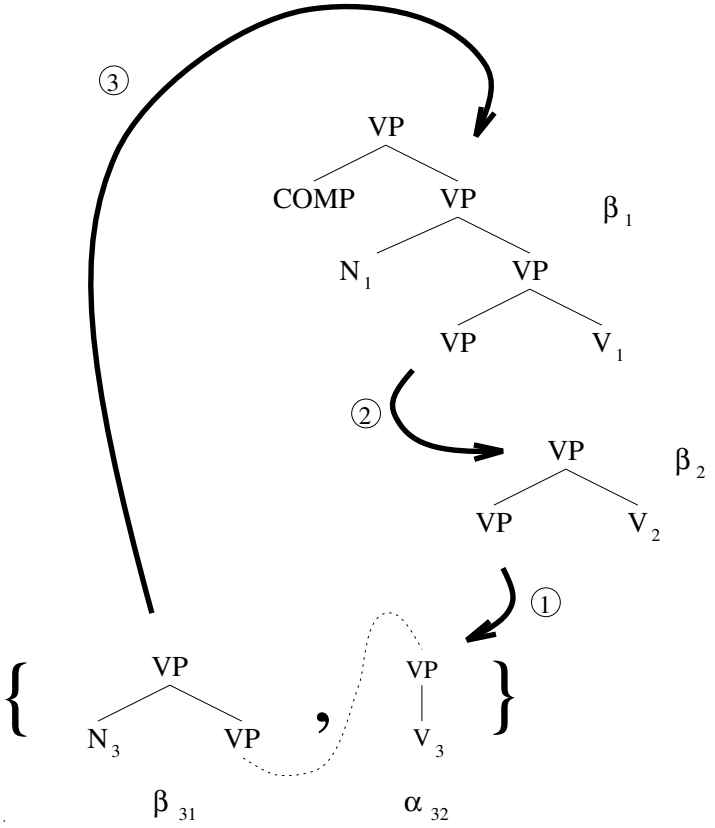


Figure 6.5: Grammar for German long-distance scrambling

How can we derive sentences with long-distance scrambling? Consider sentence (ii) from Figure 2.1, page 18, repeated here for convenience:

(21)	weil	[den Kühlschrank]$_i$	niemand	[[t$_i$ zu reparieren]	zu versuchen]	verpricht
	that	[the refrigerator]$_{\text{ACC}}$	no-one$_{\text{NOM}}$	to repair	to try	promises
	Comp$_1$	N$_3$	N$_1$	V$_3$	V$_2$	V$_1$

Because no-one promises to repair the refrigerator

The accusative NP *den Kühlschrank* 'the refrigerator' has been scrambled out of the most deeply embedded clause into the matrix clause.[10] We briefly review the linguistic derivation presented in Section 5.4.6, page 173. A complete grammar that can be used to derive this sentence is given in Figure 6.5. There are two auxiliary trees for the matrix verb V_1 *versprechen* 'to promise' (β_1) and for the intermediate verb V_2 *versuchen* 'to try' (β_2). Finally, there is the tree set introduced earlier for the most deeply embedded verb V_3, *reparieren* 'to repair', containing auxiliary tree β_{31} and initial tree α_{32}. In the interest of readability, we use abbreviations for the terminal symbols, and we omit empty categories (PRO). These issues do not affect our discussion.

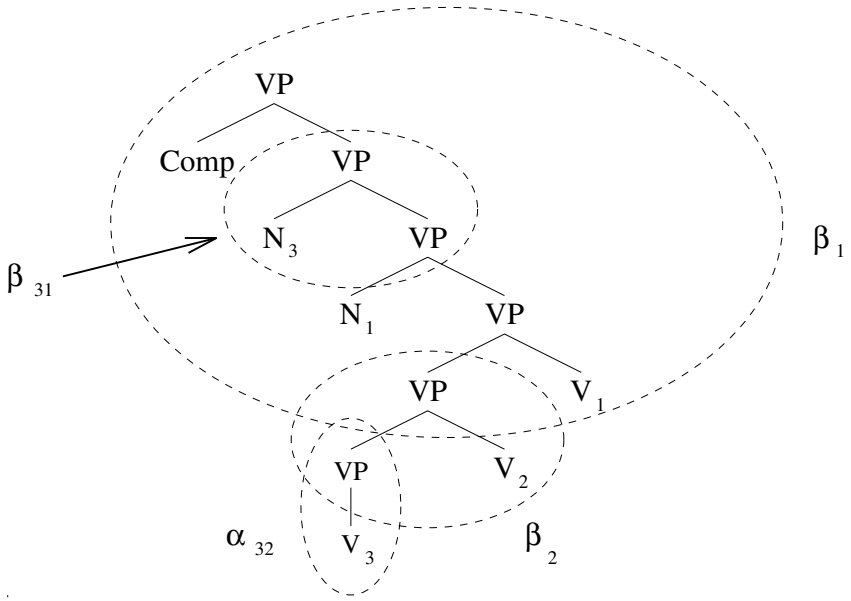


Figure 6.6: Derived tree for sentence (21)

The derivation is shown by the arrows in Figure 6.5. We start out by adjoining the intermediate clause into the verbal tree (α_{32}) of the most deeply embedded clause at its root node, and then we adjoin the matrix clause into the intermediate clause at the root of the intermediate clause. Since we have not yet used the nominal argument tree from the most deeply embedded clause (tree β_{31}), the derivation is not yet complete. We choose to adjoin the most deeply embedded argument into the matrix clause, at the VP node between the complementizer and its nominal argument. This choice corresponds to long-distance scrambling. The resulting derived tree is shown in Figure 6.6.

Now let us turn to our processing model. Since the BEPDA is formally equivalent to TAG, and since TAG is formally inadequate for the long-distance phenomena we are interested in, we will use the {}-BEPDA defined in Section 4.6.4, page 112, which we have shown is equivalent to V-TAG. We informally review the definition of the {}-BEPDA, and as mentioned in Section 4.6.4, we will make use of the "extended definition", which is a notational variant of the original definition, convenient for our purposes. In the {}-BEPDA, every stack symbol in the pushdown store is

[10] Of course, (21) is embedded in some other clause which we consistently omit in order to avoid the complications of the verb-second effect. Our use of the term 'matrix clause' to denote the topmost of the recursively embedded clauses is thus sloppy, but the intended meaning is clear.

associated with a set of indices. Intuitively, these indices represent trees that still need to be adjoined in order for the derivation to be successful. Since we are using auxiliary trees in sets to represent nominal arguments, we can think of these sets as unfulfilled (nominal) subcategorization requirements. We see that the notation is consistent with the quasi-categorial notation we adopted previously. If we do not allow stack symbols to pass subcategorization requirements to other stack symbols, we simply have a BEPDA which just recognizes tree adjoining languages. However, if we allow symbols in the same stack of the push-down store to pass a subcategorization requirement to a stack symbol immediately above or below it, then we increase the power since we can now simulate the "detaching" of nominal arguments from their verbs.

The "easy construction" of a {}-BEPDA from a V-TAG. which is an extension of the easy construction of a BEPDA from a TAG, was given in Section 4.6.4, page 115. The types of moves of the thus constructed {}-BEPDA are the same as those of a BEPDA constructed by the easy method, except that the moves also detail the actions on index sets. In addition, we have the **raise** and **lower** moves to transfer indices between the index sets of stack symbols within the same stack. We claim that the **lower** move is never used. The **raise** move is ordered behind all other moves except **terminal-shift**, it must be performed on the top stack, and it must be specifically motivated by a lexical item on the input tape or on the second stack (i.e., that item must match the **raised** subcategorization requirement). We will take these stipulations as a way of reducing processing complexity, by limiting searches through potentially unbounded stacks (though of course such searches cannot be avoided altogether).

We will illustrate the functioning of the {}-BEPDA by showing how it performs on sentence (ii) (= (21)), which is like the simple center-embedded sentence (i), except that the most embedded argument has scrambled long-distance.

(22) Sentence (ii): $\mathsf{Comp}_1\ \mathsf{N}_3\ \mathsf{N}_1\ \mathsf{V}_3\ \mathsf{V}_2\ \mathsf{V}_1$ Judgment: ?

Step	Store	Metric 1 New	Metric 1 Cum	Metric2 New	Metric2 Cum
1a	[Comp_1	1	1	1	1
2a	[Comp_1 [N_3	2	3	2	3
3a	[Comp_1 [N_3 [N_1	3	6	3	6
4a	[Comp_1 [N_3 [N_1 [V_3 {N_3}	4	10	4	10
5a	[Comp_1 [N_3 [N_1 [V_3 {N_3} [V_2 {V_3}	5	15	5	15
5b	[Comp_1 [N_3 [N_1 [V_3 {N_3} V_2 {}		15	5	20
6a	[Comp_1 [N_3 [N_1 [V_3 {N_3} V_2 {} [V_1 {N_1 V_2}	6	21	6	26
6b	[Comp_1 [N_3 [N_1 [V_3 {N_3} V_2 {} V_1 {N_1}		21	6	32
6c	[Comp_1 [N_3 [V_3 {N_3} V_2 {} V_1 {}		21	6	38
6d	[Comp_1 [N_3 [V_3 {} V_2 {N_3} V_1 {}	6	27	6	44
6e	[Comp_1 [N_3 [V_3 {} V_2 {} V_1 {N_3}	6	33	6	50
6f	[Comp_1 [V_3 {} V_2 {} V_1 {}		33	6	56
6g	[V_3 {} V_2 {} V_1 {}		33	5	61
6h	[V_3 {} V_2 {}		33	3	64
6i	[V_3 {}		33	2	66
6j			33	0	66

String accepted. Scores: 33 by Metric 1, 66 by Metric 2.

In steps 1 through 3, the complementizer and the two nouns are read in. In step 4, verb V_3 is

read in, but no **reduce** is possible, since V_3 is not next to its nominal argument (as determined by case features). In steps 5a and 5b, verb V_2 is read in and **adjoined**. In steps 6a, 6b, and 6c, the last verb V_1 is read in, and **adjoined** around the stack of verbs and its nominal argument. At this point, no further reduction is possible without passing index symbols within a stack. This happens in steps 6d and 6e. When V_1 has "inherited" the subcategorization requirement of V_3, N_3 can be discharged. The fact that N_3 is **reduced** with V_1 (and absolves a subcategorization requirement out of the index set of V_1) corresponds to the fact that in the linguistic analysis it has scrambled into the clause of V_1.

When considering cases involving long-distance scrambling, we must re-evaluate the metrics. Metric 2, in which a processing step is interpreted as a move of the automaton, captures the added processing required for long-distance scrambling since the **raise** operation is a separate processing step. Metric 1 as currently defined, however, equates a processing step with all the moves executed following a **terminal-shift**. Clearly, this metric will not assign different scores to sentence pairs such as (i) and (ii), which differ only in that the two nominal arguments are permuted, with (ii) showing long-distance scrambling. If Metric 1 is to be useful, we must therefore extend its definition. We will say that a new processing step for Metric 1 is initiated either by a **terminal-shift** move or a **raise** move. The definition of Metric 1 thus loses its intuitive simplicity.

If we apply this method to some other sentences in Figure 2.1 which only involve nominal scrambling and extraposition, we get the following results:

(23)

No.	Sentence	Score		Judgment
		Metric 1	Metric 2	
(xvi)	$Comp_1$ N_1 V_1 V_2 N_3 V_3	10	17	ok
(xiii)	$Comp_1$ N_1 V_1 N_3 V_3 V_2	12	24	ok
(i)	$Comp_1$ N_1 N_3 V_3 V_2 V_1	21	52	(ok)
(iv)	$Comp_1$ N_1 N_3 V_2 V_3 V_1	26	58	?
(ii)	$Comp_1$ N_3 N_1 V_3 V_2 V_1	33	66	?

Sentence (xvi) is fully extraposed: the score is low because material can be removed from the processor as soon as a clause is complete. In sentence (xiii), the two most embedded clauses have been extraposed behind the matrix clause, but they have been left in a center-embedded construction. Thus the matrix clause can be removed from the automaton before any embedded clause is reached, but then the items must remain in the automaton until the whole sentence has been read in. Sentence (i), which is prescriptively acceptable, is simply the fully center-embedded version; all lexical items must remain in the automaton until the entire sentence has been read in. This is also true for sentence (iv), but the score gets even worse, since N_3 has been long-distance scrambled out of the most deeply embedded clause (which has been extraposed) into the second clause. Finally, sentence (ii) is worst of all, since here there is long-distance scrambling over two clause boundaries. We see that the ordering of scores corresponds to the ordering by acceptability that we proposed earlier (recakll that (i) is only prescriptively ok). Furthermore, the ordering within each judgment group (ok and ?) are compatible with native-speaker intuition. Generally speaking, extraposition improves sentences, while long-distance scrambling degrades them. We have summed up all results in Figure 6.7.

Input		Judgment	Metric 1	Metric2
No Extraposition Past verspricht				
(i)	$Comp_1\ N_1\ N_3\ V_3\ V_2\ V_1$	ok	21	52
(ii)	$Comp_1\ N_3\ N_1\ V_3\ V_2\ V_1$	?	33	66
(iii)	$Comp_1\ N_1\ V_2\ N_3\ V_3\ V_1$	?	21	53
(iv)	$Comp_1\ N_1\ N_3\ V_2\ V_3\ V_1$	?	26	58
(v)	$Comp_1\ N_3\ N_1\ V_2\ V_3\ V_1$	?	33	66
(vi)	$Comp_1\ N_3\ V_3\ N_1\ V_2\ V_1$	ok	27	56
(vii)	$Comp_1\ N_3\ V_3\ V_2\ N_1\ V_1$	?	21	50
(viii)	$Comp_1\ V_2\ N_3\ V_3\ N_1\ V_1$	?	21	51
(ix)	$Comp_1\ V_2\ N_3\ N_1\ V_3\ V_1$	*	REJECTED	
(x)	$Comp_1\ N_3\ V_2\ N_1\ V_3\ V_1$	*	REJECTED	
(xi)	$Comp_1\ V_2\ N_1\ N_3\ V_3\ V_1$	*	REJECTED	
(xii)	$Comp_1\ N_3\ V_2\ V_3\ N_1\ V_1$	?	25	55
Extraposing the versuchen *Clause*				
(xiii)	$Comp_1\ N_1\ V_1\ N_3\ V_3\ V_2$	ok	12	24
(xiv)	$Comp_1\ N_1\ N_3\ V_1\ V_3\ V_2$	?	33	67
(xv)	$Comp_1\ N_3\ N_1\ V_1\ V_3\ V_2$	?	33	65
(xvi)	$Comp_1\ N_1\ V_1\ V_2\ N_3\ V_3$	ok	10	17
(xvii)	$Comp_1\ N_1\ V_1\ N_3\ V_2\ V_3$	?	15	28
(xviii)	$Comp_1\ N_1\ N_3\ V_1\ V_2\ V_3$	ok	39	64
(xix)	$Comp_1\ N_3\ N_1\ V_1\ V_2\ V_3$	?	39	62
(xx)	$Comp_1\ N_1\ N_3\ V_3\ V_1\ V_2$	?	27	57
(xxi)	$Comp_1\ N_3\ V_3\ N_1\ V_1\ V_2$	?	27	55
(xxii)	$Comp_1\ N_3\ N_1\ V_3\ V_1\ V_2$	?	39	65
Extraposing Only the reparieren *Clause*				
(xxiii)	$Comp_1\ N_1\ V_2\ V_1\ N_3\ V_3$	?	24	48
(xxiv)	$Comp_1\ N_1\ V_2\ N_3\ V_1\ V_3$	*	REJECTED	
(xxv)	$Comp_1\ N_1\ N_3\ V_2\ V_1\ V_3$	*?	39	64
(xxvi)	$Comp_1\ N_3\ N_1\ V_2\ V_1\ V_3$	*?	39	63
(xxvii)	$Comp_1\ V_2\ N_1\ N_3\ V_1\ V_3$	*	REJECTED	
(xxviii)	$Comp_1\ V_2\ N_1\ V_1\ N_3\ V_3$	*?	24	48
(xxix)	$Comp_1\ V_2\ N_3\ N_1\ V_1\ V_3$	*	REJECTED	
(xxx)	$Comp_1\ N_3\ V_2\ N_1\ V_1\ V_3$	*?	39	63

Figure 6.7: Results of running the automaton on 30 sentences

6.3.2 Clausal Scrambling

We now turn to sentences that also include clausal scrambling. Recall from Section 5.4.7, page 177, that we handle local and long-distance scrambling differently. In local scrambling, the clausal subcategorization structure is adjoined during the lexical derivation above one or more nominal arguments. The resulting tree for the syntactic derivation tree is an auxiliary tree which is adjoined, in the standard manner, into its subordinate clause. Consider the run of the automaton for sentence (vii):

Sentence (vii): $\mathsf{Comp}_1\ \mathsf{N}_3\ \mathsf{V}_3\ \mathsf{V}_2\ \mathsf{N}_1\ \mathsf{V}_1$ Judgment: ?

Step	Store	Metric 1 New	Metric 1 Cum	Metric2 New	Metric2 Cum
1a	[Comp_1	1	1	1	1
2a	[Comp_1 [N_3	2	3	2	3
3a	[Comp_1 [N_3 [V_3 {N_3}	3	6	3	6
3b	[Comp_1 [V_3 {}		6	3	9
4a	[Comp_1 [V_3 {} [V_2 {V_3}	4	10	4	13
4b	[Comp_1 [V_3 {} V_2 {}		10	4	17
5a	[Comp_1 [V_3 {} V_2 {} [N_1	5	15	5	22
6a	[Comp_1 [V_3 {} V_2 {} [N_1 [V_1 {N_1 V_2}	6	21	6	28
6b	[Comp_1 [V_3 {} V_2 {} [V_1 {V_2}		21	6	34
6c	[Comp_1 [V_3 {} V_2 {} V_1 {}		21	6	40
6d	[V_3 {} V_2 {} V_1 {}		21	5	45
6e	[V_3 {} V_2 {}		21	3	48
6f	[V_3 {}		21	2	50
6g			21	0	50

String accepted. Scores: 21 by Metric 1, 50 by Metric 2.

We see that the score is marginally reduced as compared to the center-embedded case (sentence (i)), since the combination of N_3 and V_3 happens when fewer elements are on the stack (Step 3a). This score is compatible with intuitions (recall that intuitions are to be taken with even more caution than usually when they concern the prescriptively acceptable center-embedded construction). The situation is more complex with long-distance scrambling. In that case, we suggested in Section 5.4.7, page 177 that clausal subcategorization is handled by substitution, just like nominal subcategorization. This will be reflected in our automaton model by the fact that we will **raise** clausal subcategorization requirements within single stacks, just as we do with nominal subcategorization requirements. A problem arises. Suppose we have the configuration [V_2{V_3} V_1{} [V_3{N_3}. We perform a **raise** move to obtain [V_2{} V_1{V_3} [V_3{N_3}, and then **reduce** in the usual manner to get [V_2{} V_1{}. But what do we do with the nominal subcategorization requirement of V_3? If we simply add it to the top of the stack with which V_3 is **reduced**, we do not properly account for the fact that the nominal argument of V_3has also long-distance scrambled. We will therefore assign its subcategorization requirement to the governor of V_3 (which must be in the stack since otherwise V_3 could not have been **reduced** with the stack. As an example, consider sentence (xxv), judged extremely marginal.

(24) Sentence (xxv): $\mathsf{Comp}_1\ \mathsf{N}_1\ \mathsf{N}_3\ \mathsf{V}_2\ \mathsf{V}_1\ \mathsf{V}_3$ Judgment: *?

Step	Store	Metric 1		Metric2	
		New	Cum	New	Cum
1a	[$Comp_1$	1	1	1	1
2a	[$Comp_1$ [N_1	2	3	2	3
3a	[$Comp_1$ [N_1 [N_3	3	6	3	6
4a	[$Comp_1$ [N_1 [N_3 [V_2 {V_3}	4	10	4	10
5a	[$Comp_1$ [N_1 [N_3 [V_2 {V_3} [V_1 {N_1 V_2}	5	15	5	15
5b	[$Comp_1$ [N_1 [N_3 [V_2 {V_3} V_1 {N_1}		15	5	20
6a	[$Comp_1$ [N_1 [N_3 [V_2 {V_3} V_1 {N_1} [V_3 {N_3}	6	21	6	26
6b	[$Comp_1$ [N_1 [N_3 [V_2 {} V_1 {V_3 N_1} [V_3 {N_3}	6	27	6	32
6c	[$Comp_1$ [N_1 [N_3 [V_2 {N_3} V_1 {N_1}	6	33	6	38
6d	[$Comp_1$ [N_1 [N_3 [V_2 {} V_1 {N_3 N_1}	6	39	6	44
6e	[$Comp_1$ [N_1 [V_2 {} V_1 {N_1}		39	6	50
6f	[$Comp_1$ [V_2 {} V_1 {}		39	6	56
6g	[V_2 {} V_1 {}		39	5	61
6h	[V_2 {}		39	3	64
6i			39	0	64

String accepted. Scores: 39 by Metric 1, 64 by Metric 2.

Three of the four sentences judged extremely marginal, (xxv), (xxvi), and (xxx), receive high scores (63 or 64 by Metric 2). Sentence (xxviii) has a score of 48 (by Metric 2). We take the fact that it is more degraded than the score suggests to be related to the pragmatic problems: it is extremely difficult to construct a context that warrants local scrambling of the *versuchen* 'to try' clause while its embedded *reparieren* 'to repair' clause has been extraposed. We observe that the equivalent sentence without the local scrambling of V_2, (xxiii), appears to be quite good. Such a contrast between absence and presence of local scrambling does not lead to a similar contrast in judgments in the pair (i), (vii). Finally, there cannot be a competence restriction on extraction from scrambled clauses, as shown in (24):

(25) ? Weil das Fahrrad der Meister zu reparieren niemandem versprochen hat
because the bike no-one to repair the master promised has

Because no-one has promised the master to repair the bike

This sentence is marginally acceptable in a context in which *zu reparieren* receives contrastive stress. We conclude that the degradation of (xxviii) is not due to either competence or processing factors, but to pragmatic factors.

6.3.3 Problems

The full range of sentences including judgments and scores in shown in Figure 6.7. Roughly, one would want scores under 40 to correspond to "ok", scores between 41 and 60 correspond to "?", and scores above correspond to "*?". This section will discuss several problems related to these predictions.

We have seen that the automaton model makes plausible predictions with respect to the following phenomena:

- Extraposition is greatly preferred over center-embedding.
- Nominal and clausal scrambling lead to degradation.

However, there are several problematical predictions. We have seen the case of sentence (xxviii), which is predicted to be better than it is judged. This we have explained by appealing to pragmatic factors, and the impossibility of finding a coherent context which would license the word order. Far more troubling for the model, however, are those cases in which the prediction is that the sentence should be *worse* than it is actually judged to be: clearly, we cannot appeal to pragmatic circumstance to make a sentence better! We have already pointed out that the prescriptivist/metalinguistic influence on native speakers may make them judge center-embedded structures better than psycholinguistic experiments would reveal them to be, since the grammaticality judgments that linguists are trained to make abstracts from processing difficulty. This explains why sentence (i), with a Metric 2 score of 52, is judged "ok". However, it does not explain why sentence (ii), which is like sentence (i) except for long-distance scrambling of the embedded object, has a score of 66, while sentences such as (xxv), which borders on unintelligible, had a score of 64. The comparisons between the scores of sentences (i), (ii) and (xxv), appear to make wrong predictions, namely that sentence (ii) is more like (xxv) in processing difficulty than like sentence (i).

A possible solution is to assume that, *contra* our own argument in Section 6.3.1, nominal scrambling in fact does not contribute to processing difficulty. Let us pursue this possibility a bit. If we assume that nominal scrambling is "free", sentences (i), (ii), and (xxv) have scores of 52, 54, and 58, respectively. This is compatible with the judgments, since, given the problem of the prescriptive correctness of (i), the difference in judgments between (i) and (ii) on the one hand and (xxv) on the other is qualitative, not quantitative. Observe that such an analysis would in fact relate to the "clause union" interpretation of the linguistic system that we discussed in Section 5.4.4, page 166. There, we suggested that the competence grammar that we have proposed allows derivations that can be interpreted as the formation of a complex predicate with a merged θ–grid, which is what underlies the compelling intuition shared by native speakers about the relevant construction and which motivated the analysis of Evers (1975) and subsequent work. The processing account proposed here would then implement the other part of Evers's analysis, namely the "clause union" analysis: by associating a single index set with verbs stacked in a stack, we dissolve the distinction between the clauses that these verbs head. Thus, we interpret the clause union as an effect of the processor, rather than as a fact of the competence grammar.

The problem then remains of accounting for acceptability fluctuations due to word order variations within the *Mittelfeld*. We could adopt an approach similar to that proposed by Uszkoreit (1987, Chapter 5), which we diuscussed in Section 5.6.4, page 198. Uszkoreit uses disjunctive LP rule bundles to encode information about MF ordering.[11] In this manner, different types of factors can be expressed in the same framework, such as the default word order, the focus, the syntactic form of the constituent (pronoun). (See Lenerz (1977), Höhle (1982) for arguments for the need for considering these factors.) Presumably, other factors such as agenthood/empathy (Lenerz (1977)'s "Mitteilungszentrum") could also be incorporated. A disjunctive LP rule is satisfied if

[11] We would want to implement such conditions in terms of feature structures, extending their definition in a similar way.

it is true that when at least one of its disjuncts is violated, then at least one of them is satisfied. Uszkoreit also proposes that this system can make predictions about acceptability through the number of violated LP statements in a disjunctive LP rule.[12] While it may seem inelegant to make predictions about word order variations within the *Mittelfeld* using a completely different mechanism from the syntactic automaton (which is of course still needed for predictions relating extraposition and clausal scrambling, among others), it is clear that the overall acceptability of a sentence cannot be entirely determined by an automaton that models only syntactic processing. The exact demarcation of the role of the syntactic automaton in determining acceptability is an empirical question. While we probably want the automaton to predict a strong preference for extraposition, it is not *a priori* certain that we want it to make predictions about order within the *Mittelfeld*. A major open problem with Uszkoreit's approach is the question of how it extends to multclausal *Mittelfelder*. Further research is required.

6.4 Topicalization

In Section 2.4, page 30, we presented evidence for a difference in processing load between long-distance topicalization and long-distance scrambling (over comparable distances). We repeat the contrasting sentences, first given as (49), page 31:

(26) a. Sentence with long-distance scrambling:

? Der Meister	hat	den Kühlschrank	niemandem	zu reparieren	versprochen
[the master]$_{\mathrm{NOM}}$	has	[the refrigerator]$_{\mathrm{ACC}}$	no-one$_{\mathrm{DAT}}$	to repair	promised
N_{11}	Aux_1	N_2	N_{12}	V_2	V_1

The master has promised no-one to repair the refrigerator

b. Sentence with long-distance topicalization:

Den Kühlschrank	hat	der Meister	niemandem	zu reparieren	versprochen
[the refrigerator]$_{\mathrm{ACC}}$	has	[the master]$_{\mathrm{NOM}}$	no-one$_{\mathrm{DAT}}$	to repair	promised
N_2	Aux_1	N_{11}	N_{12}	V_2	V_1

The master has promised no-one to repair the refrigerator

Let us first consider the processing of the sentence with long-distance scrambling. The run of the automaton is similar to the one for sentence (21), except that we now have a full sentence with a matrix auxiliary in second position. We will assume it has been adjoined to the matrix verb and contributes features, in a manner similar to the complementizer in our previous example.

[12] Psycholinguistic experiments are currently underway to test these predictions. Initial results are encouraging. (Hans Uszkoreit, personal communication.)

(27)

Step	Store	New	Cum
1	$[N_{11}$	1	1
2	$[N_{11}$ $[Aux_1$	2	3
3	$[N_{11}$ $[Aux_1$ $[N_2$	3	6
4	$[N_{11}$ $[Aux_1$ $[N_2$ $[N_{12}$	4	10
5	$[N_{11}$ $[Aux_1$ $[N_2$ $[N_{12}$ $[V_2\{N_2\}$	5	15
6a	$[N_{11}$ $[Aux_1$ $[N_2$ $[N_{12}$ $[V_2\{N_2\}$ $[V_1\{N_{11}, N_{12}\}$	6	21
6b	$[N_{11}$ $[Aux_1$ $[N_2$ $[N_{12}$ $[V_2\{N_2\}$ $V_1\{N_{11}, N_{12}\}$	6	27
6c	$[N_{11}$ $[Aux_1$ $[N_2$ $[V_2\{N_2\}$ $V_1\{N_{11}\}$	6	33
6d	$[N_{11}$ $[Aux_1$ $[N_2$ $[V_2\{\}$ $V_1\{N_{11}, N_2\}$	6	39
6e	$[N_{11}$ $[Aux_1$ $[V_2\{\}$ $V_1\{N_{11}\}$	6	45
6f	$[N_{11}$ $[V_2\{\}$ $V_1\{N_{11}\}$	5	50
6g	$[V_2\{\}$ V_1 $\{\}$	5	55
6h	$[V_2\{\}$	2	57
6i		0	57

Again, the long-distance scrambling is achieved by passing the subcategorization requirement from a verb to its governing verb (in step 6d), which represents the fact that N_2 has scrambled into the matrix clause. Now let us consider the run of the automaton in the topicalized case.

(28)

Step	Store	New	Cum
1	$[N_2$	1	1
2	$[N_2$ $[Aux_1$	2	3
3	$[N_2$ $[Aux_1$ $[N_{11}$	3	6
4	$[N_2$ $[Aux_1$ $[N_{11}$ $[N_{12}$	4	10
5	$[N_2$ $[Aux_1$ $[N_{11}$ $[N_{12}$ $[V_2\{N_2\}$	5	15
6a	$[N_2$ $[Aux_1$ $[N_{11}$ $[N_{12}$ $[V_2\{N_2\}$ $[V_1\{N_{11}, N_{12}\}$	6	21
6b	$[N_2$ $[Aux_1$ $[N_{11}$ $[N_{12}$ $[V_2\{N_2\}$ $V_1\{N_{11}, N_{12}\}$	6	27
6c	$[N_2$ $[Aux_1$ $[N_{11}$ $[V_2\{N_2\}$ $V_1\{N_{11}\}$	6	33
6d	$[N_2$ $[Aux_1$ $[V_2\{N_2\}$ $V_1\{\}$	6	39
6e	$[N_2$ $[V_2\{N_2\}$ $V_1\{\}$	5	44
6f	$[N_2$ $[V_2\{N_2\}$	2	46
6g	$V_2\{\}$	2	48
6h		0	48

In the case of topicalization into sentence-initial position, we see that it is not necessary to pass subcategorization requirements among verbs. Instead, once the matrix clause has been removed (step 6e), the embedded verb is adjacent to its argument which can **reduce** in the usual manner. This results in a lower score (48 as opposed to 57). We see that by **reducing** the embedded argument with its governing verb, we are simulating the fact that that adjunction may be part of the lexical derivation, not the syntactic derivation, thus mirroring the linguistic analysis we proposed in Section 5.4.9: while scrambling is achieved by adjoining NP arguments separately and in arbitrary order, topicalization is achieved by choosing a different elementary tree prior to the syntactic derivation. In topicalization, the long-distance effect is achieved by adjoining the matrix clause below the topicalized element, thus stretching it away from its verb. We have

seen in Section 5.4.10 that the difference in representation of scrambling and topicalization in the competence grammar is justified by the linguistic differences between these two word order variation types. Again, we see that the independently motivated competence theory leads to automata models that make highly plausible predictions.

6.5 Discussion

We have presented a model of human syntactic processing that makes plausible predictions for a range of word-order variation phenomena in German (and Dutch). Our model of the human syntactic processor is directly linked to a V-TAG-based model of human syntactic competence. This direct link gives our model two major characteristics that differentiate it from other models:

- The processor is not concerned directly with phrase-structure trees, but with relations between lexical items.
- The processor is defined in terms of a set of formally defined operations, which may at first appear arbitrary.

We will briefly discuss these two points in turn, and finish with a brief note on syntactic ambiguity.

The parser simulates a derivation in the formalism of the competence theory, TAG. TAG is a tree-rewriting system, and therefore derivations in a TAG are not recorded by a phrase-structure tree (as is the case for CFG), but by the so-called *derivation tree*, which is a tree that represents adjunctions and substitutions performed during the derivation. Each node in the derivation tree corresponds to one elementary tree, and a dominance relation represents adjunction (or substitution) of the tree represented by the daughter node into the mother. Since the representation of competence exploits the lexicalizability of TAG, each tree in our competence grammar is associated with one lexical item. This means that the parser is in fact establishing direct dependencies between lexical items (heads). However, our approach does not build a phrase-structure tree (though one can be derived from its actions, just as one can be derived from the derivation tree). While explanatory approaches such as Minimal Attachment are less appealing in our model (since no phrase-structure tree is explicitly represented during the parse), licensing properties of lexical items can be represented in straightforward manner. In this respect, our approach is close to licensing-based or head-driven approaches (Abney, 1986; Pritchett, 1991). Lexical licensing relations, in particular θ-role assignment, also play a crucial role in approaches that are not "head-driven" (Gibson, 1991; Inoue and Fodor, 1993). We suspect that such conditions will find a straightforward representation in our processing model. Furthermore, the lexicon-oriented processing model allows for an elegant integration of lexical co-occurrence effects, which, it is generally believed, play a crucial role in parsing.

The second characteristic of our model that we would like to discuss is its very precise definition, which may seem somewhat arbitrary at first: why does the push-down store contain stacks of stack symbols, and why may each stack symbol be associated with an index set? The justification for this machinery comes from a careful study of the requirements of competence syntax. It is known that (case-marked) cross-serial dependencies are not context-free (Shieber, 1985) – therefore, the

representation of competence syntax cannot be based on a transformation-free CFG, nor can the parser be, say, a simple PDA. The representation of competence must therefore either include transformations, in which case we give up formal constraints and any hope that a formal analysis can guide us in modeling the parser, or we can look for other (more powerful yet still constrained) formalisms for the expression of competence. We claim that the complexity of the machinery of the processing model is justified by the details of the competence model. If the processing model makes empirically interesting predictions, then the complexity of its operations and data structures should not be held against it (on the basis of scientific parsimony) since they are independently motivated.

Finally, we need to address the issue of ambiguity. The model, as presented in (Joshi, 1990) and extended and modified here, does not address the issue of the resolution of syntactic ambiguity. (The reader will have observed that in all of our examples, the syntactic structure is in fact unambiguous – partly due to the case-marking. Furthermore, in all cases, the processing difficulties are not of the garden-path variety, since they persist even when the reader is primed for the syntactic structure.) *A priori*, it seems that our model can be integrated into a variety of ambiguity-resolution models, including parallelism, limited parallelism, deterministic with lookahead, and serial with (limited) backtracking. In further work, we intend to investigate whether the particular features of our model favor one or the other of these approaches.

Chapter 7

Conclusion

This thesis has attempted to develop a formal framework for the representation of the syntax of free word-order languages. This formal framework, called V-TAG, has been carefully motivated. The initial assumptions were kept to a minimum:

- We proposed to develop a formally restricted mathematical framework. This choice was motivated both from meta-theoretical considerations stemming from the theory of syntax and from practical considerations relating to natural language processing.
- We proposed to use a representation that explicitly encodes phrase-structure. This choice is perhaps more arbitrary than the first one; its motivation must come from the success of the research.

Beyond these initial assumptions, we have carefully motivated each feature of V-TAG from the linguistic facts.

The definition of V-TAG that is derived from an examination of the data has been tested in two ways. First, it has been subjected to a rigorous formal examination. Second, we have examined whether it can in fact be used to express an interesting theory of grammar. In both instances, we have found V-TAG to be well-suited for the description of natural language syntax.

Formally, while the resulting system is considerably more powerful than, for instance, context-free grammar, it has properties that make it attractive for formal and computational purposes. We have shown that in its lexicalized version, it is both no more powerful than context-sensitive grammar (and presumably less so), and polynomially parsable. Furthermore, we have sketched a parser which shows increased time complexity only in the presence of long-distance dependencies.

Linguistically, we have sketched a lexicon-based theory of grammar which is expressed in V-TAG (and a related formalism, UVG-DL). This theory of grammar, V-grammar,follows the principles-and-parameters methodology. The sole locus of application of principles and parameters is the lexicon; all syntactic derivations are based on purely formal operations. This linguistic theory has not been developed in a vacuum: it bears crucial resemblances to various recent developments in syntactic theory. From Government and Binding Theory, V-grammar takes the notion of

extended projection. From Head-Driven Phrase Structure Grammar, V-grammar takes the idea that much information should be carried by features. As in the Minimalist program, V-grammar derives phrase structure incrementally. These similarities are taken as a confirmation of the choice of formalism, since they show that certain key linguistic intuitions can be implemented in the chosen framework.

Finally, as an illustration of the advantages associated with using a formal framework for the expression of linguistic competence, we have discussed how the formal automaton associated with V-TAG can be used to model syntactic performance. This allows us to maintain a close relation between performance and competence models.

One issue has consistently been left aside throughout this thesis: the issue of pragmatic and semantic constraints on word order variation. It is clear that any full linguistic or formal discussion of free word order must take into account such constraints. This became particularly clear during the discussion of the competence model. The elaboration of an integrated theory of syntax and pragmatics in a formal framework awaits future research.

Bibliography

Abeillé, A. (1988). A lexicalized tree adjoining grammar for French: the general framework. Technical Report MS-CIS-88-64, University of Pennsylvania.

Abeillé, A., Bishop, K., Cote, S., and Schabes, Y. (1990). A lexicalized tree adjoining grammar for English. Technical Report MS-CIS-90-24, Department of Computer and Information Science, University of Pennsylvania.

Abney, S. (1986). Licensing and parsing. In *Proceedings of the 16th Annual Meeting of the North Eastern Linguistics Society*. Graduate Linguistics Students Association, University of Massachusetts.

Abraham, S. (1965). Some questions of phrase-structure grammars. *Computational Linguistics*, 4:61–70.

Aho, A. V. (1968). Indexed grammars – an extension to context free grammars. *J. ACM*, 15:647–671.

Aho, A. V. and Ullman, J. D. (1972). *The Theory of Parsing, Translation, and Compiling*. Prentice Hall, Englewood Cliffs, NJ.

Ajdukiewicz, K. (1935). Die syntaktische Konnexität. *Studia Philosophica*, 1:1–27. English translation in Storrs McCall (ed), *Polish Logic 1920-1939*, Oxford University Press, pp. 207-231.

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

Baker, M. (1985). The Mirror Principle and morphosyntactic explanation. *Linguistic Inquiry*, 16(3):373–415.

Baker, M. (1988). *Incorporation. A theory of grammatical function changing*. University of Chicago Press, Chicago.

Bar-Hillel, Y. (1953). A quasi-arithmetical notation for syntactic description. *Language*, 29.

Bayer, J. and Kornfilt, J. (1989). Restructuring effects in German. In *Parametric Variation in Germanic and Romance*. Centre for Cognitive Science, University of Edinburgh.

Bech, G. (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. 2nd unrevised edition published 1983 by Max Niemeyer Verlag, Tübingen (Linguistische Arbeiten 139).

Becker, T. (1990). Meta-rules on tree adjoining grammars. In *Proceedings of the 1st International Workshop on Tree Adjoining Grammars*, Schloß Dagstuhl.

Becker, T. (1993). *HyTAG: A new Type of Tree Adjoining Grammars for Hybrid Syntactic Representation of Free Word Order Languages*. PhD thesis, Universität des Saarlandes.

Becker, T., Joshi, A., and Rambow, O. (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. ACL.

Becker, T. and Rambow, O. (1990). Formal aspects of long distance scrambling. Unpublished Paper, University of Pennsylvania.

Becker, T. and Rambow, O. (1994). Parsing free word order in polynomial time.

Becker, T., Rambow, O., and Niv, M. (1992). The derivational generative power, or, scrambling is beyond LCFRS. Technical Report IRCS-92-38, Institute for Research in Cognitive Science, University of Pennsylvania. A version of this paper was presented at MOL3, Austin, Texas, November 1992.

Benveniste, É. (1966). *Problèmes de Linguistique Générale, 1*, chapter La Communication, pages 49–55. Gallimard, Paris.

Bhatt, R. and Yoon, J. (1991). On the composition of COMP and parameters of V2. In *Proceedings of WCCFL X*.

Borer, H. (1983). *Parametric syntax. Case studies in Semitic and Romance languages.* Studies in generative grammar 13. Foris, Dordrecht.

Brandt, M., Reis, M., Rosengren, I., and Zimmermann, I. (1992). Satztyp, Satzmodus und Illokution. In Rosengren, I., editor, *Satz und Illokution*, pages 1 – 90. Max Niemeyer Verlag, Tübingen. Series: Linguistische Arbeiten 278.

Bresnan, J., Kaplan, R., Peters, S., and Zaenen, A. (1983). Cross-serial dependencies in Dutch. *Linguistic Inquiry*, 13:613–635.

Chomsky, N. (1977). On *wh*-movement. In Culicover, P. W., Wasow, T., and Akmajian, A., editors, *Formal syntax*, pages 71–132. Academic Press, New York.

Chomsky, N. (1981). *Lectures in Government and Binding.* Studies in generative grammar 9. Foris, Dordrecht.

Chomsky, N. (1986a). *Barriers.* MIT Press, Cambridge, Mass.

Chomsky, N. (1986b). *Knowledge of Language.* Praeger Publishers, New York.

Chomsky, N. (1992). A minimalist program for linguistic theory. MIT Occasional Papers in Linguistics 1.

Comrie, B. (1973). Clause structure and movement constraints in Russian. In *Papers from the Ninth Regional Meeting*, pages 291–304, Chicago, Ill. Chicago Linguistics Society, University of Chicago.

Covington, M. (1990). Parsing discontinuous constituents in dependency grammar. *Computational Linguistics*, 16(4):234–236.

Cremers, A. B. and Mayer, O. (1973). On matrix languages. *Information and Control*, 23:86–96.

Cremers, A. B. and Mayer, O. (1974). On vector languages. *J. Comput. Syst. Sci.*, 8:158–166.

Culicover, P. (1993). Topicalization, inversion, and complementizers in English. Ms., The Ohio State University.

Dahlhaus, E. and Warmuth, M. K. (1986). Membership for growing context-sensitive grammars is polynomial. *J. Comput. Syst. Sci.*, 33:456–472.

Dassow, J. and Păun, G. (1989). *Regulated Rewriting in Formal Language Theory.* Springer Verlag, Berlin, Heidelberg, New York.

de Smedt, K. (1990). IPF: An incremental parallel formulator. In Dale, R., Mellish, C., and Zock, M., editors, *Current Research in Natural Langugae Generation.* Academic Press, London.

den Besten, H. (1983). On the interaction of root transformations and lexical deletive rules. In Abraham, W., editor, *On the formal syntax of the Westgermania*, pages 47–131. John Benjamins, Amsterdam. Draft version circulated in 1977.

den Besten, H. and Edmondson, J. A. (1983). The verbal complex in continental west germanic. In Abraham, W., editor, *On the formal syntax of the Westgermania*, pages 155–216. John Benjamins, Amsterdam.

den Besten, H. and Rutten, J. (1989). On verb raising, extraposition and free word order in dutch. In Jaspers, D., editor, *Sentential complementation and the lexicon*, pages 41–56. Foris, Dordrecht.

Diesing, M. (1988). Word order and the subject position in yiddish. In Blevins, J. and Carter, J., editors, *Proceedings of NELS 18*, pages 124–140, Amherst, MA. GSLA.

Diesing, M. (1990). Verb movement and the subject position in yiddish. *Natural Language and Linguistic Theory*, 10:41–79.

Diesing, M. (1992). *Indefinites.* MIT Press, Cambridge, Mass.

Drach, E. (1937). *Grundgedanken der deutschen Satzlehre.* —, Frankfurt am Main.

Emonds, J. E. (1978). The verbal complex V′–V in French. *Linguistic Inquiry*, 9:151–175.

Engelfriet, J. and Heyker, L. (1991). The string generating power of context-free hypergraph grammars. *J. Comput. Syst. Sci.*, 43:328–360.

Evers, A. (1975). *The Transformational Cycle in Dutch and German.* PhD thesis, University of Utrecht. Distributed by the Indiana University Linguistics Club.

Frank, R. (1992). *Syntactic Locality and Tree Adjoining Grammar: Grammatical, Acquisition and Processing Perspectives.* PhD thesis, Department of Computer and Information Science, University of Pennsylvania.

Frank, R., Lee, Y.-S., and Rambow, O. (1992). Scrambling as non-operator movement and the special status of subjects. In Barbiers, S., den Dikken, M., and Levelt, C., editors, *Proceedings of the Third Leiden Conference for Junior Linguists*, pages 135–154, Leiden.

Fukui, N. and Speas, M. (1986). Specifiers and projection. In Fukui, N., Rappaport, T., and Sagey, E., editors, *MIT Working Papers in Linguistics 8*. MIT Department of Linguistics.

Gaifman, H. (1965). Dependency systems and phrase-structure systems. *Information and Control*, 8:304–337.

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

Gazdar, G., Klein, E., Pullum, G., and Sag, I. (1985). *Generalized Phrase Structure Grammar*. Harvard University Press, Cambridge, Mass.

Gibson, E. (1991). *A Computational Theory of Human Linguistic Processing: Memory Limitations and Processing Breakdown*. PhD thesis, Carnegie Mellon University.

Grewendorf, G. (1988). *Aspekte der deutschen Syntax*. Studien zur deutschen Grammatik 33. Gunter Narr Verlag, Tübingen.

Grimshaw, J. (1990). *Argument Structure*. MIT Press, Cambridge, Mass.

Grimshaw, J. (1991). Extended projection. Manuscript, Brandeis University.

Grimshaw, J. (1993). Minimal projections. Talk presented at the University of Pennsylvania, March 4.

Haegeman, L. and van Riemsdijk, H. (1986). Verb projection raising scope and the typology of rules affecting verbs. *Linguistic Inquiry*, 17:417–466.

Haider, H. (1988). Matching projections. In *Annali di ca' foscari*, pages 101–121. Rivista della facoltà di lingue e letterature straniere dell'università di Venezia, Venice.

Haider, H. (1991). Argument structure: Semantic basis and syntactic effects. Class notes from the Third European Summer School in Language, Logic and Information, Saarbrücken.

Harbusch, K. (1990). Constraining tree adjoining grammars by unification. In *Proceedings of the 13th International Conference on Computational Linguistics (COLING'90)*, Helsinki.

Harbusch, K., Finkler, W., and Schauder, A. (1991). Incremental syntax generation with tree adjoining grammars. In *Proceedings 4.Int. GI-Kongress Wissensbasierte Systeme*, München. GWAI.

Hegarty, M. (1993a). Deriving clausal structure in Tree Adjoining Grammar. Unpublished Paper.

Hegarty, M. (1993b). *Wh* fronting and the composition of phrase structure in Tree Adjoining Grammar. Unpublished Paper.

Heycock, C. (1987). The structure of the japanese causative. Technical report MS-CIS-87-55, Department of Computer and Information Science, University of Pennsylvania.

Heycock, C. (1991). *Layers of Predication: the Non-lexical Syntax of Clauses*. PhD thesis, University of Pennsylvania.

Heycock, C. and Kroch, A. S. (1992). Verb movement and coordination in the Germanic languages. Ms., Yale University and University of Pennsylvania. Paper presented at the Workshop on Germanic Syntax, Tromsø.

Heycock, C. and Kroch, A. S. (1993). Verb movement and the status of subjects: implications for the theory of licensing. *Groninger Arbeiten zur germanistischen Linguistik*, 36:75–102.

Heycock, C. and Santorini, B. (1992). Head movement and the licensing of nonthematic positions. Ms., Yale University and Northwestern University. To appear in the Proceedings of WCCFL XI.

Hoffman, B. (1992). A CCG approach to free word order languages. In *30th Meeting of the Association for Computational Linguistics (ACL'92)*.

Hoffman, B. (1994). Generating context-appropriate word orders in turkish. Submitted to the *International Generation Workshop*.

Höhle, T. (1982). Explikation für 'normale Betonung' und 'normale Wortstellung'. In Abraham, W., editor, *Satzglieder im Deutschen*, pages 75–153. Gunter Narr Verlag, Tübingen.

Höhle, T. (1983). Subjektlücken in Koordinationen. Ms.

Hopcroft, J. and Ullman, J. (1979). *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, Reading, MA.

Huang, J. (1982). *Logical Relations in Chinese and the Theory of Grammar*. PhD thesis, Massachusetts Institute of Technology.

Hudson, R. (1984). *Word Grammar*. Blackwell, Oxford.

Iatridou, S. (1990). About Agr(P). *Linguistic Inquiry*, 21(4):551–577.

Inoue, A. and Fodor, J. D. (1993). Information-paced parsing of Japanese. In Mazuka, R. and Nagai, N., editors, *Japanese Syntactic Processing*. Lawrence Erlbaum Associates.

Joseph, B. D. and Smirniotopoulos, J. C. (1993). The morphosyntax of the Modern Greek verb as morphology and not syntax. *Linguistic Inquiry*, 24(2):388–398.

Joshi, A., Levy, L., and Takahashi, M. (1975). Tree adjunct grammars. *J. Comput. Syst. Sci.*, 10:136–163.

Joshi, A. K. (1985). How much context-sensitivity is necessary for characterizing structural descriptions — Tree Adjoining Grammars. In Dowty, D., Karttunen, L., and Zwicky, A., editors, *Natural Language Processing — Theoretical, Computational and Psychological Perspective*, pages 206–250. Cambridge University Press, New York, NY. Originally presented in 1983.

Joshi, A. K. (1987a). An introduction to Tree Adjoining Grammars. In Manaster-Ramer, A., editor, *Mathematics of Language*, pages 87–115. John Benjamins, Amsterdam.

Joshi, A. K. (1987b). Word-order variation in natural language generation. Technical report, Department of Computer and Information Science, University of Pennsylvania.

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

Joshi, A. K., Vijay-Shanker, K., and Weir, D. (1991). The convergence of mildly context-sensitive grammatical formalisms. In Sells, P., Shieber, S., and Wasow, T., editors, *Foundational Issues in Natural Language Processing*, pages 31–81. MIT Press, Cambridge, Mass.

Kaplan, R. M. and Bresnan, J. W. (1981). Lexical-functional grammar: A formal system for grammatical rep resentation. In Bresnan, J. W., editor, *The Mental Representation of Grammatical Relations*. MIT Press, Cambridge, MA.

Kaplan, R. M. and Zaenen, A. (1989). Long distance dependencies, constituent structure, and functional uncertainity. In Baltin, M. and Kroch, A., editors, *Alternative Conceptions of Phrase Structure*. University of Chicago Press, Chicago. IL.

Karttunen, L. (1989). Radical lexicalism. In Baltin, M. and Kroch, A. S., editors, *Alternative conceptions of phrase structure*, pages 43–65. University of Chicago Press, Chicago.

Kasami, T. (1965). An efficient recognition and syntax algorithm for context-free languages. Scientific Report AFCRL-65-758, AFCRL, Bedford MA.

Kasper, R. (1992). Compiling head-driven phrase structure grammar into lexicalized tree adjoining grammar. Presented at the TAG+ Workshop, University of Pennsylvania.

Koster, J. (1975). Dutch as an SOV language. *Linguistic analysis*, 1:111–136.

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

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

Kroch, A. and Joshi, A. K. (1985). The linguistic relevance of tree adjoining grammar. Technical Report MS-CS-85-16, Department of Computer and Information Sciences, University of Pennsylvania.

Kroch, A. and Joshi, A. K. (1987). Analyzing extraposition in a tree adjoining grammar. In Huck, G. and Ojeda, A., editors, *Syntax and Semantics: Discontinuous Constituents*. Academic Press, New York, NY.

Kroch, A. and Santorini, B. (1991). The derived constituent structure of the West Germanic Verb Raising construction. In Freidin, R., editor, *Principles and parameters in comparative grammar*, pages 269–338. MIT Press, Cambridge, Mass.

Larson, R. (1988). On the double object construction. *Linguistic Inquiry*, 19(3):335–392.

Lee, Y.-S. (1991). Scrambling and the Adjoined Argument Hypothesis. Thesis Proposal, University of Pennsylvania.

Lee, Y.-S. (1993). *Scrambling as case-driven obligatory movement.* PhD thesis, University of Pennsylvania. Available as Technical Report 93–06 from the Institute for Research in Cognitive Science at the University of Pennsylvania.

Lenerz, J. (1977). *Zur Abfolge Nominaler Satzglieder im Deutschen.* Gunter Narr Verlag, Tübingen.

Mahajan, A. (1989). On the A/A-bar distinction: scrambling and weak crossover and binding in Hindi. Ms., M.I.T.

Marcus, M. (1980). *A Theory of Syntactic Recognition for Natural Language.* MIT Press, Cambridge, Mass.

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

Milgram, D. and Rosenfeld, A. (1971). A note on scattered context grammars. *Inform. Proc. Letters*, 1:47–50.

Miller, P. (1991). Scandinavian extraction phenomena revisited: Weak and strong generative capacity. *Linguistics and Philosophy*, 14.

Moltmann, F. (1990). Scrambling in German and the specificity effect. Unpublished Paper, MIT.

Müller, G. and Sternefeld, W. (1993). Improper movement and unambiguous binding. *Linguistic Inquiry*, 24(3):461–507.

Pesetsky, D. (1982). *Paths and Categories.* PhD thesis, Massachusetts Institute of Technology.

Peters, S. and Ritchie, R. W. (1973). On the generative power of transformational grammars. *Inf. Sci.*, 6.

Pollard, C. and Sag, I. (1987). *Information-Based Syntax and Semantics. Vol 1: Fundamentals.* CSLI.

Pollard, C. and Sag, I. (1991). Information-based syntax and semantics. vol 2: Agreement, binding and control. Draft distributed at the Third European Summer School in Language, Logic and Information, Saarbrücken.

Pollard, C. and Sag, I. (1994). *Head-Driven Phrase Structure Grammar.* University of Chicago Press, Chicago. Draft distributed at the Third European Summer School in Language, Logic and Information, Saarbrücken, 1991.

Pollock, J.-Y. (1989). Verb movement, Universal Grammar, and the structure of IP. *Linguistic Inquiry*, 20.

Pritchett, B. (1991). Head positions and parsing ambiguity. *Journal of Psycholinguistic Research*, 20(3):251–270.

Rambow, O. (1992a). Immediate dominance, linear precedence, and the representation of syntax. Technical Report IRCS-92-42, Institute for Research in Cognitive Science, University of Pennsylvania. A version of this paper was presented at the TAG+ Workshop, Philadelphia, 1992.

Rambow, O. (1992b). Natural language syntax and formal systems. Dissertation Proposal, University of Pennsylvania. Available as Technical Report 93-18 from the Institute for Research in Cognitive Science (IRCS) at the University of Pennsylvania.

Rambow, O. (1993). Mobile heads and strict lexicalization. Unpublished Manuscript, University of Pennsylvania.

Rambow, O. (1994). Mobile heads and strict lexicalization. Master's thesis, Department of Linguistics, University of Pennsylvania, Philadelphia.

Rambow, O. and Joshi, A. (1994). A formal look at dependency grammars and phrase-structure grammars, with special consideration of word-order phenomena. In Wanner, L., editor, *Current Issues in Meaning-Text Theory*. Pinter, London. To appear.

Rambow, O. and Lee, Y.-S. (1994). Word order variation and Tree Adjoining Grammar. To appear in *Computational Intelligence*.

Rambow, O. and Satta, G. (1992). Formal properties of non-locality. Paper Presented at the TAG+ Workshop.

Rambow, O. and Satta, G. (1994a). A rewriting system for free word order syntax that is non-local *and* mildly context sensitive. In Martín-Vide, C., editor, *Current Issues in Mathematical Linguistics*, North-Holland Linguistic series, Volume 56. Elsevier-North Holland, Amsterdam.

Rambow, O. and Satta, G. (1994b). A two-dimensional hierarchy for parallel rewriting systems. Technical Report IRCS-94-03, Institute for Research in Cognitive Science, University of Pennsylvania.

Reape, M. (1990). Getting things in order. In *Proceedings of the Symposium on Discontinuous Constituents*, Tilburg, Holland.

Rizzi, L. (1990). Speculations on verb second. In Mascaró, J. and Nespor, M., editors, *Grammar in Progress*, pages 375–386. Foris, Dordrecht.

Ross, J. R. (1967). *Constraints on variables in syntax*. PhD thesis, Massachusetts Institute of Technology.

Rothstein, S. (1983). *The Syntactic Forms of Predication*. PhD thesis, MIT. Distributed by the Indiana University Linguistics Club.

Saito, M. (1992). Long-distance scrambling in Japanese. *Journal of East Asian Linguistics*, 1(1):69–118.

Salomaa, A. (1973). *Formal Languages*. Academic Press, Orlando, Forida.

Santorini, B. (1989). *The generalization of the verb-second constraint in the history of Yiddish.* PhD thesis, University of Pennsylvania.

Santorini, B. (1990). Scrambling and INFL in German. Paper presented at CUNY; Manuscript, University of Pennsylvania.

Santorini, B. and Kroch, A. (1990). Remnant extraposition in German. Unpublished Paper, University of Pennsylvania.

Satta, G. (1993). Recognition of vector languages. Unpublished manuscript, Università di Venezia.

Schabes, Y. (1990). *Mathematical and Computational Aspects of Lexicalized Grammars.* PhD thesis, Department of Computer and Information Science, University of Pennsylvania.

Schabes, Y. and Shieber, S. (1990). Synchronous tree adjoining grammars. In *Proceedings of the 13th International Conference on Computational Linguistics*, Helsinki.

Schabes, Y. and Vijay-Shanker, K. (1990). Deterministic left to right parsing of tree adjoining languages. In *28th Meeting of the Association for Computational Linguistics (ACL'90)*, Pittsburgh.

Seki, H., Matsumura, T., Fujii, M., and Kasami, T. (1991). On multiple context-free grammars. *Theoretical Computer Science*, 88:191–229.

Sgall, P., Hajicova, E., and Panevova, J. (1986). *The meaning of the sentence and its semantic and pragmatic aspects.* Reidel, Dordrecht.

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

Stabler, E. (1992). *The Logical Approach to Syntax.* ACL-MIT Press series in natural language processing. MIT Press, Cambridge, Mass.

Steedman, M. (1985). Dependency and coordination in the grammar of Dutch and English. *Language*, 61.

Steedman, M. (1991). Structure and Intonation. *Language*, 68(2):260–296.

Sudborough, I. H. (1977). The complexity of the membership problem for some extensions of context-free languages. *International J. Computer Math.*, 6:191–215.

Tesnière, L. (1959). *Eléments de syntaxe structurale.* Klincksieck, Paris.

Thiersch, C. (1978). *Topics in German syntax.* PhD thesis, MIT.

Uszkoreit, H. (1987). *Word Order and Constituent Structure in German.* CSLI, Stanford, CA.

Vallduvi, E. (1992). *The Informational Component.* Garland, New York.

Vijay-Shanker, K. (1987). *A study of Tree Adjoining Grammars.* PhD thesis, Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA.

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

Vijay-Shanker, K. and Joshi, A. K. (1988). Feature structure based tree adjoining grammars. In *Proceedings of the 12th International Conference on Computational Linguistics (COLING'88)*, pages 714–719.

Vijay-Shanker, K. and Schabes, Y. (1992). Structure sharing in lexicalized tree-adjoining grammars. In *Proceedings of the 14th International Conference on Computational Linguistics (COLING'92)*, pages 205–211.

Vijay-Shanker, K. and Weir, D. (1992). The equivalence of four extensions of context-free grammars. Technical Report CSRP 236, School of Cognitive and Computing Sciences, University of Sussex. To appear in *Math. Syst. Theory*.

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

von Stechow, A. and Sternefeld, W. (1988). *Bausteine syntaktischen Wissens*. Westdeutscher Verlag.

Watanabe, A. (1993). The notion of finite clauses in agr-based case theory. In *MIT Working Papers in Linguistics 18*, pages 281–306. MIT Press.

Webelhuth, G. (1989). *Syntactic saturation phenomena and the modern Germanic languages*. PhD thesis, University of Massachussetts.

Weir, D. J. (1988). *Characterizing Mildly Context-Sensitive Grammar Formalisms*. PhD thesis, Department of Computer and Information Science, University of Pennsylvania.

Weir, D. J. (1992). Linear context-free rewriting systems and deterministic tree-walk transducers. In *30th Meeting of the Association for Computational Linguistics (ACL'92)*.

Williams, E. (1980). Predication. *Linguistic Inquiry*, 11.

Younger, D. H. (1967). Recognition and parsing of context-free languages in time $O(n^3)$. *Information and Control*, 10(2):189–208.

Zaenen, A. (1979). Infinitival complements in dutch. In *Papers from the 15th regional meeting of the Chicago Linguistic Society*, pages 378–389.

Appendix A

Runs of the {}-BEPDA

This appendix shows the run of the {}-BEPDA derived from the simple scrambling grammar discussed in Section 6.3, page 214. These runs have been generated by a LISP implementation of the automaton. Section A.1 shows center-embedding constructiosn at various depths, while Section A.2 shows the runs for 30 sentences involving scrambling and extraposition.

A.1 Center Embedding

Depth of embedding: 1

N_1 V_1

Step	Store	Metric 1 New	Metric 1 Cum	Metric 2 New	Metric 2 Cum
1a	[N_1	1	1	1	1
2a	[N_1 [V_1 {N_1 }	2	3	2	3
2b	[V_1 {}		3	2	5
2c			3	0	5

String accepted. Scores: 3 by Metric 1, 5 by Metric 2.

Depth of embedding: 2

N_1 N_2 V_2 V_1

Step	Store	Metric 1 New	Metric 1 Cum	Metric 2 New	Metric 2 Cum
1a	[N_1	1	1	1	1
2a	[N_1 [N_2	2	3	2	3
3a	[N_1 [N_2 [V_2 {N_2 }	3	6	3	6
3b	[N_1 [V_2 {}		6	3	9
4a	[N_1 [V_2 {} [V_1 {N_1 V_2 }	4	10	4	13
4b	[N_1 [V_2 {} V_1 {N_1 }		10	4	17
4c	[V_2 {} V_1 {}		10	4	21
4d	[V_2 {}		10	2	23
4e			10	0	23

String accepted. Scores: 10 by Metric 1, 23 by Metric 2.

Depth of embedding: 3

$N_1\ N_2\ N_3\ V_3\ V_2\ V_1$

Step	Store	Metric 1 New	Metric 1 Cum	Metric 2 New	Metric 2 Cum
1a	[N_1	1	1	1	1
2a	[N_1 [N_2	2	3	2	3
3a	[N_1 [N_2 [N_3	3	6	3	6
4a	[N_1 [N_2 [N_3 [V_3 {N_3 }	4	10	4	10
4b	[N_1 [N_2 [V_3 {}		10	4	14
5a	[N_1 [N_2 [V_3 {} [V_2 {N_2 V_3 }	5	15	5	19
5b	[N_1 [N_2 [V_3 {} V_2 {N_2 }		15	5	24
5c	[N_1 [V_3 {} V_2 {}		15	5	29
6a	[N_1 [V_3 {} V_2 {} [V_1 {N_1 V_2 }	6	21	6	35
6b	[N_1 [V_3 {} V_2 {} V_1 {N_1 }		21	6	41
6c	[V_3 {} V_2 {} V_1 {}		21	6	47
6d	[V_3 {} V_2 {}		21	4	51
6e	[V_3 {}		21	2	53
6f			21	0	53

String accepted. Scores: 21 by Metric 1, 53 by Metric 2.

Depth of embedding: 4

$N_1\ N_2\ N_3\ N_4\ V_4\ V_3\ V_2\ V_1$

Step	Store	Metric 1 New	Metric 1 Cum	Metric 2 New	Metric 2 Cum
1a	[N_1	1	1	1	1
2a	[N_1 [N_2	2	3	2	3
3a	[N_1 [N_2 [N_3	3	6	3	6
4a	[N_1 [N_2 [N_3 [N_4	4	10	4	10
5a	[N_1 [N_2 [N_3 [N_4 [V_4 {N_4 }	5	15	5	15
5b	[N_1 [N_2 [N_3 [V_4 {}		15	5	20
6a	[N_1 [N_2 [N_3 [V_4 {} [V_3 {N_3 V_4 }	6	21	6	26
6b	[N_1 [N_2 [N_3 [V_4 {} V_3 {N_3 }		21	6	32
6c	[N_1 [N_2 [V_4 {} V_3 {}		21	6	38
7a	[N_1 [N_2 [V_4 {} V_3 {} [V_2 {N_2 V_3 }	7	28	7	45
7b	[N_1 [N_2 [V_4 {} V_3 {} V_2 {N_2 }		28	7	52
7c	[N_1 [V_4 {} V_3 {} V_2 {}		28	7	59
8a	[N_1 [V_4 {} V_3 {} V_2 {} [V_1 {N_1 V_2 }	8	36	8	67
8b	[N_1 [V_4 {} V_3 {} V_2 {} V_1 {N_1 }		36	8	75
8c	[V_4 {} V_3 {} V_2 {} V_1 {}		36	8	83
8d	[V_4 {} V_3 {} V_2 {}		36	6	89
8e	[V_4 {} V_3 {}		36	4	93
8f	[V_4 {}		36	2	95
8g			36	0	95

String accepted. Scores: 36 by Metric 1, 95 by Metric 2.

A.2 Scrambling and Extraposition

Sentence (i): $Comp_1\ N_1\ N_3\ V_3\ V_2\ V_1$ Judgment: ok

Step	Store	Metric 1 New	Metric 1 Cum	Metric2 New	Metric2 Cum
1a	[$Comp_1$	1	1	1	1
2a	[$Comp_1$ [N_1	2	3	2	3
3a	[$Comp_1$ [N_1 [N_3	3	6	3	6
4a	[$Comp_1$ [N_1 [N_3 [V_3 {N_3}	4	10	4	10
4b	[$Comp_1$ [N_1 [V_3 {}		10	4	14
5a	[$Comp_1$ [N_1 [V_3 {} [V_2 {V_3}	5	15	5	19
5b	[$Comp_1$ [N_1 [V_3 {} V_2 {}		15	5	24
6a	[$Comp_1$ [N_1 [V_3 {} V_2 {} [V_1 {N_1 V_2}	6	21	6	30
6b	[$Comp_1$ [N_1 [V_3 {} V_2 {} V_1 {N_1}		21	6	36
6c	[$Comp_1$ [V_3 {} V_2 {} V_1 {}		21	6	42
6d	[V_3 {} V_2 {} V_1 {}		21	5	47
6e	[V_3 {} V_2 {}		21	3	50
6f	[V_3 {}		21	2	52
6g			21	0	52

String accepted. Scores: 21 by Metric 1, 52 by Metric 2.

Sentence (ii): $Comp_1\ N_3\ N_1\ V_3\ V_2\ V_1$ Judgment: ok

Step	Store	Metric 1 New	Metric 1 Cum	Metric2 New	Metric2 Cum
1a	[$Comp_1$	1	1	1	1
2a	[$Comp_1$ [N_3	2	3	2	3
3a	[$Comp_1$ [N_3 [N_1	3	6	3	6
4a	[$Comp_1$ [N_3 [N_1 [V_3 {N_3}	4	10	4	10
5a	[$Comp_1$ [N_3 [N_1 [V_3 {N_3} [V_2 {V_3}	5	15	5	15
5b	[$Comp_1$ [N_3 [N_1 [V_3 {N_3} V_2 {}		15	5	20
6a	[$Comp_1$ [N_3 [N_1 [V_3 {N_3} V_2 {} [V_1 {N_1 V_2}	6	21	6	26
6b	[$Comp_1$ [N_3 [N_1 [V_3 {N_3} V_2 {} V_1 {N_1}		21	6	32
6c	[$Comp_1$ [N_3 [V_3 {N_3} V_2 {} V_1 {}		21	6	38
6d	[$Comp_1$ [N_3 [V_3 {} V_2 {N_3} V_1 {}	6	27	6	44
6e	[$Comp_1$ [N_3 [V_3 {} V_2 {} V_1 {N_3}	6	33	6	50
6f	[$Comp_1$ [V_3 {} V_2 {} V_1 {}		33	6	56
6g	[V_3 {} V_2 {} V_1 {}		33	5	61
6h	[V_3 {} V_2 {}		33	3	64
6i	[V_3 {}		33	2	66
6j			33	0	66

String accepted. Scores: 33 by Metric 1, 66 by Metric 2.

Sentence (iii): $\mathsf{Comp}_1\ \mathsf{N}_1\ \mathsf{V}_2\ \mathsf{N}_3\ \mathsf{V}_3\ \mathsf{V}_1$ Judgment: ?

Step	Store	Metric 1 New	Metric 1 Cum	Metric2 New	Metric2 Cum
1a	$[\mathsf{Comp}_1$	1	1	1	1
2a	$[\mathsf{Comp}_1\ [\mathsf{N}_1$	2	3	2	3
3a	$[\mathsf{Comp}_1\ [\mathsf{N}_1\ [\mathsf{V}_2\ \{\mathsf{V}_3\}$	3	6	3	6
4a	$[\mathsf{Comp}_1\ [\mathsf{N}_1\ [\mathsf{V}_2\ \{\mathsf{V}_3\}\ [\mathsf{N}_3$	4	10	4	10
5a	$[\mathsf{Comp}_1\ [\mathsf{N}_1\ [\mathsf{V}_2\ \{\mathsf{V}_3\}\ [\mathsf{N}_3\ [\mathsf{V}_3\ \{\mathsf{N}_3\}$	5	15	5	15
5b	$[\mathsf{Comp}_1\ [\mathsf{N}_1\ [\mathsf{V}_2\ \{\mathsf{V}_3\}\ [\mathsf{V}_3\ \{\}$		15	5	20
5c	$[\mathsf{Comp}_1\ [\mathsf{N}_1\ [\mathsf{V}_3\ \{\}\ \mathsf{V}_2\ \{\}$		15	5	25
6a	$[\mathsf{Comp}_1\ [\mathsf{N}_1\ [\mathsf{V}_3\ \{\}\ \mathsf{V}_2\ \{\}\ [\mathsf{V}_1\ \{\mathsf{N}_1\ \mathsf{V}_2\}$	6	21	6	31
6b	$[\mathsf{Comp}_1\ [\mathsf{N}_1\ [\mathsf{V}_3\ \{\}\ \mathsf{V}_2\ \{\}\ \mathsf{V}_1\ \{\mathsf{N}_1\}$		21	6	37
6c	$[\mathsf{Comp}_1\ [\mathsf{V}_3\ \{\}\ \mathsf{V}_2\ \{\}\ \mathsf{V}_1\ \{\}$		21	6	43
6d	$[\mathsf{V}_3\ \{\}\ \mathsf{V}_2\ \{\}\ \mathsf{V}_1\ \{\}$		21	5	48
6e	$[\mathsf{V}_3\ \{\}\ \mathsf{V}_2\ \{\}$		21	3	51
6f	$[\mathsf{V}_3\ \{\}$		21	2	53
6g			21	0	53

String accepted. Scores: 21 by Metric 1, 53 by Metric 2.

Sentence (iv): $\mathsf{Comp}_1\ \mathsf{N}_1\ \mathsf{N}_3\ \mathsf{V}_2\ \mathsf{V}_3\ \mathsf{V}_1$ Judgment: ?

Step	Store	Metric 1 New	Metric 1 Cum	Metric2 New	Metric2 Cum
1a	$[\mathsf{Comp}_1$	1	1	1	1
2a	$[\mathsf{Comp}_1\ [\mathsf{N}_1$	2	3	2	3
3a	$[\mathsf{Comp}_1\ [\mathsf{N}_1\ [\mathsf{N}_3$	3	6	3	6
4a	$[\mathsf{Comp}_1\ [\mathsf{N}_1\ [\mathsf{N}_3\ [\mathsf{V}_2\ \{\mathsf{V}_3\}$	4	10	4	10
5a	$[\mathsf{Comp}_1\ [\mathsf{N}_1\ [\mathsf{N}_3\ [\mathsf{V}_2\ \{\mathsf{V}_3\}\ [\mathsf{V}_3\ \{\mathsf{N}_3\}$	5	15	5	15
5b	$[\mathsf{Comp}_1\ [\mathsf{N}_1\ [\mathsf{N}_3\ [\mathsf{V}_3\ \{\mathsf{N}_3\}\ \mathsf{V}_2\ \{\}$		15	5	20
5c	$[\mathsf{Comp}_1\ [\mathsf{N}_1\ [\mathsf{N}_3\ [\mathsf{V}_3\ \{\}\ \mathsf{V}_2\ \{\mathsf{N}_3\}$	5	20	5	25
5d	$[\mathsf{Comp}_1\ [\mathsf{N}_1\ [\mathsf{V}_3\ \{\}\ \mathsf{V}_2\ \{\}$		20	5	30
6a	$[\mathsf{Comp}_1\ [\mathsf{N}_1\ [\mathsf{V}_3\ \{\}\ \mathsf{V}_2\ \{\}\ [\mathsf{V}_1\ \{\mathsf{N}_1\ \mathsf{V}_2\}$	6	26	6	36
6b	$[\mathsf{Comp}_1\ [\mathsf{N}_1\ [\mathsf{V}_3\ \{\}\ \mathsf{V}_2\ \{\}\ \mathsf{V}_1\ \{\mathsf{N}_1\}$		26	6	42
6c	$[\mathsf{Comp}_1\ [\mathsf{V}_3\ \{\}\ \mathsf{V}_2\ \{\}\ \mathsf{V}_1\ \{\}$		26	6	48
6d	$[\mathsf{V}_3\ \{\}\ \mathsf{V}_2\ \{\}\ \mathsf{V}_1\ \{\}$		26	5	53
6e	$[\mathsf{V}_3\ \{\}\ \mathsf{V}_2\ \{\}$		26	3	56
6f	$[\mathsf{V}_3\ \{\}$		26	2	58
6g			26	0	58

String accepted. Scores: 26 by Metric 1, 58 by Metric 2.

Sentence (v): $Comp_1\ N_3\ N_1\ V_2\ V_3\ V_1$ Judgment: ?

Step	Store	Metric 1 New	Metric 1 Cum	Metric2 New	Metric2 Cum
1a	[$Comp_1$	1	1	1	1
2a	[$Comp_1$ [N_3	2	3	2	3
3a	[$Comp_1$ [N_3 [N_1	3	6	3	6
4a	[$Comp_1$ [N_3 [N_1 [V_2 {V_3}	4	10	4	10
5a	[$Comp_1$ [N_3 [N_1 [V_2 {V_3} [V_3 {N_3}	5	15	5	15
5b	[$Comp_1$ [N_3 [N_1 [V_3 {N_3} V_2 {}		15	5	20
6a	[$Comp_1$ [N_3 [N_1 [V_3 {N_3} V_2 {} [V_1 {N_1 V_2}	6	21	6	26
6b	[$Comp_1$ [N_3 [N_1 [V_3 {N_3} V_2 {} V_1 {N_1}		21	6	32
6c	[$Comp_1$ [N_3 [V_3 {N_3} V_2 {} V_1 {}		21	6	38
6d	[$Comp_1$ [N_3 [V_3 {} V_2 {N_3} V_1 {}	6	27	6	44
6e	[$Comp_1$ [N_3 [V_3 {} V_2 {} V_1 {N_3}	6	33	6	50
6f	[$Comp_1$ [V_3 {} V_2 {} V_1 {}		33	6	56
6g	[V_3 {} V_2 {} V_1 {}		33	5	61
6h	[V_3 {} V_2 {}		33	3	64
6i	[V_3 {}		33	2	66
6j			33	0	66

String accepted. Scores: 33 by Metric 1, 66 by Metric 2.

Sentence (vi): $Comp_1\ N_3\ V_3\ N_1\ V_2\ V_1$ Judgment: ok

Step	Store	Metric 1 New	Metric 1 Cum	Metric2 New	Metric2 Cum
1a	[$Comp_1$	1	1	1	1
2a	[$Comp_1$ [N_3	2	3	2	3
3a	[$Comp_1$ [N_3 [V_3 {N_3}	3	6	3	6
3b	[$Comp_1$ [V_3 {}		6	3	9
4a	[$Comp_1$ [V_3 {} [N_1	4	10	4	13
5a	[$Comp_1$ [V_3 {} [N_1 [V_2 {V_3}	5	15	5	18
6a	[$Comp_1$ [V_3 {} [N_1 [V_2 {V_3} [V_1 {N_1 V_2}	6	21	6	24
6b	[$Comp_1$ [V_3 {} [N_1 [V_2 {V_3} V_1 {N_1}		21	6	30
6c	[$Comp_1$ [V_3 {} [V_2 {V_3} V_1 {}		21	6	36
6d	[$Comp_1$ [V_3 {} [V_2 {} V_1 {V_3}	6	27	6	42
6e	[$Comp_1$ [V_2 {} V_1 {}		27	6	48
6f	[V_2 {} V_1 {}		27	5	53
6g	[V_2 {}		27	3	56
6h			27	0	56

String accepted. Scores: 27 by Metric 1, 56 by Metric 2.

Sentence (vii): $\text{Comp}_1\ \text{N}_3\ \text{V}_3\ \text{V}_2\ \text{N}_1\ \text{V}_1$ Judgment: ?

Step	Store	Metric 1 New	Metric 1 Cum	Metric2 New	Metric2 Cum
1a	[Comp_1	1	1	1	1
2a	[Comp_1 [N_3	2	3	2	3
3a	[Comp_1 [N_3 [V_3 {N_3}	3	6	3	6
3b	[Comp_1 [V_3 {}		6	3	9
4a	[Comp_1 [V_3 {} [V_2 {V_3}	4	10	4	13
4b	[Comp_1 [V_3 {} V_2 {}		10	4	17
5a	[Comp_1 [V_3 {} V_2 {} [N_1	5	15	5	22
6a	[Comp_1 [V_3 {} V_2 {} [N_1 [V_1 {N_1 V_2}	6	21	6	28
6b	[Comp_1 [V_3 {} V_2 {} [V_1 {V_2}		21	6	34
6c	[Comp_1 [V_3 {} V_2 {} V_1 {}		21	6	40
6d	[V_3 {} V_2 {} V_1 {}		21	5	45
6e	[V_3 {} V_2 {}		21	3	48
6f	[V_3 {}		21	2	50
6g			21	0	50

String accepted. Scores: 21 by Metric 1, 50 by Metric 2.

Sentence (viii): $\text{Comp}_1\ \text{V}_2\ \text{N}_3\ \text{V}_3\ \text{N}_1\ \text{V}_1$ Judgment: ?

Step	Store	Metric 1 New	Metric 1 Cum	Metric2 New	Metric2 Cum
1a	[Comp_1	1	1	1	1
2a	[Comp_1 [V_2 {V_3}	2	3	2	3
3a	[Comp_1 [V_2 {V_3} [N_3	3	6	3	6
4a	[Comp_1 [V_2 {V_3} [N_3 [V_3 {N_3}	4	10	4	10
4b	[Comp_1 [V_2 {V_3} [V_3 {}		10	4	14
4c	[Comp_1 [V_3 {} V_2 {}		10	4	18
5a	[Comp_1 [V_3 {} V_2 {} [N_1	5	15	5	23
6a	[Comp_1 [V_3 {} V_2 {} [N_1 [V_1 {N_1 V_2}	6	21	6	29
6b	[Comp_1 [V_3 {} V_2 {} [V_1 {V_2}		21	6	35
6c	[Comp_1 [V_3 {} V_2 {} V_1 {}		21	6	41
6d	[V_3 {} V_2 {} V_1 {}		21	5	46
6e	[V_3 {} V_2 {}		21	3	49
6f	[V_3 {}		21	2	51
6g			21	0	51

String accepted. Scores: 21 by Metric 1, 51 by Metric 2.

Sentence (ix): $\text{Comp}_1\ \text{V}_2\ \text{N}_3\ \text{N}_1\ \text{V}_3\ \text{V}_1$ Judgment: *

Step	Store	Metric 1 New	Metric 1 Cum	Metric2 New	Metric2 Cum
1a	[Comp_1	1	1	1	1
2a	[Comp_1 [V_2 {V_3}	2	3	2	3
3a	[Comp_1 [V_2 {V_3} [N_3	3	6	3	6
4a	[Comp_1 [V_2 {V_3} [N_3 [N_1	4	10	4	10
5a	[Comp_1 [V_2 {V_3} [N_3 [N_1 [V_3 {N_3}	5	15	5	15
6a	[Comp_1 [V_2 {V_3} [N_3 [N_1 [V_3 {N_3} [V_1 {N_1 V_2}	6	21	6	21

String rejected.

Sentence (x): $Comp_1\ N_3\ V_2\ N_1\ V_3\ V_1$ Judgment: *

Step	Store	Metric 1 New	Metric 1 Cum	Metric2 New	Metric2 Cum
1a	[$Comp_1$	1	1	1	1
2a	[$Comp_1$ [N_3	2	3	2	3
3a	[$Comp_1$ [N_3 [V_2 {V_3}	3	6	3	6
4a	[$Comp_1$ [N_3 [V_2 {V_3} [N_1	4	10	4	10
5a	[$Comp_1$ [N_3 [V_2 {V_3} [N_1 [V_3 {N_3}	5	15	5	15
6a	[$Comp_1$ [N_3 [V_2 {V_3} [N_1 [V_3 {N_3} [V_1 {N_1 V_2}	6	21	6	21

String rejected.

Sentence (xi): $Comp_1\ V_2\ N_1\ N_3\ V_3\ V_1$ Judgment: *

Step	Store	Metric 1 New	Metric 1 Cum	Metric2 New	Metric2 Cum
1a	[$Comp_1$	1	1	1	1
2a	[$Comp_1$ [V_2 {V_3}	2	3	2	3
3a	[$Comp_1$ [V_2 {V_3} [N_1	3	6	3	6
4a	[$Comp_1$ [V_2 {V_3} [N_1 [N_3	4	10	4	10
5a	[$Comp_1$ [V_2 {V_3} [N_1 [N_3 [V_3 {N_3}	5	15	5	15
5b	[$Comp_1$ [V_2 {V_3} [N_1 [V_3 {}		15	5	20
6a	[$Comp_1$ [V_2 {V_3} [N_1 [V_3 {} [V_1 {N_1 V_2}	6	21	6	26

String rejected.

Sentence (xii): $Comp_1\ N_3\ V_2\ V_3\ N_1\ V_1$ Judgment: ?

Step	Store	Metric 1 New	Metric 1 Cum	Metric2 New	Metric2 Cum
1a	[$Comp_1$	1	1	1	1
2a	[$Comp_1$ [N_3	2	3	2	3
3a	[$Comp_1$ [N_3 [V_2 {V_3}	3	6	3	6
4a	[$Comp_1$ [N_3 [V_2 {V_3} [V_3 {N_3}	4	10	4	10
4b	[$Comp_1$ [N_3 [V_3 {N_3} V_2 {}		10	4	14
4c	[$Comp_1$ [N_3 [V_3 {} V_2 {N_3}	4	14	4	18
4d	[$Comp_1$ [V_3 {} V_2 {}		14	4	22
5a	[$Comp_1$ [V_3 {} V_2 {} [N_1	5	19	5	27
6a	[$Comp_1$ [V_3 {} V_2 {} [N_1 [V_1 {N_1 V_2}	6	25	6	33
6b	[$Comp_1$ [V_3 {} V_2 {} [V_1 {V_2}		25	6	39
6c	[$Comp_1$ [V_3 {} V_2 {} V_1 {}		25	6	45
6d	[V_3 {} V_2 {} V_1 {}		25	5	50
6e	[V_3 {} V_2 {}		25	3	53
6f	[V_3 {}		25	2	55
6g			25	0	55

String accepted. Scores: 25 by Metric 1, 55 by Metric 2.

Sentence (xiii): $\mathsf{Comp}_1\ \mathsf{N}_1\ \mathsf{V}_1\ \mathsf{N}_3\ \mathsf{V}_3\ \mathsf{V}_2$ Judgment: ok

Step	Store	Metric 1 New	Metric 1 Cum	Metric2 New	Metric2 Cum
1a	[Comp_1	1	1	1	1
2a	[Comp_1 [N_1	2	3	2	3
3a	[Comp_1 [N_1 [V_1 {N_1 V_2}	3	6	3	6
3b	[Comp_1 [V_1 {V_2}		6	3	9
3c	[V_1 {V_2}		6	2	11
3d			6	0	11
4a	[N_3	1	7	1	12
5a	[N_3 [V_3 {N_3}	2	9	2	14
5b	[V_3 {}		9	2	16
6a	[V_3 {} [V_2 {V_3}	3	12	3	19
6b	[V_3 {} V_2 {}		12	3	22
6c	[V_3 {}		12	2	24
6d			12	0	24

String accepted. Scores: 12 by Metric 1, 24 by Metric 2.

Sentence (xiv): $\mathsf{Comp}_1\ \mathsf{N}_1\ \mathsf{N}_3\ \mathsf{V}_1\ \mathsf{V}_3\ \mathsf{V}_2$ Judgment: ok

Step	Store	Metric 1 New	Metric 1 Cum	Metric2 New	Metric2 Cum
1a	[Comp_1	1	1	1	1
2a	[Comp_1 [N_1	2	3	2	3
3a	[Comp_1 [N_1 [N_3	3	6	3	6
4a	[Comp_1 [N_1 [N_3 [V_1 {N_1 V_2}	4	10	4	10
5a	[Comp_1 [N_1 [N_3 [V_1 {N_1 V_2} [V_3 {N_3}	5	15	5	15
6a	[Comp_1 [N_1 [N_3 [V_1 {N_1 V_2} [V_3 {N_3} [V_2 {V_3}	6	21	6	21
6b	[Comp_1 [N_1 [N_3 [V_1 {N_1 V_2} [V_3 {N_3} V_2 {}		21	6	27
6c	[Comp_1 [N_1 [N_3 [V_3 {N_3} V_2 {} V_1 {N_1}		21	6	33
6d	[Comp_1 [N_1 [N_3 [V_3 {} V_2 {N_3} V_1 {N_1}	6	27	6	39
6e	[Comp_1 [N_1 [N_3 [V_3 {} V_2 {} V_1 {N_3 N_1}	6	33	6	45
6f	[Comp_1 [N_1 [V_3 {} V_2 {} V_1 {N_1}		33	6	51
6g	[Comp_1 [V_3 {} V_2 {} V_1 {}		33	6	57
6h	[V_3 {} V_2 {} V_1 {}		33	5	62
6i	[V_3 {} V_2 {}		33	3	65
6j	[V_3 {}		33	2	67
6x			33	0	67

String accepted. Scores: 33 by Metric 1, 67 by Metric 2.

Sentence (xv): $\mathsf{Comp}_1\ \mathsf{N}_3\ \mathsf{N}_1\ \mathsf{V}_1\ \mathsf{V}_3\ \mathsf{V}_2$ Judgment: ?

Step	Store	Metric 1 New	Metric 1 Cum	Metric2 New	Metric2 Cum
1a	$[\mathsf{Comp}_1$	1	1	1	1
2a	$[\mathsf{Comp}_1\ [\mathsf{N}_3$	2	3	2	3
3a	$[\mathsf{Comp}_1\ [\mathsf{N}_3\ [\mathsf{N}_1$	3	6	3	6
4a	$[\mathsf{Comp}_1\ [\mathsf{N}_3\ [\mathsf{N}_1\ [\mathsf{V}_1\ \{\mathsf{N}_1\ \mathsf{V}_2\}$	4	10	4	10
4b	$[\mathsf{Comp}_1\ [\mathsf{N}_3\ [\mathsf{V}_1\ \{\mathsf{V}_2\}$		10	4	14
5a	$[\mathsf{Comp}_1\ [\mathsf{N}_3\ [\mathsf{V}_1\ \{\mathsf{V}_2\}\ [\mathsf{V}_3\ \{\mathsf{N}_3\}$	5	15	5	19
6a	$[\mathsf{Comp}_1\ [\mathsf{N}_3\ [\mathsf{V}_1\ \{\mathsf{V}_2\}\ [\mathsf{V}_3\ \{\mathsf{N}_3\}\ [\mathsf{V}_2\ \{\mathsf{V}_3\}$	6	21	6	25
6b	$[\mathsf{Comp}_1\ [\mathsf{N}_3\ [\mathsf{V}_1\ \{\mathsf{V}_2\}\ [\mathsf{V}_3\ \{\mathsf{N}_3\}\ \mathsf{V}_2\ \{\}$		21	6	31
6c	$[\mathsf{Comp}_1\ [\mathsf{N}_3\ [\mathsf{V}_3\ \{\mathsf{N}_3\}\ \mathsf{V}_2\ \{\}\ \mathsf{V}_1\ \{\}$		21	6	37
6d	$[\mathsf{Comp}_1\ [\mathsf{N}_3\ [\mathsf{V}_3\ \{\}\ \mathsf{V}_2\ \{\mathsf{N}_3\}\ \mathsf{V}_1\ \{\}$	6	27	6	43
6e	$[\mathsf{Comp}_1\ [\mathsf{N}_3\ [\mathsf{V}_3\ \{\}\ \mathsf{V}_2\ \{\}\ \mathsf{V}_1\ \{\mathsf{N}_3\}$	6	33	6	49
6f	$[\mathsf{Comp}_1\ [\mathsf{V}_3\ \{\}\ \mathsf{V}_2\ \{\}\ \mathsf{V}_1\ \{\}$		33	6	55
6g	$[\mathsf{V}_3\ \{\}\ \mathsf{V}_2\ \{\}\ \mathsf{V}_1\ \{\}$		33	5	60
6h	$[\mathsf{V}_3\ \{\}\ \mathsf{V}_2\ \{\}$		33	3	63
6i	$[\mathsf{V}_3\ \{\}$		33	2	65
6j			33	0	65

String accepted. Scores: 33 by Metric 1, 65 by Metric 2.

Sentence (xvi): $\mathsf{Comp}_1\ \mathsf{N}_1\ \mathsf{V}_1\ \mathsf{V}_2\ \mathsf{N}_3\ \mathsf{V}_3$ Judgment: ok

Step	Store	Metric 1 New	Metric 1 Cum	Metric2 New	Metric2 Cum
1a	$[\mathsf{Comp}_1$	1	1	1	1
2a	$[\mathsf{Comp}_1\ [\mathsf{N}_1$	2	3	2	3
3a	$[\mathsf{Comp}_1\ [\mathsf{N}_1\ [\mathsf{V}_1\ \{\mathsf{N}_1\ \mathsf{V}_2\}$	3	6	3	6
3b	$[\mathsf{Comp}_1\ [\mathsf{V}_1\ \{\mathsf{V}_2\}$		6	3	9
3c	$[\mathsf{V}_1\ \{\mathsf{V}_2\}$		6	2	11
3d			6	0	11
4a	$[\mathsf{V}_2\ \{\mathsf{V}_3\}$	1	7	1	12
4b			7	0	12
5a	$[\mathsf{N}_3$	1	8	1	13
6a	$[\mathsf{N}_3\ [\mathsf{V}_3\ \{\mathsf{N}_3\}$	2	10	2	15
6b	$[\mathsf{V}_3\ \{\}$		10	2	17
6c			10	0	17

String accepted. Scores: 10 by Metric 1, 17 by Metric 2.

Sentence (xvii): $Comp_1\ N_1\ V_1\ N_3\ V_2\ V_3$ Judgment: ?

Step	Store	Metric 1 New	Metric 1 Cum	Metric2 New	Metric2 Cum
1a	[$Comp_1$	1	1	1	1
2a	[$Comp_1$ [N_1	2	3	2	3
3a	[$Comp_1$ [N_1 [V_1 {N_1 V_2}	3	6	3	6
3b	[$Comp_1$ [V_1 {V_2}		6	3	9
3c	[V_1 {V_2}		6	2	11
3d			6	0	11
4a	[N_3	1	7	1	12
5a	[N_3 [V_2 {V_3}	2	9	2	14
6a	[N_3 [V_2 {V_3} [V_3 {N_3}	3	12	3	17
6b	[N_3 [V_3 {N_3} V_2 {}		12	3	20
6c	[N_3 [V_3 {} V_2 {N_3}	3	15	3	23
6d	[V_3 {} V_2 {}		15	3	26
6e	[V_3 {}		15	2	28
6f			15	0	28

String accepted. Scores: 15 by Metric 1, 28 by Metric 2.

Sentence (xviii): $Comp_1\ N_1\ N_3\ V_1\ V_2\ V_3$ Judgment: ok

Step	Store	Metric 1 New	Metric 1 Cum	Metric2 New	Metric2 Cum
1a	[$Comp_1$	1	1	1	1
2a	[$Comp_1$ [N_1	2	3	2	3
3a	[$Comp_1$ [N_1 [N_3	3	6	3	6
4a	[$Comp_1$ [N_1 [N_3 [V_1 {N_1 V_2}	4	10	4	10
5a	[$Comp_1$ [N_1 [N_3 [V_1 {N_1 V_2} [V_2 {V_3}	5	15	5	15
5b	[$Comp_1$ [N_1 [N_3 [V_2 {V_3} V_1 {N_1}		15	5	20
6a	[$Comp_1$ [N_1 [N_3 [V_2 {V_3} V_1 {N_1} [V_3 {N_3}	6	21	6	26
6b	[$Comp_1$ [N_1 [N_3 [V_2 {} V_1 {V_3 N_1} [V_3 {N_3}	6	27	6	32
6c	[$Comp_1$ [N_1 [N_3 [V_2 {N_3} V_1 {N_1}	6	33	6	38
6d	[$Comp_1$ [N_1 [N_3 [V_2 {} V_1 {N_3 N_1}	6	39	6	44
6e	[$Comp_1$ [N_1 [V_2 {} V_1 {N_1}		39	6	50
6f	[$Comp_1$ [V_2 {} V_1 {}		39	6	56
6g	[V_2 {} V_1 {}		39	5	61
6h	[V_2 {}		39	3	64
6i			39	0	64

String accepted. Scores: 39 by Metric 1, 64 by Metric 2.

Sentence (xix): $\mathsf{Comp}_1\ \mathsf{N}_3\ \mathsf{N}_1\ \mathsf{V}_1\ \mathsf{V}_2\ \mathsf{V}_3$ Judgment: ok

Step	Store	Metric 1 New	Metric 1 Cum	Metric2 New	Metric2 Cum
1a	$[\mathsf{Comp}_1$	1	1	1	1
2a	$[\mathsf{Comp}_1\ [\mathsf{N}_3$	2	3	2	3
3a	$[\mathsf{Comp}_1\ [\mathsf{N}_3\ [\mathsf{N}_1$	3	6	3	6
4a	$[\mathsf{Comp}_1\ [\mathsf{N}_3\ [\mathsf{N}_1\ [\mathsf{V}_1\ \{\mathsf{N}_1\ \mathsf{V}_2\}$	4	10	4	10
4b	$[\mathsf{Comp}_1\ [\mathsf{N}_3\ [\mathsf{V}_1\ \{\mathsf{V}_2\}$		10	4	14
5a	$[\mathsf{Comp}_1\ [\mathsf{N}_3\ [\mathsf{V}_1\ \{\mathsf{V}_2\}\ [\mathsf{V}_2\ \{\mathsf{V}_3\}$	5	15	5	19
5b	$[\mathsf{Comp}_1\ [\mathsf{N}_3\ [\mathsf{V}_2\ \{\mathsf{V}_3\}\ \mathsf{V}_1\ \{\}$		15	5	24
6a	$[\mathsf{Comp}_1\ [\mathsf{N}_3\ [\mathsf{V}_2\ \{\mathsf{V}_3\}\ \mathsf{V}_1\ \{\}\ [\mathsf{V}_3\ \{\mathsf{N}_3\}$	6	21	6	30
6b	$[\mathsf{Comp}_1\ [\mathsf{N}_3\ [\mathsf{V}_2\ \{\}\ \mathsf{V}_1\ \{\mathsf{V}_3\}\ [\mathsf{V}_3\ \{\mathsf{N}_3\}$	6	27	6	36
6c	$[\mathsf{Comp}_1\ [\mathsf{N}_3\ [\mathsf{V}_2\ \{\mathsf{N}_3\}\ \mathsf{V}_1\ \{\}$	6	33	6	42
6d	$[\mathsf{Comp}_1\ [\mathsf{N}_3\ [\mathsf{V}_2\ \{\}\ \mathsf{V}_1\ \{\mathsf{N}_3\}$	6	39	6	48
6e	$[\mathsf{Comp}_1\ [\mathsf{V}_2\ \{\}\ \mathsf{V}_1\ \{\}$		39	6	54
6f	$[\mathsf{V}_2\ \{\}\ \mathsf{V}_1\ \{\}$		39	5	59
6g	$[\mathsf{V}_2\ \{\}$		39	3	62
6h			39	0	62

String accepted. Scores: 39 by Metric 1, 62 by Metric 2.

Sentence (xx): $\mathsf{Comp}_1\ \mathsf{N}_1\ \mathsf{N}_3\ \mathsf{V}_3\ \mathsf{V}_1\ \mathsf{V}_2$ Judgment: ?

Step	Store	Metric 1 New	Metric 1 Cum	Metric2 New	Metric2 Cum
1a	$[\mathsf{Comp}_1$	1	1	1	1
2a	$[\mathsf{Comp}_1\ [\mathsf{N}_1$	2	3	2	3
3a	$[\mathsf{Comp}_1\ [\mathsf{N}_1\ [\mathsf{N}_3$	3	6	3	6
4a	$[\mathsf{Comp}_1\ [\mathsf{N}_1\ [\mathsf{N}_3\ [\mathsf{V}_3\ \{\mathsf{N}_3\}$	4	10	4	10
4b	$[\mathsf{Comp}_1\ [\mathsf{N}_1\ [\mathsf{V}_3\ \{\}$		10	4	14
5a	$[\mathsf{Comp}_1\ [\mathsf{N}_1\ [\mathsf{V}_3\ \{\}\ [\mathsf{V}_1\ \{\mathsf{N}_1\ \mathsf{V}_2\}$	5	15	5	19
6a	$[\mathsf{Comp}_1\ [\mathsf{N}_1\ [\mathsf{V}_3\ \{\}\ [\mathsf{V}_1\ \{\mathsf{N}_1\ \mathsf{V}_2\}\ [\mathsf{V}_2\ \{\mathsf{V}_3\}$	6	21	6	25
6b	$[\mathsf{Comp}_1\ [\mathsf{N}_1\ [\mathsf{V}_3\ \{\}\ [\mathsf{V}_2\ \{\mathsf{V}_3\}\ \mathsf{V}_1\ \{\mathsf{N}_1\}$		21	6	31
6c	$[\mathsf{Comp}_1\ [\mathsf{N}_1\ [\mathsf{V}_3\ \{\}\ [\mathsf{V}_2\ \{\}\ \mathsf{V}_1\ \{\mathsf{V}_3\ \mathsf{N}_1\}$	6	27	6	37
6d	$[\mathsf{Comp}_1\ [\mathsf{N}_1\ [\mathsf{V}_2\ \{\}\ \mathsf{V}_1\ \{\mathsf{N}_1\}$		27	6	43
6e	$[\mathsf{Comp}_1\ [\mathsf{V}_2\ \{\}\ \mathsf{V}_1\ \{\}$		27	6	49
6f	$[\mathsf{V}_2\ \{\}\ \mathsf{V}_1\ \{\}$		27	5	54
6g	$[\mathsf{V}_2\ \{\}$		27	3	57
6h			27	0	57

String accepted. Scores: 27 by Metric 1, 57 by Metric 2.

Sentence (xxi): $\mathsf{Comp}_1\ \mathsf{N}_3\ \mathsf{V}_3\ \mathsf{N}_1\ \mathsf{V}_1\ \mathsf{V}_2$ Judgment: ?

Step	Store	Metric 1 New	Metric 1 Cum	Metric2 New	Metric2 Cum
1a	$[\mathsf{Comp}_1$	1	1	1	1
2a	$[\mathsf{Comp}_1\ [\mathsf{N}_3$	2	3	2	3
3a	$[\mathsf{Comp}_1\ [\mathsf{N}_3\ [\mathsf{V}_3\ \{\mathsf{N}_3\}$	3	6	3	6
3b	$[\mathsf{Comp}_1\ [\mathsf{V}_3\ \{\}$		6	3	9
4a	$[\mathsf{Comp}_1\ [\mathsf{V}_3\ \{\}\ [\mathsf{N}_1$	4	10	4	13
5a	$[\mathsf{Comp}_1\ [\mathsf{V}_3\ \{\}\ [\mathsf{N}_1\ [\mathsf{V}_1\ \{\mathsf{N}_1\ \mathsf{V}_2\}$	5	15	5	18
5b	$[\mathsf{Comp}_1\ [\mathsf{V}_3\ \{\}\ [\mathsf{V}_1\ \{\mathsf{V}_2\}$		15	5	23
6a	$[\mathsf{Comp}_1\ [\mathsf{V}_3\ \{\}\ [\mathsf{V}_1\ \{\mathsf{V}_2\}\ [\mathsf{V}_2\ \{\mathsf{V}_3\}$	6	21	6	29
6b	$[\mathsf{Comp}_1\ [\mathsf{V}_3\ \{\}\ [\mathsf{V}_2\ \{\mathsf{V}_3\}\ \mathsf{V}_1\ \{\}$		21	6	35
6c	$[\mathsf{Comp}_1\ [\mathsf{V}_3\ \{\}\ [\mathsf{V}_2\ \{\}\ \mathsf{V}_1\ \{\mathsf{V}_3\}$	6	27	6	41
6d	$[\mathsf{Comp}_1\ [\mathsf{V}_2\ \{\}\ \mathsf{V}_1\ \{\}$		27	6	47
6e	$[\mathsf{V}_2\ \{\}\ \mathsf{V}_1\ \{\}$		27	5	52
6f	$[\mathsf{V}_2\ \{\}$		27	3	55
6g			27	0	55

String accepted. Scores: 27 by Metric 1, 55 by Metric 2.

Sentence (xxii): $\mathsf{Comp}_1\ \mathsf{N}_3\ \mathsf{N}_1\ \mathsf{V}_3\ \mathsf{V}_1\ \mathsf{V}_2$ Judgment: ?

Step	Store	Metric 1 New	Metric 1 Cum	Metric2 New	Metric2 Cum
1a	$[\mathsf{Comp}_1$	1	1	1	1
2a	$[\mathsf{Comp}_1\ [\mathsf{N}_3$	2	3	2	3
3a	$[\mathsf{Comp}_1\ [\mathsf{N}_3\ [\mathsf{N}_1$	3	6	3	6
4a	$[\mathsf{Comp}_1\ [\mathsf{N}_3\ [\mathsf{N}_1\ [\mathsf{V}_3\ \{\mathsf{N}_3\}$	4	10	4	10
5a	$[\mathsf{Comp}_1\ [\mathsf{N}_3\ [\mathsf{N}_1\ [\mathsf{V}_3\ \{\mathsf{N}_3\}\ [\mathsf{V}_1\ \{\mathsf{N}_1\ \mathsf{V}_2\}$	5	15	5	15
6a	$[\mathsf{Comp}_1\ [\mathsf{N}_3\ [\mathsf{N}_1\ [\mathsf{V}_3\ \{\mathsf{N}_3\}\ [\mathsf{V}_1\ \{\mathsf{N}_1\ \mathsf{V}_2\}\ [\mathsf{V}_2\ \{\mathsf{V}_3\}$	6	21	6	21
6b	$[\mathsf{Comp}_1\ [\mathsf{N}_3\ [\mathsf{N}_1\ [\mathsf{V}_3\ \{\mathsf{N}_3\}\ [\mathsf{V}_2\ \{\mathsf{V}_3\}\ \mathsf{V}_1\ \{\mathsf{N}_1\}$		21	6	27
6c	$[\mathsf{Comp}_1\ [\mathsf{N}_3\ [\mathsf{N}_1\ [\mathsf{V}_3\ \{\mathsf{N}_3\}\ [\mathsf{V}_2\ \{\}\ \mathsf{V}_1\ \{\mathsf{V}_3\ \mathsf{N}_1\}$	6	27	6	33
6d	$[\mathsf{Comp}_1\ [\mathsf{N}_3\ [\mathsf{N}_1\ [\mathsf{V}_2\ \{\mathsf{N}_3\}\ \mathsf{V}_1\ \{\mathsf{N}_1\}$	6	33	6	39
6e	$[\mathsf{Comp}_1\ [\mathsf{N}_3\ [\mathsf{V}_2\ \{\mathsf{N}_3\}\ \mathsf{V}_1\ \{\}$		33	6	45
6f	$[\mathsf{Comp}_1\ [\mathsf{N}_3\ [\mathsf{V}_2\ \{\}\ \mathsf{V}_1\ \{\mathsf{N}_3\}$	6	39	6	51
6g	$[\mathsf{Comp}_1\ [\mathsf{V}_2\ \{\}\ \mathsf{V}_1\ \{\}$		39	6	57
6h	$[\mathsf{V}_2\ \{\}\ \mathsf{V}_1\ \{\}$		39	5	62
6i	$[\mathsf{V}_2\ \{\}$		39	3	65
6j			39	0	65

String accepted. Scores: 39 by Metric 1, 65 by Metric 2.

Sentence (xxiii): $Comp_1\ N_1\ V_2\ V_1\ N_3\ V_3$ Judgment: ?

Step	Store	Metric 1 New	Metric 1 Cum	Metric2 New	Metric2 Cum
1a	[$Comp_1$	1	1	1	1
2a	[$Comp_1$ [N_1	2	3	2	3
3a	[$Comp_1$ [N_1 [V_2 {V_3}	3	6	3	6
4a	[$Comp_1$ [N_1 [V_2 {V_3} [V_1 {N_1 V_2}	4	10	4	10
4b	[$Comp_1$ [N_1 [V_2 {V_3} V_1 {N_1}		10	4	14
4c	[$Comp_1$ [V_2 {V_3} V_1 {}		10	4	18
4d	[V_2 {V_3} V_1 {}		10	3	21
5a	[V_2 {V_3} V_1 {} [N_3	4	14	4	25
6a	[V_2 {V_3} V_1 {} [N_3 [V_3 {N_3}	5	19	5	30
6b	[V_2 {V_3} V_1 {} [V_3 {}		19	5	35
6c	[V_2 {} V_1 {V_3} [V_3 {}	5	24	5	40
6d	[V_2 {} V_1 {}		24	5	45
6e	[V_2 {}		24	3	48
6f			24	0	48

String accepted. Scores: 24 by Metric 1, 48 by Metric 2.

Sentence (xxiv): $Comp_1\ N_1\ V_2\ N_3\ V_1\ V_3$ Judgment: *

Step	Store	Metric 1 New	Metric 1 Cum	Metric2 New	Metric2 Cum
1a	[$Comp_1$	1	1	1	1
2a	[$Comp_1$ [N_1	2	3	2	3
3a	[$Comp_1$ [N_1 [V_2 {V_3}	3	6	3	6
4a	[$Comp_1$ [N_1 [V_2 {V_3} [N_3	4	10	4	10
5a	[$Comp_1$ [N_1 [V_2 {V_3} [N_3 [V_1 {N_1 V_2}	5	15	5	15
6a	[$Comp_1$ [N_1 [V_2 {V_3} [N_3 [V_1 {N_1 V_2} [V_3 {N_3}	6	21	6	21

String rejected.

Sentence (xxv): $Comp_1\ N_1\ N_3\ V_2\ V_1\ V_3$ Judgment: *?

Step	Store	Metric 1 New	Metric 1 Cum	Metric2 New	Metric2 Cum
1a	[$Comp_1$	1	1	1	1
2a	[$Comp_1$ [N_1	2	3	2	3
3a	[$Comp_1$ [N_1 [N_3	3	6	3	6
4a	[$Comp_1$ [N_1 [N_3 [V_2 {V_3}	4	10	4	10
5a	[$Comp_1$ [N_1 [N_3 [V_2 {V_3} [V_1 {N_1 V_2}	5	15	5	15
5b	[$Comp_1$ [N_1 [N_3 [V_2 {V_3} V_1 {N_1}		15	5	20
6a	[$Comp_1$ [N_1 [N_3 [V_2 {V_3} V_1 {N_1} [V_3 {N_3}	6	21	6	26
6b	[$Comp_1$ [N_1 [N_3 [V_2 {} V_1 {V_3 N_1} [V_3 {N_3}	6	27	6	32
6c	[$Comp_1$ [N_1 [N_3 [V_2 {N_3} V_1 {N_1}	6	33	6	38
6d	[$Comp_1$ [N_1 [N_3 [V_2 {} V_1 {N_3 N_1}	6	39	6	44
6e	[$Comp_1$ [N_1 [V_2 {} V_1 {N_1}		39	6	50
6f	[$Comp_1$ [V_2 {} V_1 {}		39	6	56
6g	[V_2 {} V_1 {}		39	5	61
6h	[V_2 {}		39	3	64
6i			39	0	64

String accepted. Scores: 39 by Metric 1, 64 by Metric 2.

Sentence (xxvi): $Comp_1\ N_3\ N_1\ V_2\ V_1\ V_3$ Judgment: *?

Step	Store	Metric 1 New	Metric 1 Cum	Metric2 New	Metric2 Cum
1a	[$Comp_1$	1	1	1	1
2a	[$Comp_1$ [N_3	2	3	2	3
3a	[$Comp_1$ [N_3 [N_1	3	6	3	6
4a	[$Comp_1$ [N_3 [N_1 [V_2 {V_3}	4	10	4	10
5a	[$Comp_1$ [N_3 [N_1 [V_2 {V_3} [V_1 {N_1 V_2}	5	15	5	15
5b	[$Comp_1$ [N_3 [N_1 [V_2 {V_3} V_1 {N_1}		15	5	20
5c	[$Comp_1$ [N_3 [V_2 {V_3} V_1 {}		15	5	25
6a	[$Comp_1$ [N_3 [V_2 {V_3} V_1 {} [V_3 {N_3}	6	21	6	31
6b	[$Comp_1$ [N_3 [V_2 {} V_1 {V_3} [V_3 {N_3}	6	27	6	37
6c	[$Comp_1$ [N_3 [V_2 {N_3} V_1 {}	6	33	6	43
6d	[$Comp_1$ [N_3 [V_2 {} V_1 {N_3}	6	39	6	49
6e	[$Comp_1$ [V_2 {} V_1 {}		39	6	55
6f	[V_2 {} V_1 {}		39	5	60
6g	[V_2 {}		39	3	63
6h			39	0	63

String accepted. Scores: 39 by Metric 1, 63 by Metric 2.

Sentence (xxvii): $Comp_1\ V_2\ N_1\ N_3\ V_1\ V_3$ Judgment: *

Step	Store	Metric 1 New	Metric 1 Cum	Metric2 New	Metric2 Cum
1a	[$Comp_1$	1	1	1	1
2a	[$Comp_1$ [V_2 {V_3}	2	3	2	3
3a	[$Comp_1$ [V_2 {V_3} [N_1	3	6	3	6
4a	[$Comp_1$ [V_2 {V_3} [N_1 [N_3	4	10	4	10
5a	[$Comp_1$ [V_2 {V_3} [N_1 [N_3 [V_1 {N_1 V_2}	5	15	5	15
6a	[$Comp_1$ [V_2 {V_3} [N_1 [N_3 [V_1 {N_1 V_2} [V_3 {N_3}	6	21	6	21

String rejected.

Sentence (xxviii): $Comp_1\ V_2\ N_1\ V_1\ N_3\ V_3$ Judgment: ?

Step	Store	Metric 1 New	Metric 1 Cum	Metric2 New	Metric2 Cum
1a	[$Comp_1$	1	1	1	1
2a	[$Comp_1$ [V_2 {V_3}	2	3	2	3
3a	[$Comp_1$ [V_2 {V_3} [N_1	3	6	3	6
4a	[$Comp_1$ [V_2 {V_3} [N_1 [V_1 {N_1 V_2}	4	10	4	10
4b	[$Comp_1$ [V_2 {V_3} [V_1 {V_2}		10	4	14
4c	[$Comp_1$ [V_2 {V_3} V_1 {}		10	4	18
4d	[V_2 {V_3} V_1 {}		10	3	21
5a	[V_2 {V_3} V_1 {} [N_3	4	14	4	25
6a	[V_2 {V_3} V_1 {} [N_3 [V_3 {N_3}	5	19	5	30
6b	[V_2 {V_3} V_1 {} [V_3 {}		19	5	35
6c	[V_2 {} V_1 {V_3} [V_3 {}	5	24	5	40
6d	[V_2 {} V_1 {}		24	5	45
6e	[V_2 {}		24	3	48
6f			24	0	48

String accepted. Scores: 24 by Metric 1, 48 by Metric 2.

Sentence (xxix): $Comp_1\ V_2\ N_3\ N_1\ V_1\ V_3$ Judgment: *

Step	Store	Metric 1 New	Metric 1 Cum	Metric2 New	Metric2 Cum
1a	[$Comp_1$	1	1	1	1
2a	[$Comp_1$ [V_2 {V_3}	2	3	2	3
3a	[$Comp_1$ [V_2 {V_3} [N_3	3	6	3	6
4a	[$Comp_1$ [V_2 {V_3} [N_3 [N_1	4	10	4	10
5a	[$Comp_1$ [V_2 {V_3} [N_3 [N_1 [V_1 {N_1 V_2}	5	15	5	15
5b	[$Comp_1$ [V_2 {V_3} [N_3 [V_1 {V_2}		15	5	20
6a	[$Comp_1$ [V_2 {V_3} [N_3 [V_1 {V_2} [V_3 {N_3}	6	21	6	26

String rejected.

Sentence (xxx): $Comp_1\ N_3\ V_2\ N_1\ V_1\ V_3$ Judgment: *?

Step	Store	Metric 1 New	Metric 1 Cum	Metric2 New	Metric2 Cum
1a	[$Comp_1$	1	1	1	1
2a	[$Comp_1$ [N_3	2	3	2	3
3a	[$Comp_1$ [N_3 [V_2 {V_3}	3	6	3	6
4a	[$Comp_1$ [N_3 [V_2 {V_3} [N_1	4	10	4	10
5a	[$Comp_1$ [N_3 [V_2 {V_3} [N_1 [V_1 {N_1 V_2}	5	15	5	15
5b	[$Comp_1$ [N_3 [V_2 {V_3} [V_1 {V_2}		15	5	20
5c	[$Comp_1$ [N_3 [V_2 {V_3} V_1 {}		15	5	25
6a	[$Comp_1$ [N_3 [V_2 {V_3} V_1 {} [V_3 {N_3}	6	21	6	31
6b	[$Comp_1$ [N_3 [V_2 {} V_1 {V_3} [V_3 {N_3}	6	27	6	37
6c	[$Comp_1$ [N_3 [V_2 {N_3} V_1 {}	6	33	6	43
6d	[$Comp_1$ [N_3 [V_2 {} V_1 {N_3}	6	39	6	49
6e	[$Comp_1$ [V_2 {} V_1 {}		39	6	55
6f	[V_2 {} V_1 {}		39	5	60
6g	[V_2 {}		39	3	63
6h			39	0	63

String accepted. Scores: 39 by Metric 1, 63 by Metric 2.