HyTAG:
A New Type of Tree Adjoining Grammars for Hybrid Syntactic Representation of Free Word Order Languages

Dissertation

zur Erlangung des Grades des
Doktors der Naturwissenschaften
der Technischen Fakultät
der Universität des Saarlandes

von

Tilman Becker

Saarbrücken
1994

Tag des Kolloquiums:	31. Januar 1994
Dekan:	Prof. Dr. G. Hotz
Berichterstatter:	Prof. Dr. W. Wahlster
	Prof. K. Vijay-Shanker, Ph.D.

Into the dark ("Ins Dunkel"), Wassily Kandinsky, May 1928

Contents

Zusammenfassung

Einleitung

Das Ziel dieser Arbeit ist es, ein formales Modell für die Repräsentation der Syntax natürlicher Sprache zu entwickeln, einem Kernpunkt der Verarbeitung natürlicher Sprache. Diese wird üblicherweise als ein mehrschichtiger Prozeß beschrieben, beginnend mit akustischen Daten, gefolgt von einer phonologischen, morphologischen, syntaktischen, semantischen und pragmatischen Ebene. In allen diesen Ebenen gibt es zwei wesentliche Informationsquellen: zum einen ein Lexikon, daß alle wortspezifischen Informationen (für alle Ebenen) enthält und zum anderen jeweils ebenenspezifische, verallgemeinerte Information. Die ebenspezifische Information der syntaktischen Ebene ist die Grammatik. Die vorliegende Arbeit entwickelt ein hybrides Modell[1] für die Repräsentation syntaktischen Wissens und die Interaktion mit syntaktischer Information aus dem Lexikon.

Eine wichtige Entwicklung im Bereich der Verarbeitung natürlicher Sprache war die Einführung der unifikationsbasierten Formalismen. Als einer ihrer Vorteile wird neben der grossen Ausdrucksstärke ihre Uniformität herausgestellt. Ein vergleichsweise einfacher Mechanismus (die Unifikationsoperation) wird verwendet, um alle syntaktischen (und auch andere) Phänomene zu beschreiben.

Ein Nachteil unifikationsbasierter Formalismen ist allerdings ihre formale Mächtigkeit. Ein reiner unifikationsbasierter Formalismus ist im allgemeinen unentscheidbar; allerdings tritt dieses Problem bei linguistisch plausiblen Grammatiken nicht auf. Effizienz der Algorithmen (Parsen und Generierung) bleibt allerdings ein Problem unifikationsbasierter Formalismen. Andererseits werden auch in unifikationsbasierten Formalismen bestimmte syntaktische Phänomene mit speziellen, eingeschränkten Methoden beschrieben.

[1] Der Begriff wird hier nicht, wie oft üblich, in dem Sinne gebraucht, daß ein bestimmtes Phänomen innerhalb des Formalismus mit verschiedenen Methoden dargestellt werden kann. Der entwickelte Formalismus enthält verschiedene, aber hochspezialisierte Methoden, die verschiedenen Phänomene der Syntax zu repräsentieren.

Es ist wünschenswert, mehr über die genauen (formalen) Eigenschaften der verschiedenen syntaktischen Phänomene zu wissen. Dies birgt auch die Hoffnung, daß die formalen Verabeitungsmodelle größere Relevanz für psychologische Modelle haben.

Daher stellt sich die Aufgabe, einen Formalismus zu entwickeln, der im Idealfall für jedes syntaktische Phänomen die effizienteste Beschreibung verwendet. Ein solcher Ansatz hat den scheinbaren Nachteil einer komplexen Beschreibungssprache im Gegensatz zu der einfachen Unifikationsoperation. Viele unifikationsbasierte linguistische Theorien verwenden jedoch eine Anzahl spezieller Operation, z.B. kontextfreie Regeln (LFG, PATR-II, bzw. Schemata in HPSG), die die Unifikation von Featurestrukturen beschreiben.

Es erscheint sinnvoll, solche eingeschränkten Methoden bereits in den Formalismus selbst zu integrieren und die eingeschränkte Mächtigkeit spezieller Mechanismen auszunutzen. Da ein solcher Formalismus verschiedene Darstellungsformen hat, kann man ihn als hybride Repräsentation bezeichnen.

Der HyTAG Formalismus, der in dieser Arbeit entwickelt wird, ist eine derartige hybride Repräsentation. Sie basiert auf einer Erweiterung von "Tree Adjoining Grammar" (TAG, Baumadjunktionsgrammatik) und verwendet Unifikation nur für die Aspekte der Syntax, die dafür am besten geeignet sind. Dies sind im wesentlichen Kongruenzen ("agreement features") wie Geschlecht, Person, Singular/Plural etc. (siehe auch Abschnitt 1.1.3). Da nur diese Phänomene mit Unifikation beschrieben werden, ist es möglich, eine stark eingeschränkte Variante von Unifikation zu verwenden, die die formale Mächtigkeit von TAG nicht vergrößert.

Der hier zugrundeliegende Formalismus TAG beschreibt die syntaktischen Phänomene der Rekursion und die Erzeugung von Phrasenstrukturen. TAGs haben dafür eine Reihe günstiger Eigenschaften. Erstens haben die elementaren Strukturen der Grammatik eine erweiterte Lokalitätsdomäne, die es erlaubt, alle (auch unbeschränkte) Abhängigkeiten eines Wortes lokal zu beschreiben.

Zweitens werden in einer TAG alle rekursiven Phänomene in Form von auxiliaren Bäumen explizit faktorisiert.

Drittens können TAGs lexikalisiert werden, d.h. alle Informationen, die ein Wort zur Phrasenstruktur beiträgt, sind direkt mit dem Wort assoziiert und jede elementare Struktur der Grammatik ist einem Wort zugeordnet.

Wie im folgenden Abschnitt weiter ausgeführt, ergibt sich durch die Untersuchung des Long-Distance Scrambling Phänomen die Notwendigkeit, eine erweiterte Version von TAGs zu verwenden, die dann die Grundlage für zwei weitere Mechanismen von HyTAG bildet, Metaregeln und Vererbungsnetzwerke.

Übersicht und Ergebnisse

In dieser Arbeit entwickele ich HyTAG, eine Architektur zur kompakten und effizienten Repräsentation syntaktischen Wissens in der Verarbeitung natürlicher Sprache. Das System basiert auf MC-TAG-DL, dem in dieser Arbeit entwickelten Baum-Ersetzungssystem, das Phrasenstrukturen als syntaktische Beschreibung natürlichsprachlicher Sätze erzeugt. Die Wahl des MC-TAG-DL Formalismus ergibt sich als Ergebnis der Untersuchung des "Long-Distance Scrambling" (LDS) Phänomens, einem Phänomen, das in Sprachen mit freier Wortstellung, wie z.B. Deutsch oder Koreanisch, auftritt. Die Untersuchungen zeigen, daß TAGs keine adäquate Beschreibung des LDS Phänomens ermöglichen. Als eine Methode der formalen Beschreibung von Adäquatheit wird der Begriff der *Ableitungsmächtigkeit* ("derivational generative capacity") entwickelt und verwendet, um diese Aussage formal zu beweisen. Die elementaren Strukturen einer MC-TAG-DL Grammatik sind *elementare Baumsequenzen*, für die eine kompakte Repräsentation entwickelt wird. Diese basiert auf Metaregeln und Vererbungsnetzwerken. Beide Mechanismen erlauben es, sowohl Redundanzen zu faktorisieren als auch linguistische Generalisierungen zu erfassen. In einer solchen Repräsentation kann eine Grammatik besser entwickelt und gewartet werden und es kann einfacher sichergestellt werden, daß linguistische (und andere) Prinzipien in der gesamten Grammatik eingehalten werden.

Das erste Kapitel führt alle relevanten Formalismen ein und enthält die Einführung von 2-SA, einem neuen Automatenmodell für TAGs. 2-SA haben im Gegensatz zu herkömmlichen Kellerautomaten zwei Keller, allerdings kann der zweite Keller nur mit bestimmten Einschränkungen verwendet werden. Das Ablegen von Symbolen ist jederzeit erlaubt, das Löschen von Symbolen auf dem zweiten Keller jedoch nur, wenn der erste Keller leer ist. Der Äquivalenzbeweis beruht auf einer Simulation durch EPDA ("Embedded Push Down Automaton"), dem ersten Automatenmodell für TAGs. Die verallgemeinerte Variante des Automaten (n-SA) bildet eine arithmetische Hierarchie und ist ein Kandidat für ein Automatenmodell für "Linear Kontextfreie Ersetzungssysteme" (LCFRS).

Das nächste Kapitel diskutiert zuerst bekannte Phänomene in natürlichen Sprachen, die nicht mit TAGs beschrieben werden können. Dies sind zum einen Extraktionsphänomene im Schwedischen und Norwegischen [Mil91]. Basierend auf einer bestimmten linguistischen Analyse werden überkreuzte Abhängigkeiten als rechtsgekämmte Bäume beschrieben, eine Struktur, die mit TAGs nicht erzeugt werden kann, obwohl TAGs (mit anderen Phrasenstrukturen) die zugrunde liegende Sprache erzeugen können. Das zweite Argument basiert auf Zahlennamen im Chinesischen [Rad91], ist allerdings nicht eindeutig ein syntaktisches Argument.

In den Abschnitten 2.3 und 2.4 werden neue Daten aus dem Deutschen und Koreanischen zum Long-Distance Scrambling präsentiert und analysiert. Dabei werden Argumente von eingebetteten Sätzen in den einbettenden Satz „bewegt“:

... daß ich den Kühlschrank$_i$ dem Kunden [t_i zu reparieren] versprochen habe.

Es zeigt sich, daß Scrambling in zweifacher Hinsicht unbeschränkt ist:

1. Es gibt keine syntaktische Beschränkung der Anzahl der bewegten Argumente.
2. Es gibt keine syntaktische Beschränkung der Anzahl der übersprungenen Einbettungen.

Im folgenden Kapitel 3 wird eine formale Charakterisierung des LDS Phänomens gegeben, die es erlaubt zu zeigen, daß das LDS Phänomen mit keinem LCFRS (und damit auch nicht mit TAGs) adäquat beschrieben werden kann. Dies wird durch die Verwendung des neueingeführten Begriffes der Ableitungsmächtigkeit möglich.

Ausgehend von diesem negativen Resultat, werden in Kapitel 4 zwei neue Erweiterungen des TAG Formalismus entwickelt, die es einerseits ermöglichen, das LDS Phänomen adäquat zu beschreiben, aber andererseits möglichst viele Eigenschaften des TAG Formalismus erhalten. Die neuen Formalismen sind “Free-Order TAG” (FO-TAG) und “Multi-Component TAG with Dominance Links” (MC-TAG-DL); letzterer stellt sich als der geeignetere heraus und ist die Grundlage der weiteren Betrachtungen. Um freie Wortstellungen zu ermöglichen, werden in FO-TAG die lineare Anordungen der Knoten der elementaren Bäume nicht eindeutig festgelegt, sondern durch Anordnungsregeln (“linear precedence rules”) nur eingeschränkt. Im MC-TAG-DL Ansatz werden die elementaren Bäume in mehrere Teile aufgespalten, die an verschiedenen Knoten adjungiert werden können. Die einzelnen Teilbäume müssen bestimmte Dominanzregeln (“dominance links”) einhalten.

In Kapitel 5 untersuchen wir die Beziehung zwischen der MC-TAG-DL Analyse des LDS Phänomens und der Analyse von “unbounded dependencies” in HPSG, einem unifikationsbasierten Formalismus. Eine Analyse in HPSG (mittels des NONLOCAL features) basiert auf der Zwischenspeicherung von „bewegten“ Argumenten entlang eines Pfades im Phrasenstrukturbaum, dies entspricht den „Dominance Links“ im MC-TAG-DL Formalismus. Darauf basierend, zeigen wir die Eignung von MC-TAG-DL als Zielformalismus einer Kompilation von HPSG.

Basierend auf dem entwickelten MC-TAG-DL Formalismus, entwickeln wir den HyTAG Formalismus, der es erlaubt, Grammatiken im MC-TAG-DL Formalismus kompakt und effizient zu repräsentieren. Kapitel 6 führt Metaregeln ein, die eine erhebliche Reduzierung von Redundanzen und eine Beschreibung linguistischer Generalisierungen erlauben. Ein Beispiel ist die einheitliche Beschreibung aller elementaren Baumsequenzen für Passiv-Sätze mittels einer Metaregel (siehe auch den Anhang).

In einer regulären TAG (oder MC-TAG-DL) gibt es viele Paare von ähnlichen Baumsequenzen, die jeweils einen Satz im Aktiv und Passiv beschreiben. Die Metaregel macht diese Gemeinsamkeiten explizit und erspart die explizite Repräsentierung aller Passiv-Baumsequenzen.

In Abschnitt 6.1.4 zeige ich, wie das Problem der formalen Mächtigkeit von Metaregeln, die in anderen Formalismen zur Unentscheidbarkeit führen, in einer wohlmotivierten Weise im MC-TAG-DL Formalismus (und auch im TAG Formalismus) gelöst werden kann. Da TAGs (und auch MC-TAG-DL) zum einen Rekursion in Form von auxiliaren Bäumen beschreiben und zum anderen die erweiterte lokale Domäne eines Wortes (aber nicht mehr als diese) mit einem elementaren Baum (einer elementaren Baumsequenz) beschreiben, kann die Menge der durch Metaregeln erzeugten neuen elementaren Bäume beschränkt werden, d.h. unbeschränkte Rekursion von Metaregeln wird durch die Prinzipien des Formalismus ausgeschlossen.

In Kapitel 7 führe ich Vererbungsnetzwerke ein als ein weiteres Mittel zur kompakten Repräsentation der elementaren Baumsequenzen und definiere schließlich den hybriden Repräsentationsformalismus HyTAG als die Integration von Metaregeln, Vererbungsnetzwerk und dem MC-TAG-DL Formalismus. Es zeigt sich, daß sowohl Metaregeln als auch das Vererbungsnetzwerk zwar prinzipiell durch einander ersetzt werden können, aber tatsächlich jeweils spezielle Funktionen erfüllen.

Aufbauend auf den Ergebnissen dieser Arbeit, stellen sich eine Reihe neuer Probleme. Dazu gehören die Erweiterung des MC-TAG-DL Formalismus zu einer polynomiell parsebaren Variante. Ein solches System wird momentan (siehe [Ram92]) untersucht. Eine genaue Charakterisierung des n-SA Automatenmodells ist ein offenes Problem, von Interesse ist hier auch die Beziehung zu „Klammergrammatiken“ (CCFG, siehe [Gua92]). Eine Untersuchung der Relation zwischen der FO-TAG Analyse von LDS und der „clause union“ Analyse im HPSG Formalismus würde Ergebnisse über die formalen Eigenschaften von „clause union“ mit sich bringen.

Aus linguistischer Sicht ist es interessant, die Prinzipien, die den internen Aufbau der elementaren Baumsequenzen regeln (z.B. X-Bar Theorie, Projektionen), in den Formalismus zu integrieren. Auch das Interface zwischen Lexikon und Grammatik, dabei insbesondere die Rolle der Morphologie, ist ein interessantes Problem.

Im Anhang beschreibe ich schließlich ein kleines Fragment der Syntax der deutschen Sprache, um die Vorteile der entwickelten Architektur zu verdeutlichen.

Hauptresultate

Kurz gesagt, präsentiert diese Arbeit die folgenden Ergebnisse:

2-SA, ein neues Automatenmodell für TAGs, wird eingeführt; n-SA, ihre generalisierte Form, sind ein Kandidat für ein Automatenmodell für LCFRS.

Long-Distance Scrambling (LDS), ein Phänomen in Sprachen mit freier Wortstellung, wird untersucht und formal beschrieben. Der Begriff der *Ableitungsmächtigkeit* ("derivational generative capacity"), eine neue Methode zum Prüfen der Beschreibungsmächtigkeit eines Formalismus, wird entwickelt. Wir zeigen, daß die Formalisierung des LDS Phänomens mit LCFRS (und damit auch mit TAGs) nicht adäquat beschrieben werden kann, d.h. LCFRS haben nicht die Ableitungsmächtigkeit, das LDS Phänomen zu beschreiben.

Als Konsequenz daraus entwickeln wir FO-TAG und MC-TAG-DL, zwei neue Formalismen, die auf TAGs basieren, aber das LDS Phänomen adäquat beschreiben können. Wir verwenden den insgesamt besser geeigneten Formalismus MC-TAG-DL als Grundlage für die weiteren Untersuchungen und zeigen, wie die Behandlung des LDS Phänomens in einem unifikationsbasierten Formalismus (HPSG) in eine MC-TAG-DL-Darstellung transferiert werden kann.

Auf der Basis von MC-TAG-DL definieren wir schließlich HyTAG, einen Formalismus zur hybriden Repräsentation syntaktischen Wissens. Die beiden weiteren Hauptbestandteile von HyTAG sind Metaregeln und Vererbungsnetzwerke, die es erlauben, eine MC-TAG-DL Grammatik kompakt und effizient zu repräsentieren. Wir geben eine wohlmotivierte Definition von Metaregeln, die im Gegensatz zu andern Ansätzen keine Vergrößerung der formalen Mächtigkeit mit sich bringt.

Schließlich demonstrieren wir die Vorteile des HyTAG-Systems an einem kleinen Fragment der deutschen Syntax.

Introduction

This thesis attempts to develop a model for one of the central topics of Natural Language Processing (NLP), namely the syntax of natural language. Usually the processing of natural language is seen as a multi-tiered process, starting from the acoustic data going through tiers of phonological, morphological, syntactic, semantic and pragmatic processing[2]. All these tiers base their processing on two basic sources of information: A lexicon which stores all the idiosyncratic information (for all tiers) about lexemes and tier-specific, generalized information. The tier-specific information for the syntactic tier is, of course, the grammar. This thesis attempts to develop a hybrid model[3] for the representation of the grammar as syntactic knowledge and its interaction with the syntactic information in the lexicon.

The proposed system uses two different mechanisms, metarules and an inheritance network to allow for a very compact and efficient representation of the grammar. The underlying formal system is a new syntactic formalism, MC-TAG-DL, which is derived from Tree Adjoining Grammar (TAG). The resulting formalism which integrates MC-TAG-DL, metarules and inheritance networks is called HyTAG.

One of the major innovations in Natural Language Processing has been the introduction of so-called unification based formalisms. It has been argued that one of their main advantages besides their expressive power is their uniformity. One relatively simple formalism can be used to describe all syntactic (and even other) phenomena.

One drawback, however, is the enormous formal power which is a result of the expressive power. A pure unification based formalism is undecidable, i.e. there can be no general parser for these formalisms. However, linguistically reasonable grammars do not exhibit this undecidability and there exist numerous parsers for unification

[2] This process is not necessarily a sequential one, versions of parallel, cascaded and other models have been proposed. Also, it is of course bi-directional: In the case of language generation, the process starts with a semantic representation and produces an acoustic output.

[3] The term "hybrid" is used here not in the usual sense of allowing different ways to describe the *same* phenomena but merely in the sense that the formalism contains different, but in this case highly specialized means for describing the different phenomena of natural language syntax.

based formalisms. It turns out that some of the syntactic phenomena actually have limited complexity and can be handled quite efficiently by a parser. But in general, efficiency remains a problem for a unification-based formalism.

On the other hand, it is certainly desirable to know more about the exact limitations of the various syntactic phenomena. This would hopefully allow for the (formal) processing models to come closer to the psychological reality and also allow for more efficient parsing and generation algorithms.

This arouses the question as to why not start with a formalism that uses the most efficient description for each of the various syntactic phenomena. Such an approach seems to have the disadvantage of producing a much more complex description language and loosing the simple and uniform language of unification based formalism. But it turns out that in unification based formalisms the actual descriptions for a natural language are rather complex too, and although the formalism itself does not pose restrictions on the description, specific instances of unification-based linguistic theories (e.g. HPSG, CUG, LFG) do introduce a lot of restrictions to reflect the regularities of natural language.

Also, most unification based formalisms are already augmented by some rules that govern the combination (unification) of elementary feature structures. In HPSG, e.g. there are the schemata, in LFG and PATR-II there are explicit combination rules in the style of context-free rules.

Therefore one might introduce some of these restrictions in the formalism itself and thereby exploit consequences such as a restricted formal power of the resulting formalism. But naturally, such a formalism will then have many different means and elements for the different syntactic phenomena and might therefore be called a Hybrid Representation.

The system proposed in this thesis is such a hybrid system. It is based on an extended version of TAG and makes use of unification only for those phenomena of Syntax that have been identified as the most suitable for a unification based formalism. These are basically the agreement features such as gender, number, person and others (see section 1.1.3). Because only these phenomena are handled by unification, a considerably less powerful version of unification is sufficient and the parsing complexity of the extended TAG formalism remains basically unchanged.

The basis for the proposed formalism, however, is the mechanism which captures such syntactic phenomena as recursion and locality. It is based on a formalism that gives the most suitable description for these phenomena, namely that of Tree Adjoining Grammars (TAGs). The extension of the TAG formalism which is developed in this thesis is called MC-TAG-DL ("Multi-Component TAG with Dominance-Links").

The need for a new basic syntactic formalism arises from various phenomena in natural language, in particular this thesis investigates Long-Distance Scrambling (LDS) in free word order languages such as German and Korean. In order to formally proof the inadequacy of a wide range of formalisms (including TAGs) for the characterization of LDS, a new notion of generative capacity, "derivative generative capacity" is introduced.

However, the basis for the new syntactic formalism developed in this thesis is Tree Adjoining Grammar (TAG) (see [JLT75]). TAGs have a number of desirable properties for a formal system to describe the syntax of natural language. First, the elementary structures of the grammar have an extended domain of locality, which allows to express all (including long-distance) dependencies of some lexical item.

Second, all recursive phenomena are factored out in a TAG in the sense that elementary structures of the grammar (namely the auxiliary trees) explicitly represent the recursive phenomena of the syntactic description.

Third, TAGs can be lexicalized. That is, all the structural information that a lexical item (a word) contributes to the phrase structure of a sentence is associated with this lexical item. It also implies that no other syntactic information than that supplied by the lexemes is necessary to construct the syntactic description of a sentence.

This thesis gives a detailed analysis of the formal properties of Long-Distance Scrambling (LDS), a syntactic phenomenon found in so-called free word order languages, especially in German. With the introduction of a new notion for the generative power of formal systems (the generative derivative capacity), I can show that TAGs in its original form is not adequate to describe the LDS phenomenon. However, I propose an extension of TAGs called MC-TAG-DL that can capture the LDS phenomenon adequately and also keeps almost all of the desirable properties of TAGs.

Based on this new formalism, I then develop HyTAG, the hybrid representation formalism for the grammar (the syntactic knowledge) written in this MC-TAG-DL formalism. As I will show by examples from German and English, the grammars contain great amounts of redundancies which can be compiled out and represented much more compactly with various means such as metarules, an inheritance network, and features. Some of these means have already been introduced in various forms for TAGs and other formal systems, but e.g. metarules posed formal problems which, as chapter 6 shows, can be solved in a TAG based formalism. Another aspect of such a compact representation is the linguistic adequacy of the representation. All of the proposed means for compactifying the grammar are based on linguistic observations.

Overview

The first chapter begins with the introduction of the relevant formal systems. After introducing Tree Adjoining Grammars (TAGs) and various extensions such as the substitution operation and feature structures, I recapture the relevance of TAGs in Natural Language Processing. This is followed by a review of other formal systems that subsume TAGs, including a generalization of TAGs, namely Multi-Component TAGs (MC-TAG) and a general class of rewriting systems, namely Linear Context Free Rewriting Systems (LCFRS). This chapter also includes a short review of the automata model for TAGs and finally introduces a new automata model for TAGs, namely Two Stack Automata (2-SA) and its generalizations (n-SA), one of which is also a candidate for an automaton model for LCFRS.

The next chapter looks at phenomena from natural language which challenge the descriptive adequacy of TAGs. Two phenomena from the literature are reviewed, the first is a Scandinavian extraction phenomenon (section 2.1) which is argued to be beyond the strong generative capacity of TAGs. The second phenomenon is number names in Chinese (section 2.2) which is argued to be beyond the weak generative capacity of TAGs. In sections 2.3 and 2.4 I then investigate some of the properties of so-called free word order languages such as German and Korean with respect to the adequacy of TAGs as a descriptive system for their syntax. Especially the phenomenon of Long-Distance Scrambling (LDS) is investigated.

In chapter 3, I show how a newly introduced formalization of the LDS phenomenon can be shown to be beyond the capacity of LCFRS. As common methods of analysis such as weak and strong generative capacity fail to yield such a result (section 3.1), I introduce in section 3.2 a new means to compare grammar formalisms, the "derivative generative capacity." By using the derivative generative capacity, it is shown in section 3.3 that the formal phenomenon of Long-Distance Scrambling is beyond a whole range of formal systems (the Linear Context-Free Rewriting Systems) which subsume TAGs. Starting from this result, I investigate in chapter 4 two possible extensions of the TAG formalism that keep most of the advantages of TAGs and both capture the LDS phenomenon.

The first approach (section 4.1), the Free Order TAGs (FO-TAG) allows for less restricted word orders by separating the ordering relation (the precedence relation) from the dominance relation in the elementary trees of a TAG. This allows to underspecify the precedence relation in the FO-TAG trees. The specific variant of this separation in FO-TAGs elegantly captures the LDS phenomenon but introduces new problems.

The second approach (section 4.2), the Multi-Component TAGs with Dominance

Links (MC-TAG-DL) allows for less restricted word orders by splitting the elementary trees of a TAG into a finite number of trees which can be adjoined at different places. Such a sequence of trees is called a multi-component tree sequence. The splitting of elementary trees into subtrees can be viewed as a relaxation of the dominance relation between these subtrees from immediate dominance. Dominance links in a tree sequence can be used to ensure that (non-immediate) dominance relations still hold between the subtrees. The extended domain of locality is preserved by the constraint that all trees of a tree sequence have to be adjoined simultaneously. As the name suggests, the resulting formal system is closely related to MC-TAGs (see section 1.3). The MC-TAG-DL formalism turns out to be more powerful in terms of descriptiveness, since the constraints that govern the phenomena which are problematic for the FO-TAG approach seem to be of a structural nature and as such are more easily captured with the more complex structures provided by an MC-TAG-DL analysis.

Choosing MC-TAG-DL as the formal system best suitable for the representation of syntactic descriptions I proceed to develop a compact and efficient organization of a grammar in this formal system.

As most of the advantages and properties of TAGs are preserved in MC-TAG-DL, the results of the following chapters apply to TAGs as well as to MC-TAG-DL. In fact, in order to give a clearer presentation, the definitions for metarules and inheritance networks are first given for regular TAGs and then extended to MC-TAG-DL.

Chapter 5 compares the MC-TAG-DL approach to handle unbounded dependencies as the LDS phenomenon with approaches in Head-driven Phrase Structure Grammar (HPSG) which is a prominent example of unification based grammar formalisms. This is of particular interest as attempts are made to use TAGs as a target formalism for compiling HPSG grammars. In section 5.4 I discuss how MC-TAG-DL can also be successfully used for this purpose.

After having established MC-TAG-DL as the underlying formalism, the following chapters are concerned with a compact and efficient representation of syntactic knowledge (i.e. a grammar) which is given in this formalism. This will finally lead to the definition of HyTAG.

Traditionally, a TAG grammar for a natural language consists of one (potentially very large, see [ABCS90]) set of elementary trees that has only little internal organization. The elementary trees are grouped in so-called tree families which reflects the fact that lexical items usually select for all the elementary trees of a tree family.

This structure can be made explicit and extended considerably by the introduction of two orthogonal methods for compactifying the grammar: Metarules (chapter 6) and a hierarchical organization of the elementary structures (chapter 7). The inclusion of these methods in the definition of the grammar also has some consequences for the design of the lexicon.

Chapter 6 introduces metarules first for TAGs and finally for the MC-TAG-DL formalism. Metarules are rules (or rather relations) that state generalizations about various syntactic forms of lexical items. As an example, all verbs that can form a main clause can also be the governing verb of a relative clause. This is true regardless of the subcategorization frame of the verb. In current TAG grammars, for every subcategorization frame there is always a main clause elementary tree *and* a relative clause elementary tree for each subcategorization frame. With a metarule that captures the relation between the main clause elementary trees and the relative clause elementary trees as one general rule, the grammar can be given a much more compact representation.

Section 6.1.3 discusses an important problem which has occurred in a wide variety of other formal systems that made use of such rules on a meta-level. In general, all of these systems become undecidable unless the application of metarules is restricted in some, usually quite arbitrary, way. In section 6.1.4, I show how a restriction of the application of metarules in the MC-TAG-DL formalism can be derived from the basic principles underlying the MC-TAG-DL formalism (and also the TAG formalism). The formal power of the MC-TAG-DL formalism remains unchanged by the addition of metarules.

After first defining metarules formally for regular TAGs in section 6.1.5, the following sections discuss the adaption to TAGs with feature structures and FO-TAGs. Section 6.2 gives the necessary extensions for the MC-TAG-DL formalism.

Quite a large number of such relations have been found (and have been defined in other grammar formalisms before) that cover very different syntactic phenomena. The appendix A gives some more detailed examples of metarules in an MC-TAG-DL framework. The addition of metarules to a grammar of the type given in [ABCS90] can yield substantially more succinct grammars which naturally can be build and maintained much more efficiently.

Chapter 7 looks at another avenue of compactification of the grammar, a hierarchical organization of the grammar entries (i.e. the elementary tree sequences). In a hierarchical system, more specialized entries can inherit information from more general entries. As an example, one obvious case of specialization are (transitive) verbs with additional objects that can inherit any information (partial tree structures, applicable metarules etc.) from less specialized grammar entries (e.g. the intransitive verb).

Section 7.2 defines inheritance networks and the integration of metarules into a framework based on regular TAGs. The extensions necessary for the MC-TAG-DL formalism are given in section 7.3. The resulting formalism which integrates MC-TAG-DL, metarules and inheritance networks is called HyTAG.

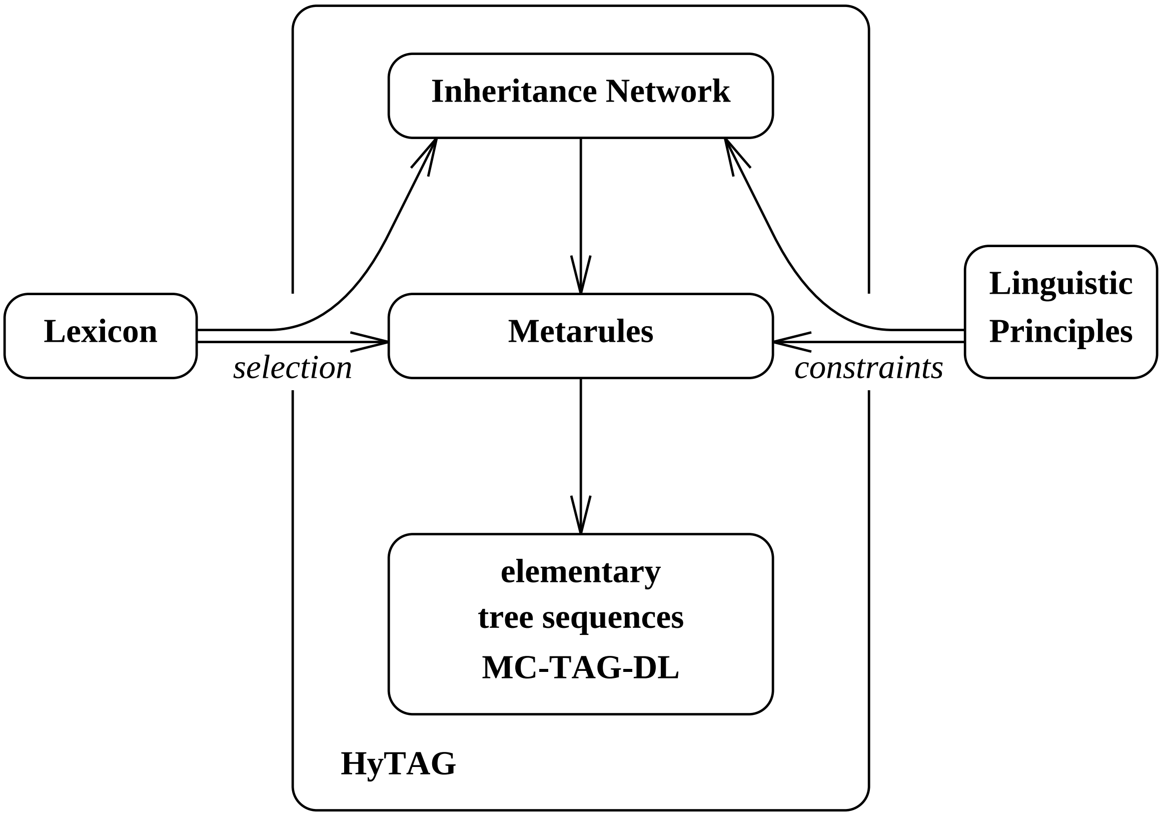

Figure 1: Various knowledge sources in HyTAG and the derivation of elementary structures.

Chapter 7 concludes with a discussion of the appropriateness of using either metarules or an inheritance network for different types of redundancies in a natural language grammar.

Figure 1 sketches the organization of the syntactic knowledge in the HyTAG formalism. It shows how elementary structures of an MC-TAG-DL grammar are derived from the various knowledge sources.

In chapter 8 the main results of the thesis are summarized and topics for further work which are initiated by this thesis are discussed.

Finally, the appendix gives a small fragment of German syntax as an example of the application of the proposed HyTAG architecture.

Acknowledgements

There are many people I have to thank and without whom this thesis would not have been possible[4]. The following list is certainly incomplete.

First, I wish to thank my advisor, Prof. Wolfgang Wahlster and also Prof. Aravind Joshi who has been my advisor at the University of Pennsylvania. Their insight in the field and support and confidence in my work have been of invaluable help.

This work is in large parts the result of my cooperation with Owen Rambow of the University of Pennsylvania, whom I thank for endless hours of fruitful discussion and also for valuable comments and proof-reading.

For his intensive proof-reading and valuable suggestions I thank Frieder Strauß of the University of Freiburg.

I also want to thank Anne Kilger, Karin Harbusch and Wolfgang Finkler of the WIP project at the DFKI for their comments and suggestions.

For important discussions I want to thank Michael Niv and Young-Suk Lee of the University of Pennsylvania.

I thank Gisela Pitsch of the University of Saarland for valuable discussions and proof-reading.

Among other people who have helped me in various ways are K. Vijay-Shanker, David Weir, Hans Uszkoreit, Tony Kroch, Bob Frank, Beatrice Santorini, Anne Abeille, Yves Schabes, B. Kasper, Klaus Netter and numerous others who will hopefully forgive me for forgetting to mention them.

Finally, I am indebted to the "Studienstiftung des deutschen Volkes" (the German National Fellowship Foundation) which made it possible for me to spend the year of 1990 at the University of Pennsylvania in Philadelphia and the "Deutsche Forschungsgemeinschaft" (DFG) which supported my work at the University of the Saarland in Saarbrücken with a grant from the "Graduiertenkolleg Informatik." At both places, I have met wonderful people, without whose cooperation and discussion this work would not have been possible.

And last but not least I wish to thank my family, my friends Frieder and Tommy and finally Barbara for their continuing support and encouragement.

[4] Of course all errors of any kind remain my responsibility.

Chapter 1

Relevant Formal Systems

This chapter introduces a number of formal systems which the following chapters are based on. After introducing Tree Adjoining Grammars (TAGs) and various extensions such as the substitution operation and feature structures, I briefly recapture the relevance of TAGs in Natural Language Processing. This is followed by a review of other formal systems that subsume TAGs, including a generalization of TAGs, namely Multi-Component TAGs (MC-TAG) and a general class of rewriting systems, namely Linear Context Free Rewriting Systems (LCFRS). This chapter also includes a short review of the automata model for TAGs and finally introduces a new automata model for TAGs, namely Two Stack Automata (2-SA) and its generalizations (n-SA), one of which is also a candidate for an automaton model for LCFRS.

1.1 Tree Adjoining Grammars (TAG)

This section begins with the basic definitions of Tree Adjoining Grammars (TAG, [JLT75]) and sketches the application of TAGs in natural language processing. After formally defining the notions of grammar, derived language etc., I summarize some of the formal properties of TAGs. Finally some examples from a TAG grammar for English are given to show the linguistic applicability of TAGs. Also, the notion of lexicalized grammars will be introduced. As TAGs have originally been used to describe the English language, the first examples will be English sentences though later examples will be in German.

1.1.1 Formal Definitions

TAGs have originally been introduced as "Tree Adjunct Grammars" by Joshi, Levy and Takahashi in [JLT75], the definitions in this chapter are based on [Jos87a]. The notion of substitution in TAGs has been introduced in [SAJ88]. Feature structures for TAGs have been introduced in [VS87] and [VSJ88].

As the name suggests, a TAG is a tree-rewriting formalism (in contrast to classic grammar formalisms for formal languages like string-rewriting systems as, e.g. context-free grammars). Starting from a finite set of so-called *elementary trees*, a TAG constructs phrase-structure trees by means of two combination-operations, called *adjoining* and *substitution*. Substitution is basically the same operation which is used in context-free grammars; it is the expansion of a non-terminal leaf-node. Adjunction can be seen as the "insertion" of one tree into another. Figures 1.1 and 1.2 give a sketch of both operations.

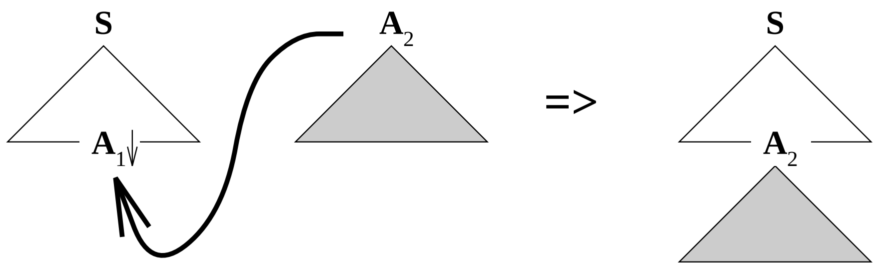

Figure 1.1: The substitution operation.

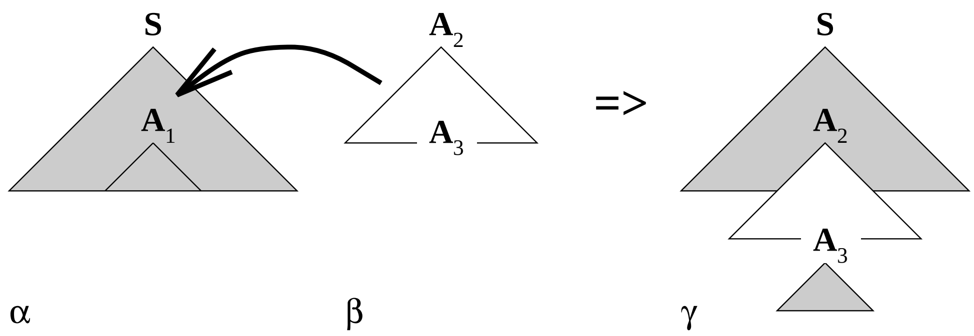

Figure 1.2: The adjunction operation.

As the resulting trees implicitly have a fixed ordering of all the leaves, every phrase-structure tree that is constructed in this way also has a unique leaf-word. This set of all such leaf-words is defined as the string language of a TAG.

Formally, a TAG is defined as follows:

Definition 1.1.1 (Tree Adjoining Grammar (TAG))

A **Tree Adjoining Grammar** *(TAG) G is a five-tuple $G = (N,T,S,I,A)$ where*

- *N is a nonempty, finite set of* nonterminal symbols*,*
- *T is a nonempty, finite set of* terminal symbols*, $N \cap T = \emptyset$,*
- *S, the* start symbol *is a distinguished element of N,*
- *I is a nonempty, finite set of* initial trees *and*
- *A is a finite set of* auxiliary trees*.*

An **initial tree** *is a (named) tree where all inner nodes are marked with a nonterminal symbol and all leaves are marked with either a terminal symbol or the empty string ϵ or a nonterminal symbol that is marked with the symbol "$\downarrow$", the* substitution mark*. At least one initial tree of G must have the start symbol S at its root.*

An **auxiliary tree** *is a (named) tree where all inner nodes are marked with a nonterminal symbol, one leave the* **foot-node** *is marked with the same nonterminal symbol as the root and all other leaves are marked with either a terminal symbol or the empty string ϵ or a nonterminal symbol that is marked with $\downarrow$.*

The set of all auxiliary trees and all initial trees is called the set of **elementary trees***.*

Any nonterminal node may be marked with one of the following three **Constraints***:*

- *NA, the* null adjoining *constraint or*
- *OA, the* obligatory adjoining *constraint followed by a set of names of auxiliary trees*[1] *or*
- *SA, the* selective adjoining *constraint, followed by a set of auxiliary trees.*

The two operations of a TAG are defined as:

Definition 1.1.2 (Substitution, Adjunction)

Given an initial tree α_1 and an initial tree α_2 where a leave node n of α_1 is a nonterminal symbol A marked with the substitution mark and the root of α_2 is marked with

[1] If this set of trees is the whole set of auxiliary trees, it can be omitted, i.e. the OA constraint without a set of trees implies that any tree may be adjoined.

the same nonterminal symbol A, then the result α of the **substitution** *of α_2 into α_1 at node n is defined as follows:*

α consists of α_1 where the node n is replaced by α_2.

Given a tree α with a node n marked with a nonterminal symbol A and a tree β with root node A (and by definition with foot node A), the result γ of the **adjunction** *of β into α at node n is defined as follows:*

If α_1 is the complete subtree of α below and including node n and α_2 is the remaining tree plus the node n, then α consists of α_2 where the node n is replaced with the auxiliary tree β where in turn the foot node is replaced with α_1.

The adjunction of tree β at node n is not *defined if the node n is marked with the NA constraint or with the OA or SA constraint such that β is not in the set of auxiliary trees listed after the OA or SA constraint.*

For a given TAG, the set of all trees that can be derived with it is defined as:

Definition 1.1.3 (Tree-set of a TAG)

The **Tree-set** *T(G) of a TAG G is defined as:*

All trees γ which are derived from an initial tree α of G with the start symbol S at the root by repeatedly adjoining and substituting elementary trees β from G such that γ has no nodes which are marked with the substitution mark or the OA constraint are in T(G).

Finally, from the tree-set, the language of a TAG can be defined:

Definition 1.1.4 (Language of a TAG)

The **String-Language** *L(G) of a TAG G is the set of all leaf-words of trees in the tree-set T(G).*

The set of all languages which can be derived by some TAG is defined as:

Definition 1.1.5 (The Tree Adjoining Languages, TAL)

The set TAL, the **Tree Adjoining Languages** *is the set of all languages L(G) that are the string-language of a TAG G.*

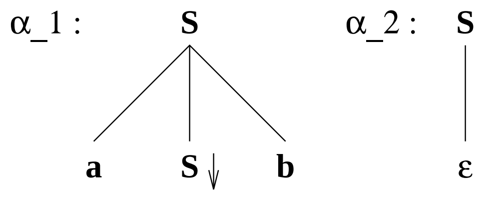

Figure 1.3: Example TAG grammar G_1 with substitution.

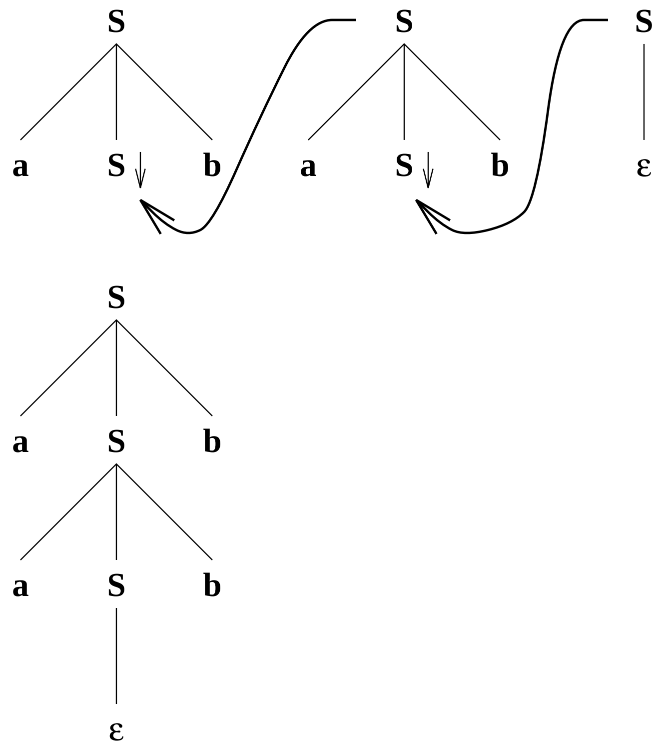

Figure 1.4: Derivation of $aabb$ from G_1.

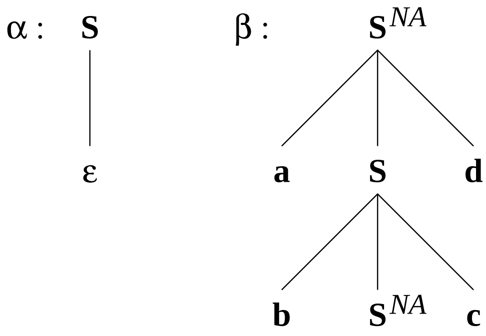

Figure 1.5: Example TAG grammar G_2 with adjunction.

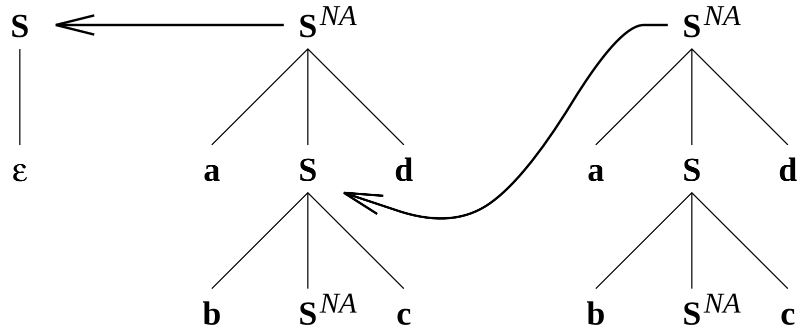

Figure 1.6: Derivation of *aabbccdd* from G_2.

The TAG G_1 of figure 1.3 is an example for the use of the substitution operation:

Figure 1.4 shows the derivation of *aabb*. The language $\mathrm{L}(G_1)$ derived by G_1 is $\{a^n b^n \mid n \geq 0\}$, known to be a context-free language. It turns out, that the substitution operation does not provide any generative power beyond that of context-free grammars. A TAG without auxiliary trees (i.e. the set of auxiliary trees is empty and thus the only applicable operation is substitution) is always (weakly) equivalent to a context-free grammar. The reason for including the substitution operation in the definition is its usefulness in the linguistic application of TAGs, which is discussed in more detail in section 1.1.3.

Only the adjoining operation extends the generative power of the formalism beyond context-freeness. The TAG G_2 of figure 1.5 is an example for the use of adjoining.

Figure 1.6 shows the derivation of *aabbccdd*. The language derived by G_2 is $\mathrm{L}(G_2)$ $= \{a^n b^n c^n d^n | n \geq 0\}$, known not to be a context-free language. This is the classic example for the extended generative power of TAGs. Note that the use of the null adjoining constraint is crucial in this example. It can be shown, that a definition of TAGs without constraints leads to a formalism that has a weaker generative power, see [Bec90] for details.

1.1.2 Formal Properties of TAGs

This section briefly summarizes some of the formal properties of TAGs.

First of all, TAL is a class of languages between CFL and CSL, the sets of context-free and context sensitive languages respectively. To see the first proper inclusion, consider the language $L_4 := \{a^n b^n c^n d^n | n \geq 0\}$ which is not a context-free language (proof by pumping lemma). However, L_4 is the language of the example TAG grammar G_2 above. Also, it is clear from the discussion above, that TAL subsumes CFL. Also, every language in TAL can be derived by some context sensitive grammar. This lead to the notion of TAG being a "mildly context sensitive" formalism, as characterized by Joshi in [Jos85]. The actual conditions on a formalism to be mildly context sensitive are polynomial parseability, constant growth property and the ability to generate limited cross-serial dependencies which all hold for TAGs.

Vijay-Shanker has shown in [VS87] that TAGs are an *Abstract Family of Languages*, i.e. they are closed under union, concatenation, Kleene-star, intersection with Regular Languages, homomorphism, inverse homomorphism and substitution. [VS87] also gives a pumping lemma for TALs. Also, a number of normal forms for TAGs have been defined, see [VS87, Har89, Gua92]

The first parsing algorithm for TAGs is given in [VSJ85]. It is based on the CKY parsing algorithm for context-free grammars and has a time complexity of $O(n^6)$. It requires the grammar to be in a normal form. The parsing algorithm of [Har89] takes a different approach, but also requires a normal form. The time complexity is also $O(n^6)$. In [Pol93] an Earley-style parser for general ID/TLP-TAG (which are strongly equivalent to TAGs but include linear precedence rules) is presented with time complexity $O(n^6)$[2]. The first general parsing algorithms for TAGs are also based on the Earley algorithm for context-free grammars. They are presented in [SJ88] and [Sch90]. The first algorithm is the first to have the valid prefix property (VPP) and has time complexity of $O(n^9)$, the second algorithm lacks the VPP but has time complexity $O(n^6)$. In [Gua92] a parser for CCFG(2) (see section 1.4) with time complexity $O(n^5)$ is presented which also requires a grammar in a normal form. As CCFG(2) and TAG derive the same languages (and there is a general construction from a TAG to an equivalent CCFG(2)) this parser can also be used for TAGs.

1.1.3 TAGs for Natural Language Processing

In [KJ85], a number of phenomena of English are analyzed in the framework of Tree Adjoining Grammars. Since then grammars for English, French and German have

[2]Currently a version with time complexity $O(n^5)$ is under investigation.

been developed at the University of Pennsylvania, at the Universite de Paris 7 and at the DFKI Saarbrücken resp. In this introductory section, only some simplified examples from the LTAG project will be given.

However, at first another formal definition – for *feature structures* – is needed for the examples.

TAGs with Feature Structures

In the literature, there exist two different approaches to define feature structures for TAGs.

The F-TAG variant (developed by Vijay-Shanker in [VS87, VSJ88]) is discussed in more detail below. It associates two feature structures with every node and demands that they unify only at the end of the derivation process. That allows for coding unification constraints (such as OA) in the feature structures.

The U-TAG variant (developed in [BPSH91]) associates only one feature structure with every node and is based on a rather complex unification procedure which takes place immediately at every adjoining. This has other advantages, e.g. a better control (i.e. an early recognition) of already illicit intermediate structures in a derivation, especially if used for text-generation.

Both definitions are discussed in detail in the work of [VS87] and [BPSH91], here I will only sketch the necessary extensions of the TAG framework. In this thesis, I will provide all examples and definitions in the style of FTAG. However, this choice is somewhat arbitrary, since an adaption to U-TAG can easily be carried out in all cases.

In the FTAG formalism as defined by [VS87] and [VSJ88], two feature structures are attached to each nonterminal node in an elementary tree, the so-called top and bottom feature structures. When an auxiliary tree β is adjoined at a node labeled A, the top feature structure of A is unified with the top feature structure of the root node of β and the bottom feature structure of A is unified with the bottom feature structure of the foot node of β as shown in figure 1.7. After a derivation is completed, the top and bottom features of every node must unify. The derived tree is in T(G) only if this unification is successful.

This approach can been extended to the substitution operation (see [VSJ90]) as shown in figure 1.8: A substitution node A has only one (top) feature structure which is unified with the top feature structure of the root node of the tree which is substituted at A.

The extended domain of locality of an elementary tree allows to express dependencies

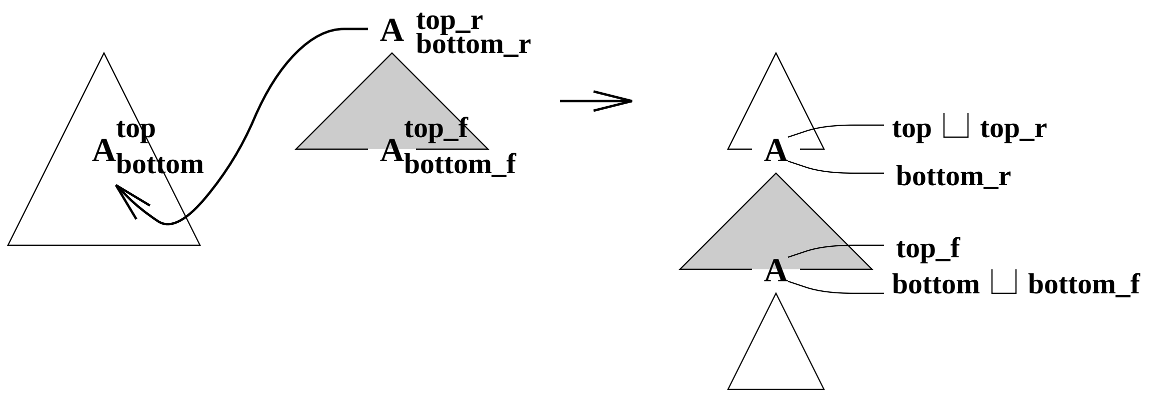

Figure 1.7: Unification of feature structures when adjunction occurs.

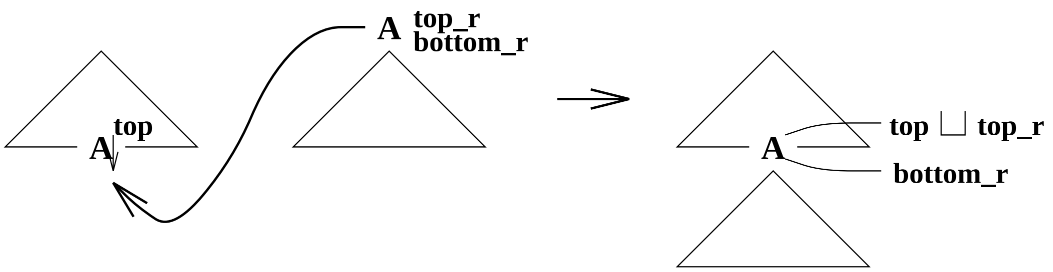

Figure 1.8: Unification of feature structures when substitution occurs.

between feature structures of nodes at different levels of an elementary tree. Thus the passing (percolation) of feature structures from two such nodes to a common ancestor is unnecessary in an FTAG. As in the PATR-II system (see [Shi84]), the dependencies between feature structures within an elementary tree can be expressed as unification equations. The syntax for feature equations (given here in a BNF) is fairly simple:

- EQUATION ::= PATH = PATH | PATH = value
- PATH ::= node-name T-OR-B-OR-EPS : FEATURES
- T-OR-B-OR-EPS ::= . t | . b | ϵ
- FEATURES ::= ϵ | $<$ feature$^+$ $>$

Consider figure 1.9, which stands for verbs like "sleep" in questions for the subject (e.g. "Who sleeps?") As an example, the equation

```
NP:<trace> = NP_0.b:<trace>
```

creates a link between the NP node and the NP_0 node (to which the subject noun phrase has "moved") over two levels in the tree without first percolating the feature value to the common ancestor node S_c.

S_c.t:<mode> = ind
S_c.b:<comp> = none
S_c.b:<mode> = VP.t:<mode>
S_c.t:<inv> = -
S_r.b:<mode> = VP.t:<mode>
S_r.b:<comp> = none
S_r.b:<tense> = VP.t:<tense>
S_r.t:<inv> = -
NP:<trace> = NP_0.b:<trace>
NP:<wh> = +

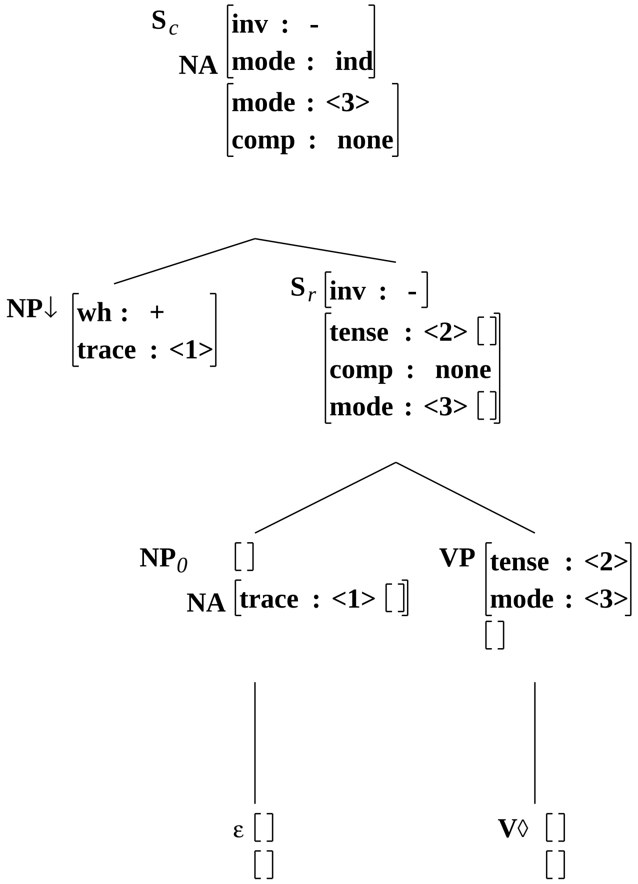

Figure 1.9: Initial tree for verbs like “sleep” in questions like “Who sleeps?” with feature structures and feature equations.

In [VS87], RFTAG, a restricted version of FTAG is defined which rules out recursive feature equations, and thus recursive feature structures. Also, the set of values is defined to be finite. The resulting feature structures are therefore over a finite domain. Even though the full FTAG formalism is undecidable, the restricted version RFTAG is not only decidable but also equivalent to TAG. Besides being sufficient for most linguistic purposes, the restricted version is also in the spirit of the way recursion is captured in TAG, namely in the auxiliary trees and not in any other mechanism. See also the extensive discussion in chapter 6.

As worked out in [VS87], the adjoining constraints can be expressed by the feature structures, which can have great benefits. E.g., an obligatory adjoining constraint at a node A for a specific class of auxiliary trees can be expressed either by using the OA constraint and explicitly enumerating all trees that are admissible, or more elegantly, by using a feature which has different values in the top and bottom features of A. Since the top and bottom features of A will thus not unify, some elementary tree must be adjoined which has the top value in the top feature structure of its root node and the bottom value in the bottom feature structure of its foot node. By giving a tree these feature values in its root and foot node, the tree is characterized as belonging to the set of admissible auxiliary trees for adjoining at A, without explicitly listing the tree at node A. This also allows for an easier maintenance of a larger grammar.

1.1.4 Lexicalized Grammars

An important aspect of TAGs is that it can be assumed in the definition that every tree must have at least one leaf with a terminal symbol (see [Sch90]). Such a grammar is called lexicalized, according to the following definition of [Sch90]:

Definition 1.1.6 (Lexicalized Grammar (Lexicalized TAG))

A grammar (TAG) is lexicalized *if it consists of:*

- *a finite set of structures (the elementary trees) each associated with a lexical item; each lexical item will be called the* anchor *of the corresponding structure; the structures define the domain of locality over which constraints are specified;*
- *an operation or operations (adjoining and substitution) for composing the structures.*

It is also shown in [Sch90] that

Theorem 1.1.1 (Schabes)

If G is a finitely ambiguous TAG, s.t. $\epsilon \notin L(G)$ then there exists a lexicalized TAG G_{lex} which generates the same language and tree set as G.

TAGs can also be developed as the result of lexicalizing context-free grammars, see the work of [SAJ88, Sch90]. Lexicalized grammars are extremely well suited for natural language processing. As will be developed in more detail throughout this thesis, lexicalized grammars have a number of advantages:

- They allow for a clear separation of the lexicon from the grammar (syntactic part of the language description).
- Furthermore, a lexicalized TAG also allows for a very efficient parsing methodology: Since every elementary tree carries at least one terminal (lexical item), in parsing a sentence only those trees that actually hold a lexical item from the sentence need to be looked at. All other elementary trees from the grammar can be disregarded during the parsing of the sentence. This keeps the actual grammars comparatively small – an important issue if the parsing algorithm has a high complexity with regard to the grammar size.
- Also, an important aspect which will be discussed in chapters 6 and 7 of the thesis, namely the compactification of the grammar by use of metarules and an inheritance network can take advantage of a lexicalized grammar.

1.2 Automata

Before introducing the new n-SA automaton model in section 1.2.2, section 1.2.1 reviews the classic automaton model for TAGs, the Embedded Push-Down Automaton (EPDA) and its generalization.

Then, in section 1.2.2, I introduce and define a new class of automata (Push Down Automata with n stacks, abbreviated as n-SA). The ultimate goal is to give a new characterization of the LCRFL class of languages. In particular, I show that the 2-SA give a new automaton model for Tree Adjoining Grammars (TAG) and I discuss the properties of the general class of n-SA.

1.2.1 Embedded Push-Down Automata (EPDA)

Before introducing the new automaton model, I review the definition of Embedded Push-Down Automata (EPDA), an automaton model for TAGs that has been introduced in [VS87] and [Wei88][3]. Even though, the EPDA (especially the hierarchy of automata that [Wei88] discusses) form a good generalization of the Push-Down Automata (PDA), the resulting operations include rather complex rewritings of a complex storage structure (i.e. multiply embedded stacks). In contrast, n-SA have a simpler storage structure (a number of parallel stacks) with the operations known from the PDA. However, in an n-SA, a number of constraints as to when these operations can be carried out apply.

I assume that the reader is familiar with the concepts of the classic Push-Down Automaton (e.g. see [HU79]). As an example, a second-order EPDA has an input tape that is read from left to right, and a store (an embedded push-down store) that consists of a stack of push-downs. There is a pointer that always points to the topmost symbol on the topmost stack[4]. Figure 1.10 shows the structure of a second-order EPDA and figure 1.11 shows the resulting configuration after a move.

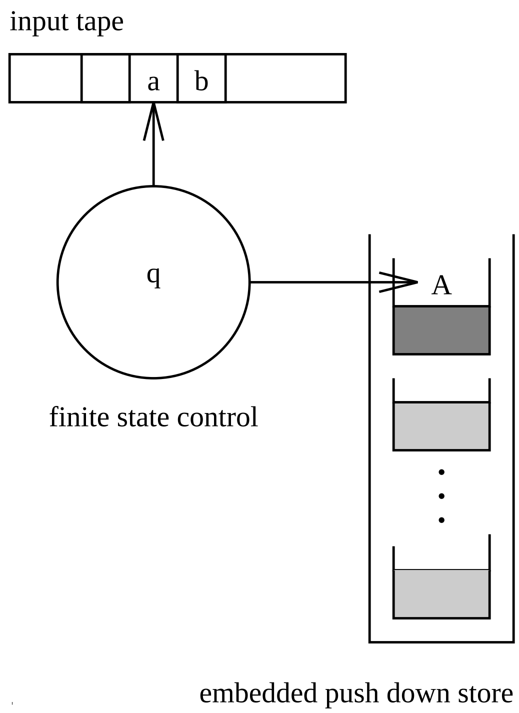

Figure 1.10: Structure of a second-order EPDA.

Informally, the second-order move of the EPDA can be seen as a generalization of a PDA move as follows: The topmost structure of the stack (the topmost stack

[3][Wei88] uses the term NPDA.

[4]In the general case this is the topmost symbol of the most deeply embedded topmost stack of the store.

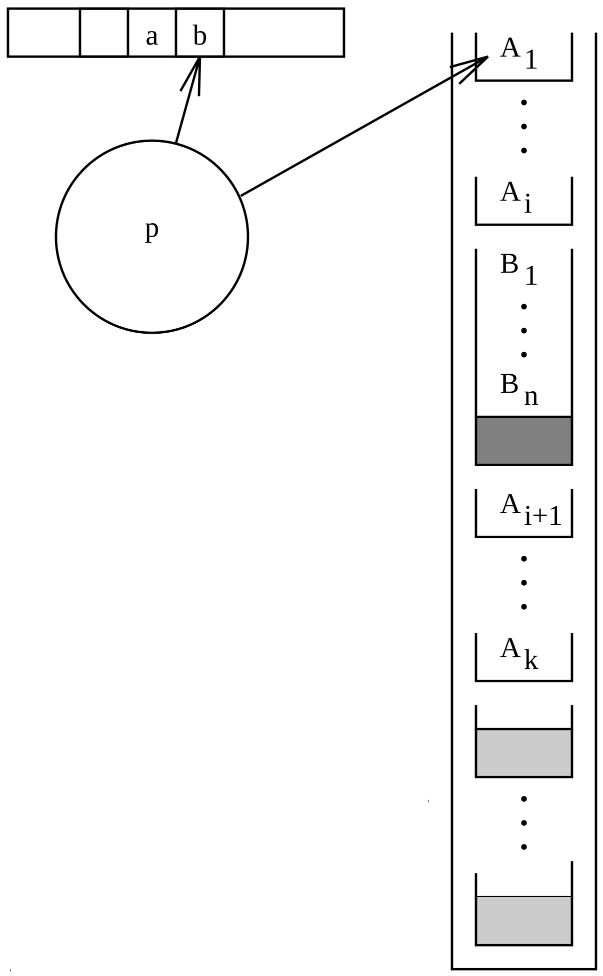

Figure 1.11: Configuration after a move.

of the EPDA; the topmost symbol of the PDA) is removed and somehow stored intermediately. Then finitely many low-level structures are pushed on the store (stacks are pushed in an EPDA; stack symbols are pushed in a PDA), the intermediately stored topmost structure is pushed on the store (the former topmost stack in an EPDA; possibly the former topmost stack symbol in an PDA), and finally, finitely many low-level structures (stacks in an EPDA; stack symbols in a PDA) are pushed on the store.

Formal Definitions and Properties

The following definitions are adapted from [Wei88].

Definition 1.2.1 (EPDA)

An **i-th order EPDA** *M is a seven-tuple* $M = (Q, \Sigma, \Gamma, q_0, Z_0, F, R)$ *where*

- *Q is a nonempty, finite set of* states*,*
- *Σ is a nonempty, finite set, the* input alphabet*,*
- *Γ is a nonempty, finite set, the* stack alphabet*,*
- *$q_0 \in Q$ is the* initial state*,*
- *$Z_0 \in \Gamma$ is the* start symbol*,*
- *$F \subseteq Q$ is the set of* final states*, and*
- *R is a finite set of i-th order rewrite rules as described below.*

The rewrite rules are defined recursively: For i=1 (regular PDAs) a rewrite rule (a first-order move) has one of the following forms,

$< q, a, [A|x^1] > \rightarrow < q', [A_1, \ldots, A_n|x^1] >$

$< q, a, x^1 > \rightarrow < q', [A_1, \ldots, A_n|x^1] >$

where $q, q' \in Q, a \in \Sigma \cup \{\epsilon\}$, and $A, A_1, \ldots, A_n \in \Gamma, n \geq 0$ and x^1 is a variable denoting a 1st-order stack.

Assume that α^i and β^i are such that

$< q, a, \alpha^i > \rightarrow < q', \beta^i >$

is an i-th order move. An (i+1)-th order move has the following form. If β^i is nonempty then

$< q, a, [\alpha^i|x^{i+1}] > \rightarrow < q', [s^i_1, \ldots, s^i_j, \beta^i, s^i_{j+1}, \ldots, s^i_n|x^{i+1}] >$

where $1 \leq j < n$, and $s^i_1, \ldots, s^i_n$ are constant i-th order pushdowns. If β^i is empty then

$< q, a, [\alpha^i|x^{i+1}] > \rightarrow < q', [s^i_1, \ldots, s^i_n|x^{i+1}] >$

where $1 \leq j < n$, and $s^i_1, \ldots, s^i_n$ are constant i-th order pushdowns.

A special case, the second-order embedded pushdowns, accept exactly the Tree Adjoining Languages (TAL). This is shown in [Wei88]. This fact will be used in the next section to show that 2-SA are also a automaton-model for TAGs. In fact, I show the equivalence of 2-SA and second-order EPDA.

To make the moves of the second-order EPDA easier to read, I will drop the variable x^2 from the rules and write the result of the above defined move as four-tuples, containing the new state, the stacks added "on top of the old topmost stack", the modified old topmost stack and the stacks added "below the old topmost stack":

$< q, a, \alpha^i > \rightarrow < q', [s^i_1, \ldots, s^i_j], \beta^i, [s^i_{j+1}, \ldots, s^i_n] >$

In their generalized form, the EPDA constitute a 'geometric hierarchy' in the following sense:

For any i > 0 there is an i-th order EPDA that recognizes the language $L_{2^i} = \{ a^n_1 \ldots a^n_{2^i} \mid n \geq 0 \}$.

A construction for these EPDA is outlined in [Wei88].

1.2.2 N-Stack Automata (n-SA)

In this section, we introduce and define a new class of automata (Push Down Automata with n stacks, abbreviated as n-SA)[5]. The ultimate goal is to give a new charaterization of the LCRFL class of languages. In particular, I show that the 2-SA give a new automaton model for Tree Adjoining Grammars (TAG) and I discuss the properties of the general class of n-SA.

In the simplest cases (0-SA and 1-SA), the languages that are accepted by the automata are regular and contextfree languages, respectively. A more complex case is the case of a 2-SA which accept TALs. This is shown in section 1.2.2. As discussed in section 1.2.2, the n-SA constitute infinite hierarchies which correspond to others in the class LCFRL.

The 2-SA corresponds very closely to the second-order EPDA. But unlike the general EPDA, which allows highly complex operations on a complex, multiply nested stack, all n-SA allow push and pop operations only on the top of their (parallel) stacks. So n-SA trade simpler operations against an (also simpler), but expanded storage structure.

Unless explicitly stated, from now on I will use EPDA to denote a second-order EPDA.

Introduction to n-SA

N-stack PDA (abbreviated as n-SA) are similar to standard PDA but can have more than one, actually n, stacks. The first stack is used just like the stack in a PDA. During a move of the n-SA, one element can be popped from the stack or a string of symbols can be pushed onto the stack. The input tape can be read once from left to right, i.e., the symbols can be consumed one at a time.

---

[5]N-SA have first been introduced in [Bec92a], see also [Bec92b].

However, the additional n-1 stacks of an n-SA can be used only in a restricted manner. In most circumstances, only push operations are allowed. Only if the first stack is empty, the top element(s) of some additional stack(s) can be popped. To make it easier to check for empty stacks, each stack has a special "bottom symbol" (which is not part of the regular stack alphabet) at its bottom. In the following, an "empty" stack refers to a stack with only the "bottom symbol" on it.

An additional difference to regular push down stores is the creation of new stacks. At any point, new empty stacks can be "pushed" on top of all n stacks simultaneously and the automaton continues to work with the new empty stacks. The previous set of stacks can only be reached if all current stacks are empty. Then all current, empty stacks are deleted and the automaton uses the previous stacks. So in a way, the stacks of an n-SA are actually nested stacks, as in [Aho69]. But in an n-SA the stack pointers always point to the top symbol of their topmost stack. Note also that the store of an n-SA is very different from the store of an n-th order EPDA. In an n-SA there is no operation that affects the whole topmost stack, e.g. an n-SA (unlike an n-th order EPDA, $n > 1$) does not have the ability to "push under the topmost stack" (see section 1.2.1).

In general, an automaton that has two or more stacks will be Turing-equivalent because two stacks can simulate a Turing-tape as follows: The first stack represents the part of the tape to the left of the head, the second stack represents the right part and the tape symbol under the head of the Turing machine is, say, the top of the first stack. Now read and write operations can be simulated on the first stack and move operations are simulated by popping a symbol from one stack and pushing it on the other.

Nevertheless, n-SA are not that powerful because they cannot freely push and pop items from one stack to another and thus cannot simulate a Turing-tape. In fact, as will be shown later, 2-SA accept exactly the Tree Adjoining Languages (TAL), and thus accept only a subset of the context sensitive languages.

For the case of 2-SA, the similarity to EPDA [VS87] is striking. However, a 2-SA has easier operations and thus is more intuitive.

In general, the possible moves of an n-SA fall into 4 groups:

- (1) Read a symbol from the input-tape,
- (2) (pop a symbol from) or (push a string onto) Stack 1,
- (3.1) push a string onto a Stack m, m > 1,
- (3.2) pop a symbol from Stack m (m > 1), provided that all Stacks k (k < m) are empty, i.e. they carry only the special bottom symbol
- (4.1) create a new set of (empty) stacks on top of the current stacks,
- (4.2) delete the current set of stacks, provided that all stacks are empty.

The constraint in (3.2) ensures the limited power of n-SA, the moves in (4) give some additional power. The move (4.2) can be integrated into the definition of a move of an n-SA.

Figure 1.12 shows the structure of a 2-SA.

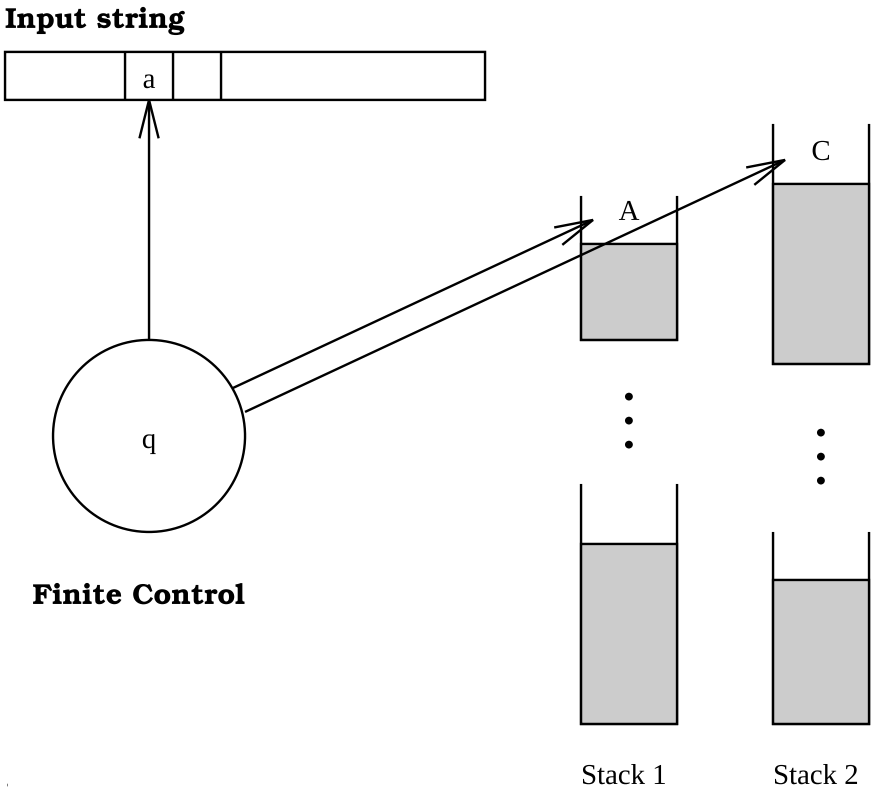

Figure 1.12: Structure of a 2-SA.

Definitions

As a first step, we will now formally define 2-SA. In the following, stacks are notated with their top to the right, nested stacks are delimited by the "$\sqsubset$" sign which is part of the formal description.

Definition 1.2.2 (2-SA)

A **2-SA** *M, is a seven-tuple $M = (Q, q_0, \Sigma, \Gamma, \sqsubset, \gamma_0, \delta)$, where*

- *Q is a nonempty, finite set of* states*,*
- *$q_0 \in Q$ is the* start state*,*

- Σ *is a nonempty, finite set of* input symbols,
- Γ *is a nonempty, finite set of* stack symbols,
- $\gamma_0 \in \Gamma$ *is the* initial stack symbol,
- $\sqsubset \notin \Gamma$ *is the* bottom symbol,
- *and* δ *is the transition function mapping* $Q \times \Sigma \cup \{\epsilon\} \times \Gamma \cup \{\sqsubset\} \times \Gamma \cup \{\sqsubset\}$ *to finite subsets of* $Q \times \chi \times \chi$ *where* χ *is* $\{\sqsubset\} \ \cup \ \Gamma^* \ \cup \ \Gamma^* \circ \sqsubset \circ \Gamma^*$ *(i.e., either an empty stack or a string of stack symbols or a string of stack symbols plus a new stack on top of it).*

Definition 1.2.3 (Configuration)

A **Configuration**[6] *of an n-SA is a (3+n)-tuple of the type* $(q, w_1, \ldots, w_n, u, v)$*, where* uv *is the input string (*u *is the part that has been consumed so far),* w_i *is the content of Stack* i *(i.e. a stack of stacks,* $1 \leq i \leq n$*) and* $q \in Q$ *is a state.*

Moves of a 2-SA

As described in the previous section, the transition function δ is restricted in order to limit the power of the automaton. For the case $n = 2$, the following restrictions apply, the general case ($n > 2$) is discussed in section 1.2.2:

All elements in $\delta(q, a, s_1, s_2)$ with $s_1 \in \Gamma$, $s_2 \in \Gamma \cup \{\sqsubset\}$ and $a \in \Sigma \cup \{\epsilon\}$ are of the form $< q', v, s_2 w >$ where v, w are in $\Gamma^* \ \cup \ \Gamma^* \circ \sqsubset \circ \Gamma^*$ (i.e., if Stack 1 is not empty, the top of Stack 2 can not be popped). If and only if w includes a new stack, v also includes a new stack.

All elements in $\delta(q, a, \sqsubset, s_2)$ with $s_2 \in \Gamma$ are of the form $< q', \sqsubset, w >$ where w is in Γ^* (i.e., only if Stack 1 is empty, the top of Stack 2 can be popped).

The derives relation $\models$ is defined just as in a PDA. Given a configuration $C = (q, w_1 s_1, w_2 s_2, u_1, a u_2)$ and $< q', v_1, v_2 > \in \delta(q, a, s_1, s_2)$, we have

$C \models (q', w_1 v_1, w_2 v_2, u_1 a, u_2)$.

Only if the resulting stacks both have the bottom symbol on the top, i.e.,

$C \models (q', w_1' \sqsubset, w_2' \sqsubset, u_1 a, u_2)$,

both empty stacks can be deleted, i.e., we also have $C \models (q', w_1', w_2', u_1 a, u_2)$.

[6] Also called *Instantaneous description*

Besides this last detail, all other notions from the standard PDA can be transferred to 2-SA. Especially, given a 2-SA M, $N(M)$, the language that M accepts by empty stack, is defined as

$$N(M) := \{w \mid (q_0, \sqsubset \sigma_0, \sqsubset \sigma_0, \epsilon, w) \overset{*}{\models} (q, \epsilon, \epsilon, w, \epsilon) \text{for some } q \in Q\}$$

Most of the definitions extend naturally to n-SA. However, the restrictions to the transition function of general n-SA are discussed below.

2-SA-L = TAL

Now, we can proceed to show the equivalence of 2-SA and TAGs. This is done by simulating a 2-SA by an EPDA and vice-versa. Note that EPDA are equivalent to TAG.

Theorem 1.2.1 *TAL = 2-SA-L*

The inclusion of TAL in 2-SA-L can also be shown directly. At the end of this section, we sketch the main idea of this proof.

The storage of a second-order EPDA is a stack of stacks, just as the stacks of a 2-SA. The moves of an EPDA can roughly be characterized as follows: New stacks can be created above and below the former topmost stack.

Now, in the following simulations, instantaneous descriptions of the EPDA correspond to instantaneous descriptions of the 2-SA in the following sense: Pushing items "below" the old topmost stack of the EPDA is simulated by pushing the items on Stack 2 of the 2-SA and vice-versa.

The definition of EPDA is quite similar to the one given for 2-SA: An EPDA M is a 6-tuple (Q, q_0, Σ, Γ, γ_0, δ) only that the transition function is a mapping from $Q \times \Sigma \cup \{\epsilon\} \times \Gamma$ to $Q \times \Upsilon^* \times \Gamma^* \times \Upsilon^*$ where Υ is the set of pushdown stores of stack symbols, thus $\Upsilon = \Gamma^*$. An instantaneous description of an EPDA is a 4-tuple of the form $< q, s, u, v > \in \; Q \times \Upsilon^* \times \Sigma^* \times \Sigma^*$ where uv again is the input string (u being the part that has already been consumed), s is a stack of stacks of symbols from Γ, delimited by "‡". The language accepted by an EPDA is defined as for 2-SA above.

2-SA-L $\subseteq$ EPDA-L

Each of the possible moves (1) to (4) of a 2-SA (see the enumeration at the beginning of this section) can be simulated by an EPDA:

(1),(2)	same moves on an EPDA,
(3.1)	Create a new stack under the current top-stack initialized with string.
(3.2)	"Stack 1 is empty" now corresponds to the top stack is empty and can thus be deleted and thus makes the simulated Stack 2 below accessible.
(4.1)	Create 2 new stacks on top which are empty (i.e., have a symbol that marks the bottom of a stack)
(4.2)	If a stack is empty (i.e., has the bottom marker "on top"), then delete it.

At any point in the simulation, the EPDA is in on of two sets of states which will be indicated by in the states of the EPDA.

(S1) The top stack of the EPDA is equivalent to the top stack on Stack 1 of the 2-SA. The next stacks of the EPDA have only one symbol; they are identical to the symbols on the topmost stack of Stack 2 of the 2-SA. The stacks of the EPDA below these stacks represent the other pairs of stacks of the 2-SA respectively.

(S2) The top stack of the EPDA is equivalent to the topmost symbol of Stack 2 of the 2-SA. This is only the case if Stack 1 of the 2-SA is empty.

Figure 1.13 sketches the relation between a 2-SA and a simulating EPDA.

Theorem 1.2.2 *2-SA-L* $\subseteq$ *EPDA-L*

Proof: (Sketch)

More precisely, given a 2-SA M_2 as defined above, an EPDA $M_E = (Q^E,\ q_0^E,\ \Sigma,\ \Gamma \cup \{\sqsubset\},\ \gamma_0,\ \delta^E)$ is constructed with states of type $[q{:}\mathrm{S}i{:}s_2]$ where q is the simulated state of M_2, Si indicates whether the simulated topmost stack on Stack 1 is empty and s_2 is used to capture the dependency of δ on the topmost symbol of Stack 2 of M_2.

A transition of M_2 of the form

$\delta(q, a, s_1, s_2)\ \ni\ < q', v, w >$ with $s_1 \in \Gamma,\ v, w \in \Gamma^*$ and $w = a_1, \dots, a_n$

is simulated by a transition of M_E of the form

$\delta^E([q : S1 : s_2], a, s_1)\ \ni\ < [q' : S1 : a_n], \ddagger a_1 \dots \ddagger a_n, v, \epsilon >.$

If $v = v_1 \sqsubset v_2$ and $w = a_1 \dots a_k \sqsubset a_{k+1} \dots a_n$, this is simulated by

$\delta^E([q : S1 : s_2], a, s_1)\ \ni\ < [q' : S1 : a_n], \ddagger a_1 \dots \ddagger a_k, v_1, \ddagger[\sqsubset, s_2] \ddagger a_{k+1} \dots \ddagger a_n \ddagger \sqsubset v_2.$
Note that $[\sqsubset, s_2]$ is a special symbol which encodes the former top-symbol of Stack 2 for later retrieval.

For all $q \in \Gamma$, transitions of the form $\delta^E([q : S1 : s_2], \epsilon, \sqsubset)\ \ni\ < [q : S2 : s_2], \epsilon, \epsilon, \epsilon >$ are added (i.e., a topmost stack of the EPDA which simulates an empty topmost stack

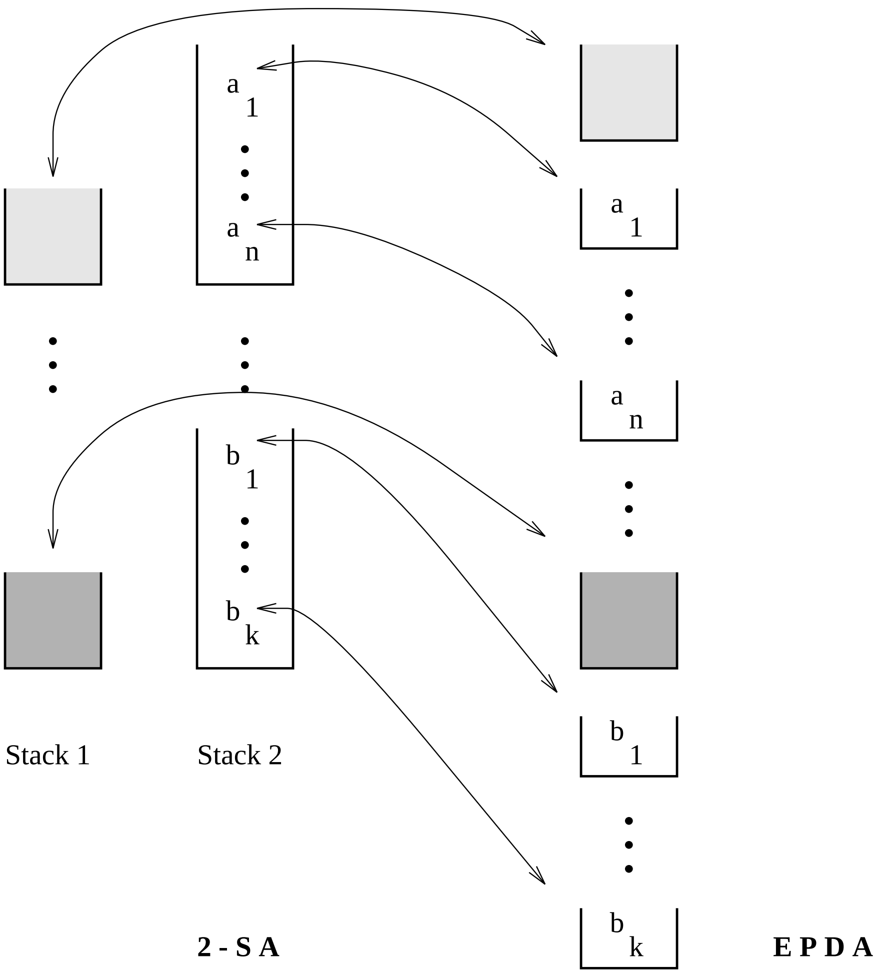

Figure 1.13: Simulation of a 2-SA by an EPDA.

of Stack 1 is deleted and the EPDA proceeds to simulate pop operations from Stack 2). Also, for all $q \in \Gamma$, transitions of the form $\delta([q : S2 : s_2], \epsilon, [\sqsubset, s']) \ni < [q : S1 : s'], \epsilon, \epsilon, \epsilon >$ are added (i.e., a topmost stack of the EPDA which simulates an empty topmost stack of Stack 2 is deleted an the EPDA proceeds to simulate operations on both stacks; also the top of the lower Stack 2 is retrieved).

If $s_1 = \sqsubset$, the simulated transition will be of the form

$\delta^E([q : S2 : s_2], a, s_2) \ni < [q' : S2 : a_n], \ldots \ddagger a_n >.$

By induction, it can be shown that M_E accepts a string $w \in \Sigma^*$ by empty stack iff M_2 accepts it.

$\square$

EPDA-L $\subseteq$ 2-SA-L

Note that there is a normal form for EPDAs such that every stack added in a move has only one stack element [VS87].

Figure 1.14 shows the structure of the simulation of an EPDA by an 2-SA.

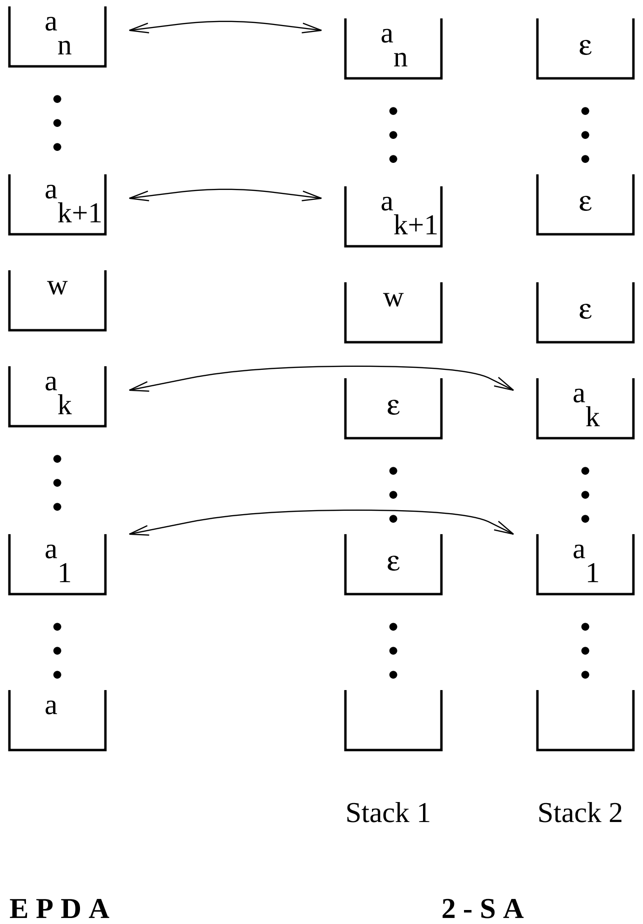

Figure 1.14: Simulation of an EPDA by an 2-SA. ϵ in a stack denotes an empty stack.

Theorem 1.2.3 *EPDA-L $\subseteq$ 2-SA-L*

Proof: (Sketch)

Given an EPDA $M_E = (Q, q_0, \Sigma, \Gamma, \gamma_0, \delta)$, each move of M_E can be simulated by a 2-SA $M_2 = (Q, q_0, \Sigma, \Gamma, \sqsubset, \gamma_0, \delta^2)$. The starting configuration $< q_0, \sqsubset \gamma_0, w, \epsilon >$ of M_E is simulated by the starting configuration $< q_0, \sqsubset \gamma_0, \sqsubset \gamma_0, w, \epsilon >$ of M_2.

A transition of M_E of the form

$\delta(q,a,s) \ni < q', \ddagger w_1 \ldots \ddagger w_k, w, \ddagger w_{k+1} \ldots \ddagger w_n >$ where $w, w_i \in \Gamma^*$, $a \in \Sigma \cup \{\epsilon\}$, $s \in \Gamma$ and $q, q' \in Q$

is simulated by a transition of M_2 of the form

$\delta^2(q,a,s,s') \ni < q', \sqsubset \ldots \sqsubset w \sqsubset w_{k+1} \ldots \sqsubset w_n, \sqsubset w_1 \ldots \sqsubset w_k \sqsubset \ldots \sqsubset >$

for all $s' \in \Gamma$ and where there are k empty stacks $\sqsubset \ldots \sqsubset$ below w on Stack 1 and n-k+1 empty stacks above $\sqsubset w_k$ on Stack 2.[7] This "padding" is necessary since empty stacks are always deleted simultaneously on Stack 1 and Stack 2. It also ensures that the ordering properly corresponds to the EPDA.

Because of the normal form, the w_i all consist of exactly one stack element so that no reordering (reversal) is necessary. In addition, the simulating 2-SA M_2 moves elements from Stack 2 to Stack 1 whenever Stack 1 becomes empty, i.e., $\delta^2(q, \epsilon, \sqsubset, s_2) \ni < q, \sqsubset s_2, \epsilon >$ for all $s_2 \in \Gamma$, $q \in Q$. . (Note that due to the normal form, in this manner complete stacks w_i are copied from Stack 2 to Stack 1.) Then the simulation proceeds as above.

By a straightforward induction proof, it can be shown that M_2 accepts a string $w \in \Sigma^*$ by empty stack iff M_E accepts it.

□

As a corollary of theorem 1.2.2 and 1.2.3 and the fact that EPDA-L=TAL, we have now proven theorem 1.2.1:

Corollary 1.2.1 *2-SA-L = TAL*

TAL $\subseteq$ 2-SA-L directly

A 2-SA can also simulate a TAG derivation directly in a top-down, left-to-right fashion just as a PDA can simulate a CFG derivation. Adjoining into a node is simulated (provided that the node is on top of Stack 1) by substituting that node with an encoding of the adjoined tree. An important notion is that of the spine: the path from the root node to the foot node of an auxiliary tree. All subtrees (i.e., further adjoinings and terminals) to the left of the spine of an adjoined tree are simulated immediately, all subtrees to the right of the spine are pushed onto Stack 2 for later simulation.

Given an auxiliary tree as in Figure 1.15, where "←" and "→" and "|" are three new symbols that represent "left subtrees are following", "right subtrees are following"

[7] Note that in the definition in section 1.2.2 only one pair of new stacks was allowed. This can easily be simulated by a number of intermediate transitions.

and "continue with spine" resp. and the $ti_{\{l,r\}j}$ are single symbols, representing the subtrees[8].

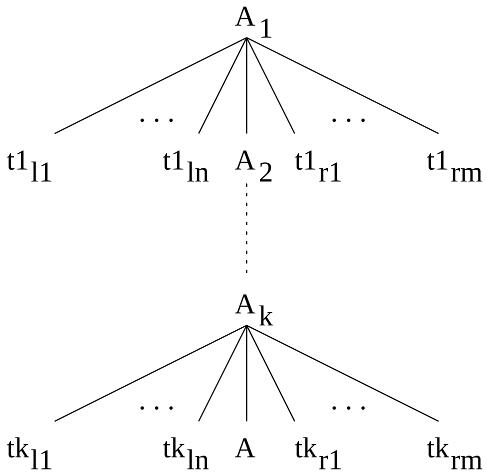

Figure 1.15: General form of an auxiliary tree.

In figure 1.15 ti_{lj} and ti_{rj} are the i-th subtrees on the j-th level to the (l)eft or (r)ight, then the corresponding string (that is pushed on Stack 1) is (top on the left):

A1 $\leftarrow$ t1l1 ... t1ln $\rightarrow$ t1rm ... t1r1 | A2 ... Ak $\leftarrow$ tkl1 ... tkln $\rightarrow$ tkrm ... tkr1 | A

All subtrees that are processed after a $\leftarrow$ sign "spawn a new simulation", i.e., new stacks are created on which the adjunctions and terminals of this subtree are simulated (and read from the input tape). After this is completed, i.e., the stacks are empty again, these stacks are deleted.

All subtrees that are processed after a "$\rightarrow$" sign are pushed onto Stack 2 for later processing. Only after all material "to the left of the spine" is processed, the subtrees "on the right" need be considered. At this point, however, Stack 1 is empty and the first subtree can be popped from Stack 2 to be processed on Stack 1.

The Hierarchy of n-SA

For general n-SA, there are several ways to restrict the pop operations from the n-1 additional stacks. Here, we will introduce two such restrictions that result in different hierarchies of accepted languages:

[8]Note that the indices for the subtrees would need to be even more complex to be exact: The rightmost subtree to the left of the spine on level k, now written as tk_{ln} should be tk_{ln_k} where n_k is the number of subtrees to the left of the spine on level k.

(i) An item may only be popped from Stack k if the topmost stacks of Stack 1 to Stack k-1 are empty. Stacks k+1 to Stack n may have items on their topmost stacks.

(ii) In addition to (i), the "additional stacks" (Stack 2 to Stack n) are "simultaneous one-turn" stacks, i.e., once Stack 1 is empty and thus an item from the first of the additional stacks is popped, nothing may be pushed onto any of the other additional stacks (Stack 2 to Stack n). This could be ensured by a flag (which can be realized/simulated in the finite control).

With either restriction, the n-SA generate an infinite hierarchy. Especially, for n>1, there is an n-SA with restriction (i) that can "count" up to 2^n, i.e., it recognizes the language $\mathrm{L}_{2^n} := \{a_1^m \ldots a_{2^n}^m \mid m > 0\}$. Note that this is exactly the property of EPDA, as discussed in section 1.2.1.

Also, for n>1, there is an n-SA with restriction (ii) that can "count" to 2n, i.e., it recognizes the language $\mathrm{L}_{2n} := \{a_1^m \ldots a_{2n}^m \mid m > 0\}$.

For n=3, the corresponding n-SA are given as an example:

Example 1: Let $\Sigma = \{a_1, \ldots, a_8\}$, $Q = \{q_0, \ldots, q_8\}$, $\Gamma = \Sigma$. M_i with the following transitions is a 3-SA with restriction (i) that counts to 8:

$$
\begin{array}{lcl}
\delta(q_0, a_1, \sigma_0, \epsilon, \epsilon) & = & \{< q_1, a_2, a_4, a_8 >\} \\
\delta(q_1, a_1, a_2, a_4, a_8) & = & \{< q_1, a_2a_2, a_4a_4, a_8a_8 >\}
\end{array}
$$

i.e. another a_2, a_4 and a_8 are pushed on Stacks 1,2 and 3 resp.

$$
\begin{array}{lcl}
\delta(q_1, a_2, a_2, a_4, a_8) & = & \{< q_2, \epsilon, a_3a_4, a_7a_8 >\} \\
\delta(q_2, a_2, a_2, a_3, a_7) & = & \{< q_2, \epsilon, a_3a_3, a_7a_7 >\} \\
\delta(q_2, a_3, \epsilon, a_3, a_7) & = & \{< q_3, \epsilon, \epsilon, a_6a_7 >\} \\
\delta(q_3, a_3, \epsilon, a_3, a_6) & = & \{< q_3, \epsilon, \epsilon, a_6a_6 >\} \\
\delta(q_3, a_4, \epsilon, a_4, a_6) & = & \{< q_4, \epsilon, \epsilon, a_5a_6 >\} \\
\delta(q_4, a_4, \epsilon, a_4, a_5) & = & \{< q_4, \epsilon, \epsilon, a_5a_5 >\} \\
\delta(q_4, a_5, \epsilon, \epsilon, a_5) & = & \{< q_5, \epsilon, \epsilon, \epsilon >\} \\
\delta(q_i, a_i, \epsilon, \epsilon, a_i) & = & \{< q_i, \epsilon, \epsilon, \epsilon >\} \ i = 5, 6, 7, 8 \\
\delta(q_i, a_{i+1}, \epsilon, \epsilon, a_{i+1}) & = & \{< q_{i+1}, \epsilon, \epsilon, \epsilon >\} \ i = 5, 6, 7
\end{array}
$$

All other transitions lead a non accepting "error-state."

Figure 1.16 shows the stacks of M_i with all their contents during a run of the 3-SA.

Example 2: M_{ii} is a 3-SA with restriction (ii) that counts to 6. Let $\Sigma = \{a_1, \ldots, a_6\}$, $q = \{q_0, \ldots, q_6\}$ and $\Gamma = \Sigma$. M_{ii} is defined by the following transitions:

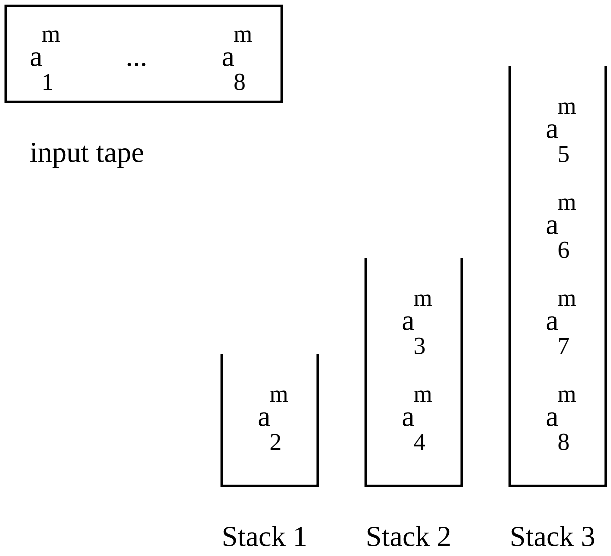

Figure 1.16: Stacks of M_i while counting to 8.

$$\begin{aligned}
\delta(q_0, a_1, \sigma_0, \epsilon, \epsilon) &= \{< q_1, a_2, a_4, a_6 >\}\\
\delta(q_1, a_1, a_2, a_4, a_6) &= \{< q_1, a_2a_2, a_4a_4, a_6a_6 >\}\\
\delta(q_1, a_2, a_2, a_4, a_6) &= \{< q_2, \epsilon, a_3a_4, a_5a_6 >\}\\
\delta(q_2, a_2, a_2, a_3, a_5) &= \{< q_2, \epsilon, a_3a_3, a_5a_5 >\}\\
\delta(q_2, a_3, \epsilon, a_3, a_5) &= \{< q_3, \epsilon, \epsilon, a_5 >\}
\end{aligned}$$

At this point, the additional stacks "turn simultaneously".
a_3 from Stack 2 is popped, a_5 on Stack 3 remains unchanged.

$$\begin{aligned}
\delta(q_3, a_3, \epsilon, a_3, a_5) &= \{< q_3, \epsilon, \epsilon, a_5 >\}\\
\delta(q_3, a_4, \epsilon, a_4, a_5) &= \{< q_4, \epsilon, \epsilon, a_5 >\}\\
\delta(q_4, a_4, \epsilon, a_4, a_5) &= \{< q_4, \epsilon, \epsilon, a_5 >\}\\
\delta(q_4, a_5, \epsilon, \epsilon, a_5) &= \{< q_5, \epsilon, \epsilon, \epsilon >\}\\
\delta(q_5, a_5, \epsilon, \epsilon, a_5) &= \{< q_5, \epsilon, \epsilon, \epsilon >\}\\
\delta(q_5, a_6, \epsilon, \epsilon, a_6) &= \{< q_6, \epsilon, \epsilon, \epsilon >\}\\
\delta(q_6, a_6, \epsilon, \epsilon, a_6) &= \{< q_6, \epsilon, \epsilon, \epsilon >\}
\end{aligned}$$

Again, all other transitions lead a non accepting "error-state."

Figure 1.17 again shows the stacks of M_{ii} with all their contents during a run of M_{ii}.

As the ability to count grows geometrically in case (i), we conjecture that these automata correspond to the NPDA hierarchy (generalized EPDA) in [Wei88]. The n-SA with restriction (ii), however, which grow linear in their ability to count, we conjecture to be indentical to local MC-TAGs [Wei88] in the sense that 2n-SA-L = tree-local-MC-TAL-n (the languages derived by local MC-TAGs with at most n elements in each tree sequence).

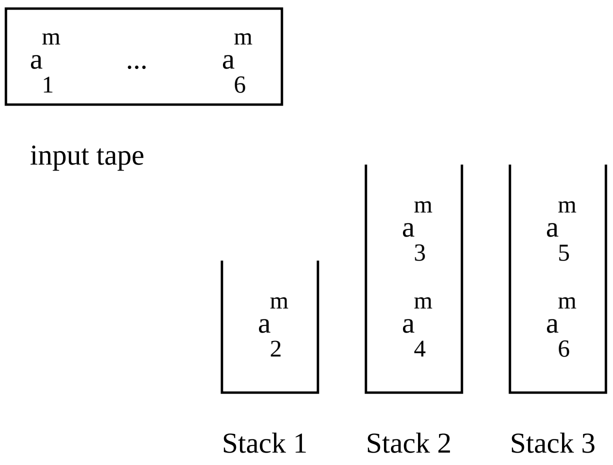

Figure 1.17: Stacks of M_{ii} while counting to 6.

1.2.3 Conclusion and Further Work

We have introduced n-SA, a new class of automata models as an extension of the classic PDA. The 2-SA were shown to be equivalent to EPDA and thus to TAG. The generalized n-SA are based on only one simple storage structure, namely the simple embedded stacks. Various constraints on the pop operation from the additional stacks ensure the restricted accepting capability of n-SA.

The proof for the equivalence of n-SA-L and local MC-TAL needs to be worked out. It will be interesting to investigate other formulations of the constraints on popping from the additional stacks. Less restricted versions of these constraints will lead beyond the class of LCFRL. E.g., using the additional stacks as an unordered multi-set (instead, as we have here, of an ordered list) leads to an automata model that accepts the scrambling language, which is known to be beyond LCFRS (see section 3.2).

As 2-SA so closely correspond to EPDA, the results of [Jos89] which accounts for psycholinguistic results in comparing crossed with nested dependencies by a complexity measure in an EPDA model still hold if the parsing model used is an 2-SA. The simpler construction of n-SA should allow for an easier extension of these investigations towards syntactic constructions which are more complex than TALs.

1.3 Multi-Component TAG (MC-TAG)

This section introduces Multi-Component TAGs (MC-TAG), a variant of TAG that has first been defined in [JLT75]. The version presented here has been introduced by [Wei88] and the extensions I define in section 4.2 will build on Weir's definition. The next section reviews a more general class of rewriting systems which includes both TAGs and MC-TAGs, the Linear Context-Free Rewriting Systems (LCFRS) which originally have been defined in [VSWJ87].

LCFRS are used in the next chapter, where I will show that a suitable formalization of the long-distance scrambling phenomenon as characterized in section 2.4.1 is beyond even the class of LCFRS and therefore also beyond TAGs.

The new grammar formalism MC-TAG-DL which will be introduced in section 4.2 below is build on one of the variants of Multi-Component TAGs (MC-TAG). In this section the three different variants of Multi-Component TAGs are introduced, closely based on the definitions in [Wei88]. The definitions are not given in full formal detail, the definitions relevant for the MC-TAG-DL formalism are given in section 4.2 below.

There are three versions of MC-TAGs that differ strongly in their generative capacity. The fundamental difference between TAGs and MC-TAGs (any of the three variants) is that instead of elementary trees, MC-TAGs have elementary tree sequences. A tree sequence is a finite list or multi-set[9] of *elementary trees* (as defined for TAGs). The difference between the variants of MC-TAGs is based in different definitions for the adjunction of such tree-sequences:

- **Tree-local MC-TAG**, sometimes called *benign* MC-TAG. In an adjunction operation, all trees of an elementary or derived tree sequence must adjoin into the same tree of another elementary tree sequence. The set of tree-local MC-TAG languages is weakly and strongly equivalent to TAL.

- **Set-local MC-TAG:** In an adjunction operation, all trees of an elementary or derived tree sequence must adjoin into the trees of another elementary tree sequence. The set of languages generated by a set-local MC-TAG (usually abbreviated as local MC-TAG) is a superset of TAL. There are two orthogonal hierarchies, the generative capacity depends on the number of trees in the tree sequences (see [Wei88]) as well as on the number of nodes in the tree sequences (see [RS92]) into which adjoining can take place. The union of all local-MC-TALs is exactly the class of LCFRS languages which is introduced in the next section.

- **Non-local MC-TAG:** In an adjunction operation, all trees of an elementary or derived tree sequence must adjoin into the trees of another elementary or derived tree sequence. This last part of the definition is crucial. It allows the trees of an elementary tree sequence to adjoin into unrelated positions in some

[9] Usually, tree sequences are denoted like sets. But as trees may appear more than once in a tree sequence, the correct formalization is in fact a (finite) sequence or multi-set.

derived tree sequence. Therefore, this variant is called *non-local*. The set of languages generated by non-local MC-TAGs is conjectured to be orthogonal to the class of LCFRS languages. There even exist non-local MC-TALs which are not semi-linear languages.

1.4 Linear Context-Free Rewriting Systems (LCFRS)

Linear Context-Free Rewriting Systems (or LCFRS) in their current form have been introduced by [VSWJ87]. They are a generalization of Context-Free Grammars and the definition we present here is actually based on a formalism called Generalized Context-Free Grammars (GCFG), first introduced by [Pol84]. Unlike other similar systems such as Multiple Context-Free Grammars (MCFG, see [SMFK91]) which are only string deriving systems[10], LCFRS are primarily systems that derive an arbitrary kind of structure (e.g. trees) which can be more complex than strings. Just as in TAGs, a derived string is computed from the derived structure only in a second step.

The definition of LCFRS presented here relies strongly on [Wei88]. An LCFRS consists of two parts. The first part of an LCFRS is basically a context-free grammar. It derives some *terms* that are then interpreted by a set of functions. This interpretation is the second part of an LCFRS. The terms are constructed in the same way as in the derivation process of a Context-Free Grammar. Starting from some distinguished *start symbol*, rewriting productions are applied that substitute one *nonterminal symbol* by terms of the form $f(A_1, \ldots, A_n)$ where $A_1, \ldots, A_n$ are terminal or nonterminal symbols. In a Context-Free Grammar, a nonterminal symbol is instead simply replaced by a (concatenated) sequence of terminals or nonterminals $A_1, \ldots, A_n$.

Example 1: CFG as an LCFRS

As a trivial example, Context-Free Grammars can be seen as an LCFRS. Given a CFG, the first part of the corresponding LCFRS is constructed by substituting the context-free rules such as $A \to A_1 \ldots A_n$ by corresponding term deriving rules as $A \to f(A_1, \ldots, A_n)$ with the same function symbol f for all productions. The second part of the corresponding LCFRS is the interpretation function. There is only one function symbol f which is interpreted as the concatenation function.

Definition 1.4.1 *A* **Generalized Context-Free Grammar** *(GCFG) G is a five-tuple (N,T,S,F,P), where*

[10] MCFG is a notational variant of the string-rewriting version of LCFRS; they are basically the same formalism.

- *N is a nonempty, finite set of* nonterminals*,*
- *T is a nonempty, finite set of* terminals *with $N \cap T = \emptyset$,*
- *S the* start symbol *is a distinguished element of N,*
- *F is a finite set of* function symbols *and*
- *P is a finite set of* productions *of the form $A \to f(A_1, \ldots, A_n)$, where $n \geq 0, f \in F$, and $A, A_1, \ldots, A_n \in N \cup T$.*

The **derives relation** $\Rightarrow$ *is defined recursively as*

- $A \Rightarrow f(A_1, \ldots, A_n)$ *if* $A \to f(A_1, \ldots, A_n)$ *is in* P.
- $A \Rightarrow f(t_1, \ldots, t_n)$ *if* $A \to f(A_1, \ldots, A_n)$ *is in* P *and* $A_i \Rightarrow t_i$ *for* $1 \leq i \leq n$.

T(G), the set of terms derived by G is defined as

$T(G) \ := \ \{ \ t \mid S \Rightarrow t \ \}$.

Note that the application of a production is completely independent of the context in which some nonterminal appears, thus justifying the term 'context-free' in GCFG.

In an LCFRS, which consists of a GCFG and an interpretation I, each term t of the GCFG represents a derivation tree (term) of some complex structure which can be constructed by the interpretation. This interpretation is a mapping from the function symbols to some composition operations. An interpretation of a term t will be written as $[[t]]$.

In a first step, we can now define Context-Free Rewriting Systems, the restriction to *linear* systems is defined later.

Definition 1.4.2 *A* **Context-Free Rewriting System** *(CFRS) is a tuple (G,I) where G is a GCFG and I is an interpretation function, mapping the function symbols of G to (composition) operations over some structure and mapping the terminal symbols of G to such structures. The application of I to some term t is written as* $[[t]]$.

Example 1 (cont.): CFG as an LCFRS

The above example of Context-Free Grammars as an (L)CFRS can now be formalized as follows: The structure that CFGs are dealing with are simply strings, the interpretation maps the only function symbol f to the concatenation function, written as "$\circ$".

A rule of a CFG G, say $A \to A_1 \dots A_n$ is seen as a production $A \to f(A_1, \dots, A_n)$ of a GCFG. Terminal symbols are interpreted as themselves and the (only) function symbol f is interpreted as the concatenation function, so some derived term t of the GCFG, say $f(t_1, \dots, t_n)$ is interpreted as follows:

$$[[f(t_1, \dots, t_n)]] \ = \ [[t_1]] \circ \dots \circ [[t_n]]$$

Two more examples show how more complex grammars can also be interpreted as LCFRS.

Example 2: TAG as an LCFRS

TAGs derive trees instead of strings and they use adjunction (and substitution) as the tree manipulating operation. Thus the interpretation function I must reflect this by interpreting the function symbols as adjunctions. But first, a corresponding GCFG must be found. As already discussed in section 1.1.1, TAGs have a *derivation tree* and a *derived tree*. The derivation tree has nodes that are labeled with elementary trees. If this derivation tree is seen as the (context-free) derivation tree of a GCFG, then each combination of an elementary tree with some other elementary trees adjoined into it must be seen as one possible production. So the corresponding GCFG of a given TAG has a (somewhat large) set of productions P defined as follows:

$P \ = \ \{ \ \gamma \to f_\gamma(\gamma_1, \dots, \gamma_n) \mid \gamma_1, \dots, \gamma_n$ elementary trees which can be adjoined into the elementary tree γ $\}$

Note that it is now crucial that the elementary tree γ and the nodes at which adjunction actually takes place are encoded in the function symbol. The terminals of the GCFG are the elementary trees themselves. If we are only looking at adjunction, then it is even sufficient to have the auxiliary trees as terminals.

Now, the interpretation function I maps the function symbols to tree manipulating functions:

$$[[f_\gamma(t_1, \dots, t_n)]] \ = \ \gamma(a_1, [[t_1]]) \dots (a_n, [[t_n]])$$

where $\gamma(a_1, [[t_1]]) \dots (a_n, [[t_n]])$ is the elementary tree γ in which the trees $[[t_1]], \dots, [[t_n]]$ have been adjoined at the addresses $a_1, \dots, a_n$. The interpretation of the elementary trees (which are the terminals of the GCFG) are just the trees themselves. So some derived term of the GCFG can now be interpreted as a tree that is constructed by successive adjoinings into an initial tree.

In this way, TAGs are reinterpreted as an LCFRS. Note that it becomes very clear here that TAGs are a tree generating system. So far nothing has been said about strings. This is left for the next section.

Example 3: MC-TAG as an LCFRS

For (local) MC-TAGs (see section 1.3) the situation is very similar, the difference being that elementary tree sequences $< \gamma_1, \ldots, \gamma_k >$ instead of elementary trees are combined. The interpretation function is defined as follows:

$$[[f_{<\gamma_1,\ldots,\gamma_k>}(t_1, \ldots, t_n)]] \; = < \gamma_1, \ldots, \gamma_k > (a_{1,1}, \ldots, a_{1,k}, [[t_1]]) \ldots (a_{n,1}, \ldots, a_{n,k}, [[t_n]])$$

where $< \gamma_1, \ldots, \gamma_k > (a_{1,1}, \ldots, a_{1,k}, [[t_1]]), \ldots, (a_{n,1}, \ldots, a_{n,k}, [[t_n]])$ is the elementary tree sequence $< \gamma_1, \ldots, \gamma_k >$ in which at the addresses $a_{p,1}, \ldots, a_{p,k}$ the k trees of the tree sequence $[[t_p]]$ have been adjoined (in that order). Note that the addresses $a_{p,q}$ can be addresses in different trees of the tree sequence $< \gamma_1, \ldots, \gamma_k >$.

The corresponding GCFG and the interpretation function can be constructed just as above. Also, MC-TAGs are tree generating systems, the interpretation of the string generation process is discussed in the next section.

Example 4: Coupled-CFG as an LCFRS

Another formalism of interest which can be interpreted as an LCFRS are Coupled Context-Free Grammars (CCFG, see [Gua92] where they are called "Klammergrammatiken"). They constitute a hierarchy with the second level, i.e. CCFG(2), deriving exactly the Tree Adjoining Languages (see [Gua92]).

Definition 1.4.3 (Coupled Context-Free Grammar (CCFG))

*Let Γ be a finite, nonempty alphabet. A set $K = \{(f_1, \ldots, f_n) \mid n \geq 1, \; f_1, \ldots, f_n \in \Gamma\}$ is called a κ-*set *(a set of parentheses) if K is finite, and all f_i of all parentheses are disjunct. K[n] is defined as the set of n-ary parentheses in K.*

Let Σ be a finite alphabet, $K \cap \Sigma = \Gamma \cap \Sigma = \emptyset$. The extended Dyck-language of K with fill-ins from Σ, *abbreviated as ED(K,Σ), is recursively defined as:*

- $\Sigma^* \subseteq$ *ED(K,Σ)*
- *K[1] $\subseteq$ ED(K,Σ)*
- *If $u_1, \ldots, u_k \in$ ED(K,Σ) and $(f_1, \ldots, f_{k+1}) \in$ K[k+1] then* $f_1 u_1 \ldots f_k u_k f_{k+1} \in$ *ED(K,Σ)*
- *If $u, v \in$ ED(K,Σ), then $uv \in$ ED(K,Σ)*

A Parentheses Rewriting System, *or* Coupled Rewriting System for ED(K,Σ) *is a finite, nonempty subset P of $K \otimes$ ED(K,Σ) with productions of the form:*

$(f_1, \ldots, f_n) \rightarrow (z_1, \ldots, z_k)$ *with* $(f_1, \ldots, f_n) \in K$ *and* $z_1 \ldots z_k \in$ *ED(K,Σ)*

Finally, a CCFG for ED(K,Σ) is a four-tuple $G = (K,\Sigma,P,S)$ where P is a coupled rewriting system for ED(K,Σ) and S is a finite subset of starting strings from ED(K,Σ). CCFG(n) is defined as the set of CCFG grammars with at most n-ary parentheses in K.

Let $V = \Gamma \cup \Sigma$. The derives relation $\rightarrow$ is defined for $\alpha, \beta \in V^*$ as follows:

$\alpha \rightarrow \beta$ if there are $u_1, u_{k+1} \in V^*$, $u_2, \ldots, u_k \in ED(K,\Sigma)$ and $(f_1, \ldots, f_n) \rightarrow (z_1, \ldots, z_k) \in P$ such that

$\alpha = u_1 f_1 \ldots u_k f_k u_{k+1}, \quad \beta = u_1 z_1 \ldots u_k z_k u_{k+1}.$

Let $\xrightarrow{*}$ be the reflexive and transitive closure of $\rightarrow$. Then

$L(G) := \{w \in \Sigma^* \mid \exists s \in S \text{ with } s \xrightarrow{*} w\}$

is the language generated by G.

The following example is a CCFG(2) grammar G = (K,Σ,P,S) which derives the language $L_4 = \{a^n b^n c^n d^n \mid n \geq 0\}$:

$$
\begin{aligned}
K &= \{(s_1, s_2)\},\\
\Sigma &= \{a, b, c, d\},\\
S &= \{s_1 s_2\},\\
P &= \{(s_1, s_2) \rightarrow (a s_1 b, c s_2 d),\ (s_1, s_2) \rightarrow (\epsilon, \epsilon)\}.
\end{aligned}
$$

Now, a production $p = (f_1, \ldots, f_n) \rightarrow (z_1, \ldots, z_k)$ of a CCFG with $z := z_1 \ldots z_k$ and $A := (f_1, \ldots, f_n)$ can be reinterpreted as a production $A \rightarrow f_p(A_1, \ldots, A_n)$ if z contains exactly the parentheses $A_1, \ldots, A_n$. The interpretation $[[f_p(A_1, \ldots, A_n)]]$ for the function symbol f_p is provided by the production p of the CCFG which defines how strings which are contributed by the elements of the A_i are rearranged into an k-ary tuple of strings. E.g. if the parenthesis A_i includes some $f_j \in \Gamma$ then the production p of the CCFG G defines the location of $[[f_j]]$ in the interpretation $[[f_p(A_1, \ldots, A_n)]]$.

In the example grammar G above, the first production of G is reinterpreted as a production $A \rightarrow f_1(A_1)$ of an LCFRS with $A = A_1 := (s_1, s_2)$ and interpreted as $[[f_1(A_1)]] := (a[[s_1]]b, c[[s_2]]d)$.

In this way, CCFG can be reinterpreted as LCFRS. The inclusion of CCFG in LCFRS is proper, since there are some LCFRS languages which can not be derived by any CCFG(n) grammar.

An example is the language

$L = \{\ ww \mid \text{ where } w \text{ is a Dyck-word }\}.$

1.4.1 Restrictions on the Composition Operations

So far, no constraints on the composition operations have been made. In all of the above examples, however, some restrictions can be recognized. First, the structures that are derived, are all tuples of some kind. Also, the combination operations observe certain restrictions on the operations over these tuples: No element of a tuple is *deleted, copied* or *changed* (e.g. splitting into parts) in the composition step. If some of those constraints are dropped, very different systems can be obtained, including the simulation of a Turing Machine. On the other hand, with a formalization of these constraints, it can be shown that an interesting restricted class of structures (and especially strings, i.e. languages) can be derived.

With a strong emphasis on the string languages, we first need to define a *yield function* of a structure, which is then used to formalize the constraints on the composition operations. The usual yield (denoted as Φ) of a structure is a function that results in a set, sequence or tuple of strings that are associated with the structure. Usually the yield is just one string but in an MC-TAG the yield of a tree sequence can be seen as a sequence of strings which each represent the yield of a single tree of the tree sequence. In addition to this usual yield function, a new yield function (denoted as f^y) is defined that results in a tuple of strings and has specific properties.

As an example, look at how TAGs observe the constraints listed above. The composition operation adjunction does not *split up* the adjoined tree but the usual yield Φ of the adjoined tree (the string composed of the terminals at the leaves of the tree) is split up into two strings to the left and right of the footnode resp. This can be accommodated by defining the new yield function f^y of an auxiliary tree as a pair of strings, each string representing the usual yield of the tree to the left and to the right of the footnode resp. Now, in terms of the new yield function f^y the composition operation of TAGs then also observes the constraint of not *splitting up* the elements of the combined tuples.

For each composition operation of an LCFRS, the definition of such a yield function must be possible. The constraints (no copying, deletion or change) can now be stated in terms of the yield function.

Definition 1.4.4 *A* **Linear Context-Free Rewriting System** *(LCFRS) is a tuple (G,I,Y) where G is a GCFG, I is an interpretation function, mapping the function symbols of G to composition operations over some structure, Φ is the usual yield function for the structures, and Y is a function which maps the composition operations $I(f)$ over the structure to composition operations f^y over strings. For each $f \in F$ there is a corresponding yield function f^y (in [Wei88] denoted as $\overline{f}$) such that if*

$$\Phi([[t_1]]) = < w_{1,1}, \ldots, w_{1,k} >, \ldots, \Phi([[t_n]]) = < w_{n,1}, \ldots, w_{n,k} >$$

and

$\Phi([[f(t_1, \ldots, t_n)]]) \ = < w_1, \ldots, w_k >$

then

$f^y(< w_{1,1}, \ldots, w_{1,k} >, \ldots, < w_{n,1}, \ldots, w_{n,k} >) \ = < w_1, \ldots, w_k >$

Furthermore, each f^y *has to be defined by an equation of the form:*

$f^y(< x_{1,1}, \ldots, x_{1,k} >, \ldots, < x_{n,1}, \ldots, x_{n,k} >) \ = < t_1, \ldots, t_k >$

where $n \geq 0$*, each* t_i *is a string of variables (x's) and some finite number of terminal symbols. All variables appear exactly once on each side of the equation. (This ensures that there is no copying or deletion or change of the composed strings which are represented by the variables.)*

[VSWJ87] shows that LCFRS[11] are semi-linear, that they can be recognized in polynomial time[12] and are a proper subset of P, the set of languages which can be recognized in polynomial time. Furthermore, LCFRS are weakly equivalent to local-MC-TAGs and some other systems, such as MCFG (see [SMFK91]).

I have now defined all the grammatical systems which will be needed in the following two chapters. Thus, I can proceed to show in what sense the LDS phenomenon is beyond the capacity of TAGs and even the whole class of LCFRS. And in chapter 4 two extensions to the TAG framework are discussed which both can capture the LDS phenomenon.

[11] I use the term LCFRS, as here, also for the class of all languages that can be derived by some LCFRS. The intended meaning should be clear from the context.

[12] By an argument giving an Alternating Turing Machine which recognizes the string language of an LCFRS in logarithmic space.

Chapter 2

Natural Language Phenomena beyond TAG

In this chapter, I first review two arguments which have been made about the inadequacy of the TAG formalism to capture certain phenomena. The first argument by [Mil91] is about the strong generative capacity of TAGs and is based on extraction phenomena in Swedish and Norwegian. Miller shows that TAGs can not generate the phrase structures of a particular linguistic analysis. The second argument by [Rad91] is about the weak generative capacity of TAGs. It is based on number names in Chinese and Radzinski shows that TAGs (and in fact no formalism in the more general class LCFRS) can weakly generate the underlying formal language.

Then, I investigate the properties of so-called free word order languages. I present data from two such languages, from German and Korean. After a short introduction to the amount of word order freedom that these languages really have, I will focus on one specific phenomenon, namely scrambling. One variant of the scrambling phenomenon, called Long-Distance Scrambling (LDS), has very interesting properties which are characterized in more detail in this chapter (section 2.4). A formal characterization of LDS which is based on the introduction of the "derivational generative capacity" is given in the following chapter. The formal proof that the LDS phenomenon is beyond the (derivational generative) capacity of TAGs, even beyond that of the whole class of LCFRS, which is given in the final section of the next chapter depends crucially on this new notion of derivational generative capacity. These formal results are the basis for the new extensions to the TAG formalism which I introduce in chapter 4.

In the now following two sections, I will briefly discuss two other approaches from the literature to show that certain natural language phenomena are too complex to be described with regular TAGs. Both arguments are not as general as the argument I bring forward in chapter 3 based on LDS. The first argument, by [Mil91], is based on the type of phrase structure trees which is needed for a specific analysis of Scandinavian extraction phenomena. The second argument, by [Rad91], is based on number names in Chinese. Though the argument is more general as it is not based on a specific linguistic theory, the phenomenon in question is not clearly a syntactic one. Radzinski himself discusses this point in detail.

2.1 Scandinavian Extraction Phenomena

In [Mil91], Miller looks at Swedish and Norwegian extraction phenomena and shows (based on a particular linguistic analysis) that neither indexed grammars nor Tree Adjoining Grammars are adequate for characterizing these phenomena[1]. In the case of TAGs the argument is based on the strong generative capacity of TAGs. The phrase structures that the linguistic analysis calls for can not be generated by a TAG.

An example of the relevant Swedish sentences is given as sentence (1)[2]:

(1) Här är pojken som jag undrar vilken flicka$_2$ Kalle undrar vilka barn$_3$ han$_1$ trodde att hon$_2$ hadde rekommenderat ϵ_3 til studenterna.
Here is the boy$_1$ that I wonder which girl$_2$ Kalle wonders which children$_3$ he$_1$ thinks that she$_2$ had recommended ϵ_3 to the students.

In general, the phrase "vilken flicka$_2$ Kalle undrar" ("which girl Kalle wonders") and the depending phrase "att hon$_2$ trodde" ("that she thought") can be repeated any number of times. This can be formalized as a regular expression (2):

$$\text{(2) Här är} \left\{\begin{array}{c}\text{M}\\\text{F}\\\text{Pl}\end{array}\right\} \text{som jag undrar} \left[\left\{\begin{array}{c}\text{vilkenM}\\\text{vilken F}\\\text{vilka Pl}\end{array}\right\}\text{SG undrar}\right]^+$$

$$\text{vilken pojke} \left\{\begin{array}{c}\text{han}\\\text{hon}\\\text{de}\end{array}\right\} \text{trodde} \left[\text{att}\left\{\begin{array}{c}\text{han}\\\text{hon}\\\text{de}\end{array}\right\}\text{trodde}\right]^*$$

$$\left\{\begin{array}{c}\text{han}\\\text{hon } c\\\text{de}\end{array}\right\} \text{hade rekommenderat till studenterna.}$$

Here is the M/F/Pl that I wonder [which M/F/Pl SG wonders]$^+$ which boy

[1] The first part of the paper is concerned with showing that Swedish is not a context free language – an argument which is based on the weak generative capacity only.

[2] Sentence (1) is appears as (9)a in [Mil91].

he/she/they thought [that he/she/they thought]* he/she/they had recommended to the students.

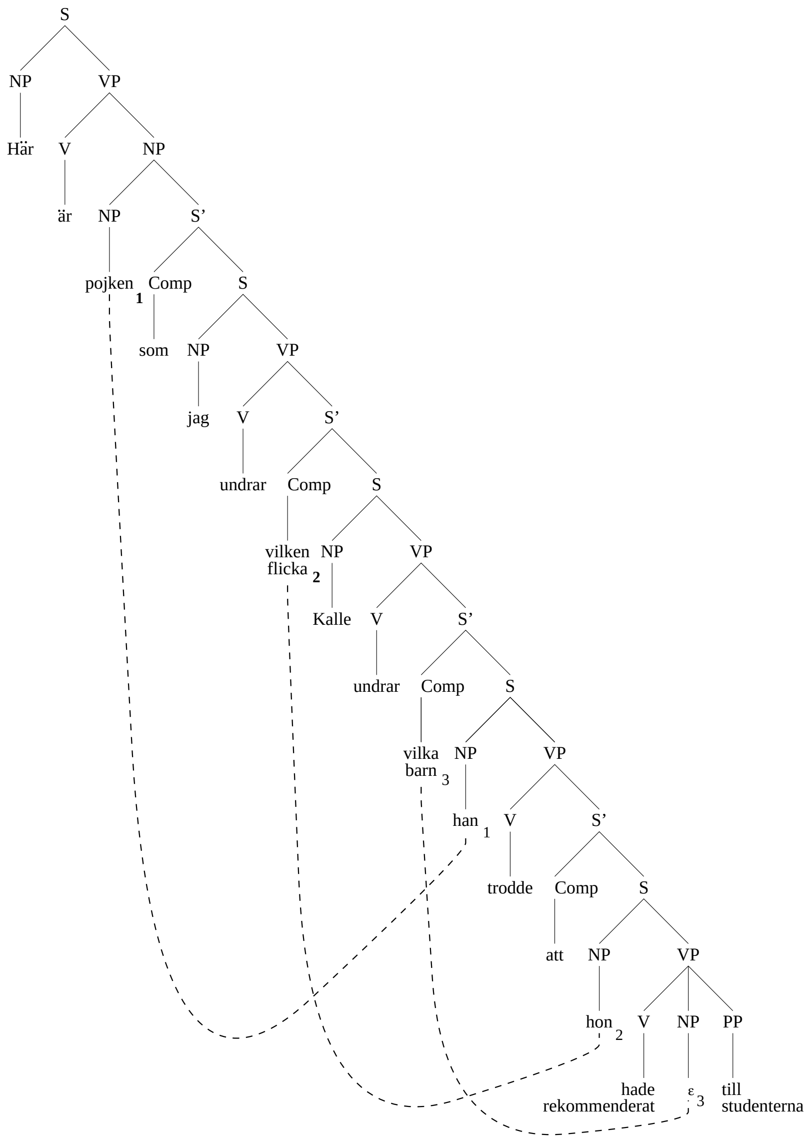

Figure 2.1: Right branching phrase structure of sentence (1). The dashed lines show the cross-serial dependencies.

Here M, F and Pl represent some masculine singular noun, feminine singular noun and plural noun respectively, SG represents some singular NP. The exact conditions on these sentences are not relevant here, it suffices to say that there are agreement constraints as the following: For each each occurence of e.g. a masculine singular noun M in "[vilken M/vilken F/vilka Pl SG undrar]" there must be a depending occurence of "han" in "[att han/hon/de trodde]". Therefore sentences with an unbounded set of cross-serially depending elements are part of Swedish.

According to Miller and [MZ82] the correct analysis of the extraction phenomenon in a phrase structure based account is a right branching structure as in figure 2.1.

Even without a constraint as the co-occurrence constraint (see section 3.2 below), it is obvious that the dependent elements must appear within the same elementary tree in order to meet the agreement constraints. Clearly, no TAG can generate unbounded right branching structures of this kind which exhibit the kind of cross-serial dependencies involved here.

Even though these cross-serial dependencies can be mapped into a copying structure and TAGs can indeed derive the copy language

$\mathrm{L} = \{\mathrm{ww} \mid \mathrm{w} \in \Sigma^*\}$

the resulting phrase structures are not right branching (see e.g. [Jos85]).

This is an argument about the inadequacy of TAGs which can only be based on the strong generative capacity (i.e. the phrase structures) of the formalism (and is thus relying on a particular linguistic analysis). In fact, on the level of the weak generative capacity (i.e. formal languages) TAGs are sufficient to describe this extraction phenomenon. Also, a very restricted version of set-local MC-TAGs (with no more than two trees in each tree sequence) can generate the copy language with right branching phrase structures.

2.2 Chinese Number-Names

The following argument of [Rad91] is based on the weak generative capacity of LCFRS and thus independent from a particular linguistic analysis. However, it is not clear whether the phenomenon discussed is a purely syntactic one.

Number-names in Chinese are constructed on the base of the names for 10^4 ("wan"), 10^8 ("wan wan" or "yi"), 10^{12} ("wan wan wan" or "wan yi" or "zhao") etc. Some native speakers find the longer number-names akward since shorter ones exists. But "zaho" (for 10^{12}) is the largest single-word name, though some archaic single-word number-names up to 10^{44} exist. So for really large numbers, sequences of "zhao" are needed.

Number-names in Chinese are (just as in English) constructed 'from left to right,' so examples (a) and (b) are well-formed number-names[3] and example (c) is not:

[3] Note that there is an error in the number given for (a) in [Rad91].

(a) wu zhao zhao wu zhao
five trillion trillion five trillion
5,000,000,000,005,000,000,000,000

(b) wu zhao zhao zhao zhao wu zhao zhao
five trillion trillion trillion trillion five trillion trillion
5,000,000,000,000,000,000,000,005,000,000,000,000,000,000,000,000

(c) *wu zhao zhao wu zhao zhao zhao
five trillion trillion five trillion trillion trillion
intended meaning: 5,000,000,000,005,000,000,000,000,000,000,000,000

The underlying pattern in the number-names is that longer sequences of "zhao" must be followed by ever smaller sequences of "zhao." A similar argument has already be made for English (based on sequences of "thousand" instead of "billion", "trillion", etc.) in [Zwi63]. This construction can roughly be characterized as follows:

(i) The term "thousand" is used as the largest single-word number-name, i.e. "billion" (Amer.) is expressed as "thousand thousand thousand."

(ii) Long sequences of "thousand" must always precede shorter sequences (and no sequence of "thousand" may be repeated).

Radzinski discusses various arguments against the validity of this construction in English, but clearly the Chinese data are of this type.

As a formal language for some of the Chinese number-names, we get (by ignoring all other number-names between 1 and 9,999 but "wu"):

J := $\{$ wu zhao^{k_1} ... wu zhao^{k_n} $\mid k_1 < \ldots < k_n < 0 \}$

Radzinski proves that J is not a Tree Adjoining language by contradiction. As we know from [VS87], the intersection of J with a regular language and the application of a homomorphism yields a language which is still a Tree Adjoining language if J is. Radzinski gives such a regular language and a homomorphism with the resulting language L.

L = $\{ ab^n ab^m ab^l ab^k ab^j \mid n > m > l > k > j > 0 \}$

Using a pumping lemma for TAGs (from [VS87]), he can then show that L is not in TAL and thus J is not in TAL. By using the intersection with a regular language, the argument can then easily be extended to the language NC of all Number-names in Chinese.

The argument is then extended to the whole class of local Multi-Component TAGs (see section 1.3) by using an appropriate pumping lemma. It follows that the Chinese number-names are beyond the weak capacity of Linear Context-Free Rewriting Systems (LCFRS, see section 1.4).

Still, it is not clear, if the constraint expressed in part (ii) of the characterization above is really a syntactic one. Radzinski concludes that even if it is not purely syntactic, it still relates to "the interface between the natural language Chinese and the mathematical component of the human cognitive endowment."

The Scandinavian extraction phenomenon argument of section 2.1 and the argument in this section both suffer from strong limitations. The first argument depends on a particular linguistic analysis while the second is not clearly a syntactic argument. A much clearer result is shown for a purely syntactic phenomenon in chapter 3, namely that the LDS phenomenon is beyond the capacity of LCFRS.

2.3 Free Word Order Languages

Many natural languages exhibit a property that is called free word order. In no case does this actually mean that the words of one sentence can come in any order, some constraints always apply. Even in the language most commonly cited as an example of free-word order, namely the Australian Aboriginal language Warlpiri, some non-syntactical constraints regarding focus apply (see [Kas87]). In other languages such as German only some of the words may appear in any order, other words have fixed positions and as in Warlpiri, topic/focus information plays a key role in determining (and further constraining) the word order.

As an example consider the following sentence[4]:

(1)	Der Lehrer	gab	gestern	den Kindern	die Bücher.
	The teacher (NOM)	gave	yesterday	the children (DAT)	the books (ACC)
	"Yesterday, the teacher gave the children the books."				

Here, the order is free only between the noun-phrases "der Lehrer", "den Kindern", "die Bücher" and the adverbial phrase "gestern" while the finite verb "gab" always appears at the second position. The word order is also fixed within one noun-phrase,

[4]The terms NOM, DAT and ACC in parentheses after nouns mark the case of the noun, i.e. nominative, dative and accusative case resp.

e.g. "Lehrer der" is unacceptable. So instead of free word-order, we should in fact call the phenomenon free phrase-order.

But for the arguments of the verb, all $4! = 24$ possible orderings are correct sentences. See (2) for some examples.

(2) Gestern gab der Lehrer die Bücher den Kindern.
Den Kindern gab die Bücher der Lehrer gestern.
Die Bücher gab gestern den Kindern der Lehrer.
Der Lehrer gab den Kindern gestern die Bücher.

This unconstraint word-order freedom of arguments of the verb is only the general rule, things change, e.g. when pronouns are used. Consider the sentence

(3) Ihr	gab	es	er.
her (DAT)	gave	it (ACC)	he (NOM)

"He gave it to her."

This sentence is only acceptable with a strong emphasis on "ihr" and on "er".

The free ordering of arguments within one clause is also called clause-internal scrambling. In the next section, I will show that arguments of a verb can also move (scramble) to positions outside the range of the originating clause.

2.4 Long-Distance Scrambling (LDS)

In this section, I will introduce the long-distance scrambling phenomenon with examples from German and Korean and discuss the range of this phenomenon. I will characterize long-distance scrambling (LDS) as an in principal unbounded phenomenon. This unboundedness will be shown to have important formal properties, which are discussed in chapter 3.

2.4.1 German Data

In German (and in many other SOV languages[5], such as Korean, Hindi and Japanese), a constituent of an embedded clause may be moved from that clause into the matrix clause.

[5]SOV stands for 'subject–object–verb' and describes the basic order of these constituents in a main clause.

Consider, for example, the following sentences[6]:

(1)	...daß	[ich	dem Kunden	[PRO den Kühlschrank	zu reparieren]
	...that	I	the client (dat)	the refrigerator (acc)	to repair

versprochen	habe]
promised	have

"...that I have promised the client to repair the refrigerator."

In German, the object of the embedded clause can be "moved"[7] to any position in the matrix clause, e.g.:

(2)	...daß	ich$_i$	[den Kühlschrank]$_j$	dem Kunden
	...that	I	the refrigerator (acc)	the client (dat)

[PRO$_i$ t$_j$ zu reparieren]	versprochen	habe
to repair	promised	have

(3)	...daß	[den Kühlschrank]$_j$	[ich$_i$	dem Kunden
	...that	the refrigerator (acc)	I	the client (dat)

[PRO$_i$ t$_j$ zu reparieren]	versprochen	habe]
to repair	promised	have

"...that I have promised the client to repair the refrigerator."

A striking feature of scrambling is its freedom. To illustrate this freedom we will present two additional examples in which scrambling of a more complex nature occurs.

First, more than one constituent from an immediately embedded clause may move into the matrix clause. In the matrix clause, the scrambled embedded constituents may occupy any position. An example of multiple movement out of one embedded clause is given in sentences (2a) and (2b) below.

(2a)	...daß	der Detektiv$_i$	dem Klienten	[PRO$_i$	den Verdächtigen
	...that	the detective (nom)	the client (dat)		the suspect (acc)
	des Verbrechens	zu überführen]	versprochen	hat.	
	the crime (gen)	to indict	promised	has	

(2b)	...daß	[des Verbrechens]$_k$	der Detektiv$_i$	[den Verdächtigen]$_j$
	...that	the crime (gen)	the detective (nom)	the suspect (acc)

[6]Imagine the example sentences to be embedded in yet another sentence, e.g. "Er sagte, daß..." ("He said that ..."). This embedding has the useful effect that the "matrix" verbs of the example sentences are in final position.

[7]We use "traces" (i.e. terms of the form "t$_i$" which stand for the empty string) only to indicate the standard (or 'unmarked') order of the phrases in the clause.

dem Klienten	[PRO$_i$ t_j t_k	zu überführen]	versprochen	hat.
the client (dat)		to indict	promised	has

"...that the detective has promised the client to indict the suspect of the crime."

Second, constituents may be moved out of more than one nested embedded clause. An example can be found in sentences (3a) and (3b).

(3a)

...daß	der Rat	dem Pfarrer	[die Witwen$_i$	[PRO$_i$
...that	the council (NOM)	the priest (DAT)	the widows (ACC)	

der Opfer	gedenken]	zu lassen]	versprochen	hat.
the victims (GEN)	commemorate	to let	promised	have

(3b)

...daß	[die Witwen]$_j$	[der Opfer]$_i$	[dem Pfarrer]$_k$
...that	the widows (ACC)	the victims (GEN)	the priest (DAT)

der Rat t_k [t_j [PRO$_j$ t_i	gedenken]	zu lassen]	versprochen	hat.
the council (NOM)	to commemorate	let	promised	have

"...that the council has promised the priest to let the widows commemorate the victims."

(2b) and (3b) are judged to be quite marginal by some native speakers, and are fairly difficult to process, but they are not ungrammatical. The processing difficulties increase as the number of embeddings, the number of scrambled NPs, and/or the distance over which the NPs are scrambled increase. Processing difficulties make grammaticality judgments difficult and unreliable. However, there is clear linguistic evidence that the number of scrambled NPs is not limited to one (as in the case of wh-movement) and that there is not only one position in the matrix clause(s) into which NPs can scramble (again, unlike in the case of wh-movement). Furthermore, it is impossible to determine a plausible grammaticality boundary for the number of NPs that may be scrambled, or on the distance of their movement. We thus conclude that scrambling is syntactically unbounded in the following two senses.

Conclusion 2.4.1 Long-Distance Scrambling is doubly unbounded *in the following two senses:*

1. *There are no syntactic restrictions on the number of verbal arguments that move.*
2. *There are no syntactic restrictions on the distances over which verbal arguments may move.*

Henceforth, we will assume without further comment that any ordering of the arguments from all clauses is possible. It is this range of constructions that we will consider the syntax of long-distance scrambling in German, and that we will formally investigate in chapter 3. Additional proof for this characterization comes from the Korean language; several examples are given in the following section.

2.4.2 Korean Data

In Korean, the same kinds of unrestricted scrambling can be observed. The examples[8] all show movement of arguments (mainly noun phrases, but Korean also has movement of adverbial phrases as in example (2)).

The first example shows that an argument of an embedded sentence can move to any position in the matrix clause. This example parallels example (1) in section 2.4.1.

The canonical sentence is taken from a well-known old Korean story "A rabbit's liver". The sentence might not make any sense to the reader but a Korean audience will understand it. Already knowing the semantics of the sentence beforehand helps greatly in overcoming difficulties in understanding the sentence and allows to concentrate on the syntactic acceptability. Sentences with the same syntactic structure but a meaning which is not already known to the reader can be a lot harder to understand and might even be judged unacceptable.

(1a) thokki$_i$ –ga yongwang –eykey [PRO$_i$ caki$_i$ –uy kan –ul
rabbit NOM king-of-the-sea DAT self POSS liver ACC

kacie–o –keyssta –ko] yaksokhayssta.
bring FUTURE COMP promised

(1b) thokki$_i$ –ga [caki$_i$ –uy kan –ul]$_j$ yongwang –eykey [PRO$_i$ t$_j$
rabbit NOM self POSS liver ACC king-of-the-sea DAT

kacie–o –keyssta –ko] yaksokhayssta.
bring FUTURE COMP promised

(1c) [caki –uy kan –ul]$_j$ thokki$_i$ –ga youngwang –eykey [PRO$_i$ t$_j$
self POSS liver ACC rabbit NOM king-of-the-sea DAT

kacie–o –keyssta –ko] yaksokhayssta.
bring FUTURE COMP promised

[8] I am indebted to Young-Suk Lee of the University of Pennsylvania for providing the Korean examples.

"The rabbit promised the king of the sea to bring his/her own liver."

The following example parallels example (2) form section 2.4.1.

(2a) na$_i$ –nun Kim kyoswunim –kkey [pro$_i$ naycwu –kkac
I TOP professor Kim DAT-honorific next week by

i project –lul mamwulicis –keyssta –ko] yaksokhayssta.
this project ACC finish up FUTURE COMP promised

(2b) [i project –lul]$_j$ na$_i$ –nun naycwu –kkaci$_k$ Kim kyoswunim –key
this project ACC I TOP next week by professor Kim DAT-hon

[pro$_i$ t$_k$ t$_j$ mamwulicis –keyssta –ko] yaksokhayssta.
finish up FUTURE COMP promised

"I promised Professor Kim to finish up this project by next week."

Notice that one of the scrambled elements "naycwu-kkaci" ("by next week") is not an argument of the embedded verb. But sentence (2b) is still a valid example of unboundedness of type 1 as defined in conclusion 2.4.1, since the postpositional phrase "by next week" clearly belongs to the embedded clause. Also the postpositional phrase seems to be an optional argument rather than a pure adverbial phrase.

The following example parallels example (3) form section 2.4.1.

(3a) na –nun [younghee –ka [chelswu –ka ku huykwypon –ul
I TOP NOM NOM that rare book ACC

hwumchie-kassta –ko] cwucanghayssta –ko] malhayssta.
steal-and-ran-away COMP claimed COMP said

(3b) [ku huykwypon –ul]$_i$ na –nun [younghee –ka [chelswu –ka t$_i$
that rare book ACC I TOP NOM NOM

hwumchie-kassta –ko] cwucanghayssta –ko] malhayssta.
steal-and-ran-away COMP claimed COMP said

"I said that younghee claimed that chelswu stole the rare book and ran away."

In sentence (3b), one argument from the most deeply embedded sentence ("rare book") has been scrambled into the matrix clause.

A prediction that the FO-TAG analysis which will be introduced in section 4.1 makes is that one can move the embedded verb beyond an argument of the matrix verb, see the following variant of example (1):

(4)

thokki$_i$	–ga	[caki$_i$	–uy	kan	–ul]$_j$	[PRO$_i$	t$_j$
rabbit	NOM	self	POSS	liver	ACC		

kacie–o	–keyssta	–ko]	yongwang	–eykey	yaksokhayssta.
bring	FUTURE	COMP	king-of-the-sea	DAT	promised

"The rabbit promised the king of the sea to bring his/her own liver."

Here, the embedded verb "kacie-o-keyssta" ("will bring") is moved beyond the noun-phrase "yongwang-eykey" ("to the king of the sea") which is an argument of the matrix verb "yaksokhayssta" ("promised").

Chapter 3

Formal Characterization of Long-Distance Scrambling

In this chapter, I give a formal characterization of the Long-Distance-Scrambling phenomenon as described in section 2.4. It turns out that the classical approaches for the formal characterization of a phenomenon in natural language are not useful in this case.

The simplest common approach is the mapping of the natural language to a formal language. This equality of string languages is also called "weak equivalence." A classic example is the argument in [Shi85] where he shows that the set of sentences of Swiss German cannot be generated by a context-free grammar (CFG). For another example see the arguments in section 2.2. A more complex approach, the comparison of the syntactic structures (i.e. the phrase-structures) of sentences is also called "strong equivalence." For an example see the arguments of [Mil91] about the syntactic structures of Swedish (see section 2.1), also [BKPZ83] argues that the syntactic structures of Dutch cannot be generated by a CFG, an argument which can be extended to TAGs.

The next section shows why both approaches fail to show the inadequacy of TAGs for the description of the LDS phenomenon. Both approaches do not allow to directly express co-occurrence restrictions, e.g. between a verb and its arguments. Such minimal semantic information can be expressed with the notion of "derivational generative capacity" which is introduced in section 3.2. By using the "derivational generative capacity" does it become possible to show that the LDS phenomenon is beyond the "derivational generative capacity" of TAGs, even beyond that of the whole class of LCFRS. The proof is given in section 3.3. This result forces us to look for extensions of the TAG formalism.

This chapter is based on joint work with O. Rambow (see also [BR90]) and M. Niv (see also [BRN92]).

3.1 The Weak and Strong Generative Capacity of TAGs

3.1.1 Weak Generative Capacity

Following the approach laid out by Shieber in [Shi85], one may inquire into the formal properties of the string language of German, given the unconstrained scrambling which German seems to exhibit (see section 2.4). I briefly review Shieber's argument.

Shieber shows that the string language of Swiss German, which exhibits both syntactic case markings and cross-serial dependencies, cannot be a CFL. His argument makes no use of semantic notions such as dependencies (which would manifest themselves in the form of restrictions on possible grammars). Shieber defines a homomorphism from German noun-phrase (NP) strings to a finite language, mapping accusative NPs to a, dative NPs to b, verbs that subcategorize for a subject NP (which, in embedded sentences, is PRO), an accusative NP and an S to c, verbs that subcategorize for a subject NP (which, again, is PRO), a dative NP and an S to d[1]. He then intersects the image of the homomorphism with the regular language $R_1 = wa^*b^*xc^*d^*$, and ends up with $L_1 = wa^mb^nxc^md^n$, which is not a CFL. He concludes that no CFG can generate Swiss German, i.e., it is weakly non-context free.

There are two crucial points in Shieber's argument:

1. Because Swiss German NPs are morphologically case-marked for dative and accusative cases, the homomorphism can be defined without reference to semantic dependency on the verb. Earlier attempts on showing that Natural Languages (Dutch) are not context-free used such semantic dependencies (and structural arguments). Furthermore, Shieber assumes that the subcategorization frame of a verb is unique[2].

[1] Actually, the mapping Shieber proposes is more restricted than that: he maps particular NPs and verbs to the formal language. We adopt a slightly more general approach here.

[2] This is not a valid assumption. As Manaster-Ramer also observes ([MR88]), the accusative or dative argument is optional for most verbs that subcategorize for a subject, an object and a clause. So in an embedded sentence with two verbs that both subcategorize for, say, an accusative argument and that has only one such argument, it is not possible to decide which verb it is an argument of. This complicates the formal argument a little but it does not invalidate it. In the argumentation this observation results in a formal language $L_1' = wa^{m_1}b^{n_1}xc^{m_2}d^{n_2}$, where $m_1 \leq m_2$ and $n_1 \leq n_2$, which can also be shown not to be a CFL.

2. There are, in Swiss German, verbs that subcategorize for nominative NP (PRO in embedded sentences), accusative NP and S, and also verbs that categorize for nominative NP (PRO in embedded sentences), dative NP and S.

3.1.2 German is Weakly Non-Context Free

Exactly the same argument can be advanced for (High) German, since in German points 1 and 2 above hold. This is may not be obvious since the unmarked structure of German sentences is center-embedding and it so does not exhibit cross-serial dependencies *per se*. But the free scrambling that may occur from center-embedded clauses each containing an accusative or dative object (in addition to PRO and the embedded S) will yield sentences with the same kind of cross-serial dependencies as in Swiss German. This allows the mapping by a similar homomorphism and intersection with the regular language R_1, yielding the same non-CF language L_1 as in the Swiss German case.

To better reflect all the word orders that scrambling leads to, one can intersect with the regular language $R_2 = w(a+b)^*x(c+d)^*,$. This yields the language

$$\begin{aligned} L_2 = \{\ wz_1xz_2y \ \mid\ & z_1 \in (a+b)^*, z_2 \in (c+d)^*, |z_1| = |z_2|, \\ & \#a\text{'s in } z_1 = \#c\text{'s in } z_2, \text{ and } \#b\text{'s in } z_1 = \#d\text{'s in } z_2\}. \end{aligned}$$

Like L_1, L_2 is not a CFL, but it is a TAL. In fact, it is the language generated by (a close variant of) the grammar given as example 2.4 in [JVSW90].

3.1.3 Limitations of Weak Generative Capacity

Can it be shown, without resorting to semantic notions, that German is not a TAL? In German, there are exactly two types of morphological markings on NPs that relate them to the verb of which they are an argument. Therefore all arguments are case-marked, and the subject agrees in person and number with the finite verb. Since in German scrambling is possible only out of non-finite clauses, subject-verb agreement is not available. Thus, case marking is the only syntactic clue that can be used in the type of analysis proposed by Shieber.

In general this seems possible. One would have to find a language in which there is a verb subcategorizing for nominative NP (PRO in embedded sentences), accusative NP, genitive NP and S, and a verb subcategorizing for nominative NP (PRO in embedded sentences), dative NP, and a forth case, say ablative NP and S. One could then derive the formal language

$$L_3 = \{\ wz_1xz_2y \ \mid\ \ |z_1| = |z_2|, z_1 \in (a+b+c+d)^*, z_2 \in (e+f)^*,$$

$\#a's$ in $z_1 = \#b's$ in $z_1 = \#e's$ in z_2, and $\#c's$ in $z_1 = \#d's$ in $z_1 = \#f's$ in $z_2\}$.

This language, a version of the triple-copy language, is not a TAL. However, there is no ablative case in German, nor do verbs exist in German that subcategorize for S and for more than two NPs. Since the nominative NP must be PRO in the non-finite clauses from which scrambling is allowed, it will be impossible to map German into a language like L_3. However, a language with a richer case system, and perhaps in which scrambling out of finite clauses is allowed, may map into L_3.

Given the facts about syntactic markings of German NPs, and the range of subcategorization frames of German verbs, we conclude that it cannot be shown that the string language of the German LDS phenomenon is not a TAL.

From a linguistic point of view, the mere existence of *some* TAG to generate the set of German strings, or even some postulated set of phrase structures, is of little interest. The crucial question is whether a derivation exists in a *linguistically plausible* grammar. We assume that any linguistically plausible grammar will meet the co-occurrence restrictions outlined below. In [BR90], we have shown that no TAG can be defined which meets the co-occurrence restrictions and which derives all indexed strings representing scrambled German sentences. In chapter 4, we finally introduce two new variants of TAG which do.

3.1.4 Strong Generative Capacity

The obvious next question is whether it can be shown that the syntactic structures of scrambled German sentences cannot be strongly generated by a TAG.

Most linguistic analyses, as disparate as they are in other points, agree that the resulting phrase structure tree is simply a single right-branching spine. With the above argument (section 3.1.2) about cross-serial dependencies in scrambled German sentences and the argument in section 2.1 about right-branching, cross-serial structures in Swedish, it could easily be shown that TAGs cannot generate the syntactic structures of the LDS phenomenon in German. But as already noted in section 2.1 the argument cannot be generalized any further. Actually, a tree-local MC-TAG which has tree sequences with no more than two trees is sufficient to generate the right-branching structures for the cross-serial dependencies.

In the next section, a new notion of generative capacity will be introduced that allows us to give a formal proof of a much more general result about the complexity of the LDS phenomenon.

3.2 The Notion of Derivational Generative Capacity

In this section[3], we introduce a third way of arguing about the formal adequacy of formal systems, the "derivational generative capacity" of a system.

The starting point for our definition is the observation that while the specifics of a syntactic analysis of a particular construction may be subject to controversy (such as the issue of verb raising in Dutch), the (semantic) predicate-argument structure of sentences is fairly uncontroversial. Furthermore, what characterizes many syntactic phenomena in natural languages is the way in which the predicate-argument structures map onto the surface strings. For example, a crucial difference between Dutch and German clausal embedding is the fact that Dutch exhibits cross-serial dependencies between verbs and subcategorized nouns, while the dependencies are nested in German. This statement holds independently of whether or not verb raising is assumed to occur in Dutch and/or in German. 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.

In TAGs, however, where the elementary structures are trees, the domain of locality of the elementary structures is extended beyond that of the context-free rule. In a TAG, it is easy 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. (This restriction is called "co-occurrence constraint" in [BJR91].)

Co-occurrence constraint:

An elementary structure of the grammar includes exactly one lexical head and (positions for) its complements.

It follows that semantically related symbols are contributed by the same elementary structure in the grammar, i.e. they are generated in 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.

[3] This and the following section are based on joint work with O. Rambow and M. Niv, see [BRN92], also presented as [RBN92].

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 that a system can 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. In the following, we will present a formalization of constraints on derivations, which we will then use to represent predicate-argument relations in scrambled strings in German.

3.2.1 A Formalization of Derivational Constraints

Definition 3.2.1 (indexed string)

An **indexed string** *over an alphabet T is a finite string $w \in T$, 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 indices on the symbols of w.*

A simple example for an indexed string is $w = a_1a_2a_3b_1b_2b_3$, where the co-indexation of a's and b's can be seen as representing cross-serial dependencies.

The following definition of deriving an indexed string strongly depends on the kind of grammar formalism in question.

Definition 3.2.2 (derivation of an indexed string)

Let w^{ind} be an indexed string and G a grammar in some grammar formalism GF. G **derives the indexed string** w^{ind} *iff there is a derivation of w in G (as defined by GF) in which all symbols bearing the same index are contributed from the same elementary tree sequence, and any derivation step contributes only symbols bearing the same index. The* language of indexed strings, *$L^{ind}(G)$ is the set of all indexed strings that can be derived by G. A grammar G has the* **derivational generative capacity** *to derive an indexed language L^{ind} if $L^{ind} = L^{ind}(G)$.*

Here are some examples:

1. In a CFG all (terminal) 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 (terminal) symbols bearing the same index (and no others) must be generated by adjunction of the same elementary tree[4].

3. In a Multi Component-TAG, all symbols bearing the same index (and no others) must be generated by multi-component adjunction of the same elementary tree sequence.

4. 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 the symbols of w^{ind} corresponding to the derivation step (and no others).

We now can 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 is simply {N,V}.

Definition 3.2.3 (Formalization of LDS)

$$SCR := \{\sigma(N_0, ..., N_n)V_0...V_n \mid n{>}0 \text{ and } \sigma \text{ a permutation}\}$$

The corresponding string language (without indices) of SCR is simply the language $\{N^m V^m \mid m{>}0\}$. 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.

3.3 LDS is beyond LCFRS

In this section, it is shown by contradiction that no LCFRS has the derivational generative capacity to derive SCR. The idea of the proof is as follows: We assume

[4]For the following prove to hold, this implies that the elementary structure for a verb already includes all its noun phrases. This contradicts the idea of a lexicalized grammar, but is otherwise no problem. One could also adjust the definition such that all terminal symbols that are generated by the substitution of an elementary tree must bear the same index as the symbols in the tree into which the substitution takes place. I.e. a substituted noun-phrase argument of a verb will bear the same index as the verb.

that there is some LCFRS-k grammar G that derives SCR. Then a new grammar G′ is constructed that introduces k′ subscripts into the terminal alphabet of G′. This allows to interpret the derivational generative capacity in terms of weak generative capacity and from this point on, the proof only deals with string languages. From G′, an LCFRS-k grammar G‴ is derived by intersecting with a regular language and by applying a homomorphism such that G‴ derives the language COUNT-k′={a_1^n,...,$a_k'^n$ | n>0}. Since k′ was chosen 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.

Now, the single steps follow in more detail:

Theorem 3.3.1 (LDS is beyond LCFRS)

There is no LCFRS G such that G has the derivational generative capacity to derive the indexed language SCR.

Proof:

(1) Suppose there is an LCFRS-k grammar G = (C,E) with C = (V,S,F,P) that derives SCR. The alphabet T of G is {N,V}.

(2) 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 $k' * (k')^l = (k')^{l+1}$ 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 terminal symbols introduced bear the subscript i. We have:

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

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 [SMFK91], there is an LCFRS-k grammar G″ that derives the language which results from the intersection of L(G′) with the regular language R:

R = $[n_0]^*$...$[n_{k'-1}]^*[v_0 \ldots v_{k'-1}]^*$

$L(G'')$ is therefore $\{[n_0]^j ... [n_{k'-1}]^j [v_0 ... v_{k'-1}]^j \mid j{>}0\}$. A string of $L(G'')$ has the following form: a sequence of j occurrences of each of n_0 through $n_{k'-1}$, followed by j copies of the string $v_0 ... v_{k'-1}$.

(4) Since the LCFRL-k are also closed under application of homomorphisms [SMFK91], there is an LCFRS-k grammar G''' that derives the language that results from the application of the following homomorphism to $L(G'')$:

$h(v_i) = \epsilon$, $h(n_i) = n_i$, where ϵ denotes the empty string.

The resulting language is

$L(G''') = \{[n_0]^j ... [n_{k'-1}]^j \mid j{>}0\}$

which is COUNT-k′. The pumping lemma for LCFRS [SMFK91] can then be used to show that COUNT-k′ is not in LCFRL-k. So we have derived a contradiction.

Therefore no LCFRS has the derivational generative capacity to derive the indexed Scrambling Language SCR. □

The next chapter discusses two approaches to an extension of the TAG formalism which has the derivational generative capacity to generate SCR and can therefore adequately capture the LDS phenomenon.

Chapter 4

Two New Types of TAG for Free Word Order Languages

This chapter presents two new formalisms which are both extensions of the TAG formalism. Both new formalisms, called Free-Order TAGs (FO-TAG) and MC-TAGs with Dominance Links (MC-TAG-DL), have extended means to generate complex word orderings. In particular, they can capture the German LDS phenomenon, i.e. they have the derivational generative capacity to generate the indexed language SCR (see the last chapter).

Both formalisms are generalizations of the TAG formalism, i.e. any TAG grammar can also be expressed as an FO-TAG or an MC-TAG-DL. In a way, both new formalisms are based on the relaxation of some constraints which are completely deterministic in TAGs.

An elementary tree in a TAG can be seen (and, if one so desires, formally defined) as a set of nodes, along with an immediate dominance relation (ID) and a linear precedence (LP) relation defined on this set. For each node there is exactly one node that immediately dominates it, except for the root node, which is dominated by no node. All sister nodes are (completely) ordered by LP, and the LP relation is inherited by daughters. In order to define formalisms that generate the syntactic variations that scrambling allows while maintaining co-occurrence relations, it is possible to relax either the requirements for the ID relation or for the LP relation. These approaches will require the redefinition of the adjunction operation, but will yield formalisms that in spirit and appearance are close to the original TAG formalism.

The first formalism, FO-TAG which is defined in section 4.1 still maintains the concept of a tree as a dominance structre, but unlike in TAGs, the linear order of the nodes (especially of the leaves) is no longer completely determined. Only a set of so-called "Linear Precedence" rules needs to be obeyed.

The second formalism, MC-TAG-DL which is defined in section 4.2 is based on MC-TAGs (see section 1.3). A tree sequence of an MC-TAG-DL can be seen as being derived from an elementary tree of a TAG. The tree structre and node ordering are kept, but the elementary tree is broken up into parts which can be adjoined at different positions, resulting in more complex word orderings. In addition to regular MC-TAGs, in MC-TAG-DL some additional dominance constraints between the trees of one tree sequence have to be obeyed.

4.1 The Free Order TAG (FO-TAG) Approach: Relaxed Linear Precedence

The first formalism, which will be called FO-TAG (Free Order TAG), is based on the LD/LP-TAG framework presented in [Jos87b]. I will first sketch the definition of the LD/LP formalism and then discuss the extensions that are motivated by the long-distance scrambling data.

4.1.1 The LD/LP Formalism

An LD/LP-TAG grammar as defined in [Jos87b] 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 LP rules. The LD structure (which will, imprecisely, be referred to as a "tree" in this section) is either an initial or an auxiliary tree. The LP rules may relate any two nodes of the tree unless one dominates the other. As a default, a node inherits all of its mother's LP rules. The default can be explicitly overridden by stating an LP rule that specifies a different precedence.

However, these precedence rules can only be stated with respect to the nodes of an elementary structure; it is not possible to relate nodes in different structures. When an auxiliary structure T_1 is adjoined into an initial structure T_2 at node A, the nodes of T_1 inherit any LP rules specified for node A. If no LP rule is specified between node A and some other node of T_2, then the nodes of T_1 are not ordered with respect to that node. This effect, which accounts for the increased generative power of this formalism, we will call "indeterminacy."

Note that unlike some other formalisms which also use linear precedence rules (e.g. Generalized Phrase Structure Grammar (GPSG), see [GKPS85]), in LD/LP TAGs as well as in FO-TAGs, there is no general set of LP rules that apply for all ID rules. Rather, each elementary structure has its own set of LP statements.

4.1.2 Necessary Extensions to the LD/LP Formalism

Indeterminacy vs. Determinacy

A formalism with indeterminacy turns out to be not powerful enough. It allows only either a very free ordering of the nodes or a very restricted one. Especially it is impossible to adequately restrict the position of nodes of an adjoined structure in the initial structure (see also the discussion in the next subsection on integrity). One possibility is to give the following constraint on the definition:

*If two nodes have no LP rule specified for each other (neither explicitly nor by inheritance), then one of the two possible orderings must hold (***determinacy** *of ordering).*

In this view, the initial structures become "macros" of sorts for a larger set of structures whose linear precedence relations have been completely specified; but then the formalism is equivalent to a TAG, and too restrictive.

To handle the unbounded scrambling phenomena, we need to keep indeterminacy in the FO-TAG formalism. However, this alone is not enough, the free orderings that indeterminacy allows need to be restricted somewhat, as the next paragraph shows.

Integrity

It turns out that in LD/LP-TAG (as defined in [Jos87b]), it is impossible to restrict the possible orderings in a manner adequate for linguistic description. Especially the possible positions of the nodes of an adjoined structure with respect to the structure that it is adjoined into cannot be constrained adequately. For example, if two NPs are unordered with respect to each other, then (because of indeterminacy) the leaves of one NP may be interspersed among the leaves of the other[1]. In order to restrict such interactions, we introduce for the FO-TAG formalism a new unary predicate on nodes, the integrity constraint, written 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. Formally,

if dom *is the domination relation partially ordering the nodes of the tree, then*

$$\begin{aligned} \Delta X \quad :\Leftrightarrow \quad & \forall\delta([\neg(X \;\mathsf{dom}\; \delta) \wedge \neg(\delta \;\mathsf{dom}\; X\,)] \\ & \Rightarrow [(\forall\epsilon(X \;\mathsf{dom}\; \epsilon) \Rightarrow \delta < \epsilon) \vee (\forall\epsilon(X \;\mathsf{dom}\; \epsilon) \Rightarrow \delta > \epsilon)]). \end{aligned}$$

This extension of the LD/LP formalism adds to the generative power, as we will

[1]This is allowed only in a very few languages with almost completely free word order like Warlpiri. In almost all other languages an NP has to be a contiguous string.

discuss in section 4.1.5.

It is interesting to note that if all the root nodes of all structures in an FO-TAG grammar are marked with the integrity constraint Δ, then the result is a TAG-equivalent grammar.

Inheritance

In FO-TAG, as in LD/LP-TAG, there is indeterminacy. However, there is no inheritance by default. Inheritance is necessary only for some nodes but not wanted for others. In the following, nodes that inherit the LP information are explicitly marked with the inheritance constraint, written as $.

Figures 4.3 and 4.4 give a good example of this situation. After substitution of the second structure into the first, there is no information in the second structure that prohibits its S and NP nodes from "moving" to the right of the V_1 node of the first structure. Note that the S substitution node in the first structure is *not* marked with the integrity constraint.

To prevent all nodes from moving to the right of V_1, in the second structure of figure 4.3 (the structure α_2) only the V_2 node really needs to inherit the precedence information of the S node. These precedence rules ($S^b < V_1$, $NP_1^1 < S^b$, $NP_1^2 < S^b$) are inherited by S_2 as it is substituted into S_b. Further, V_2 inherits these rules resulting in three new rules: $V_2 < V_1$, $NP_1^1 < V_2$, $NP_1^2 < V_2$. As a result also the NPs of the second structure (which have to be to the left of V_2) also have to stay left of V_1. Note that there is no need for the NP nodes in α_2 to inherit any of the precedence information.

Although in some cases inheritance can be forced by the integrity constraint, we need to add a mechanism that allows the selection of inheritance only for specific nodes.

Inheritance can be forced for any two nodes in a structure that are not in a dominance relation. If they are both marked with the integrity constraint (which also suspends indeterminacy for that node) inheritance results as a side-effect. However, if only one of two such nodes has the integrity constraint, then inheritance cannot be achieved. In such cases (at most one of two nodes has an integrity constraint), inheritance must be stated explicitly. In our example in figures 4.3 and 4.4 this is the case for the S^b node and the NP nodes of the elementary structure α_1. To make any node in α_2 inherit precedence information from α_1 after substitution this must be stated explicitly by marking every node with the inheritance constraint, as it is done for V_2.

4.1.3 Formal Definition of FO-TAG

Many of the definitions are similar to the definitions of the corresponding notions for standard TAGs. So sometimes only the differences might be stated explicitly. In analogy to the definition in [Jos87b] we will call the pair of local dominance and linear precedence statements an *elementary structure*. So *tree* in the following refers only to the local dominance structure.

Definition 4.1.1 (precedence statement, FO-structure)

A **precedence statement** *is a term of the form*

$$A < B$$

where A and B are tree addresses (or node names).

An **initial FO-structure** *is a pair consisting of a tree and a list of* **precedence statements** *over the nodes of the tree. All inner nodes of the tree are nonterminals of the grammar. All leaves of the tree are terminals (or ϵ). Each inner node may be marked with the* integrity constraint, *written as Δ and/or the* inheritance constraint, *written as \$.*

An **auxiliary FO-structure** *is a pair consisting of a tree and a list of* **precedence statements** *over the nodes of the tree. All inner nodes of the tree are nonterminals of the grammar. There is exactly one leaf of the tree (the* **footnode***) that is a marked with the same nonterminal as the root of the tree. All other leaves are terminals (or ϵ). Each nonterminal node may be marked with the* integrity constraint, *written as Δ and/or the* inheritance constraint, *written as \$.*

Note that the tree that is part of an FO-structure only describes the dominance structure of the FO-structure. The (arbitrary) order of the nodes that is chosen for representing a tree is not relevant. So the FO-structures in figure 4.1 are actually identical, the order of the nodes is determined only by the precedence statements. However, in the example of figure 4.1 the second version (α_2), although formally correct, is very misleading and in general the dominance tree should be as close to (at least one) admissible ordering of the FO-structure as possible.

Definition 4.1.2 (FO-TAG)

An **FO-TAG** *G is an 5-tuple $G = (N, T, S, A, I)$ with*

- *N is a nonempty, finite set of* nonterminal symbols,

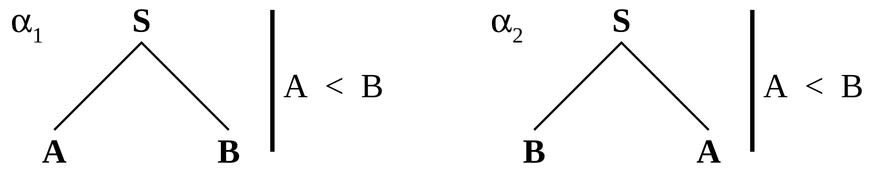

Figure 4.1: Two identical initial FO-structures. Although α_2 is formally correct, it is misleading.

- *T is a nonempty, finite set of* terminal symbols,
- *N and T are disjoint sets,*
- *S is a distinguished element of N, the* starting symbol,
- *I is a nonempty, finite set of* initial FO-structures *and*
- *A is a finite set of* auxiliary FO-structures.

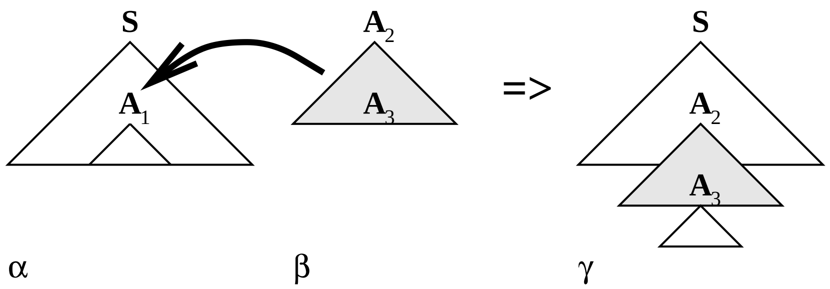

Figure 4.2: Adjunction of the (local) dominance trees. β is adjoined into α at node A_1, the result is γ. Note that the indices are only used to identify the tree addresses, all nodes shown are marked with either the nonterminal A or S

Definition 4.1.3 (Adjunction in FO-TAG)

The **adjunction** *of an auxiliary FO-structure β into an initial FO-structure α (that may already be a derived FO-structure) at node A results in a new,* derived *initial FO-structure γ. γ has the same dominance structure as the standard TAG-adjunction of the dominance structure of the auxiliary FO-structure into the dominance structure of the initial FO-structure would yield. The linear precedence statements of the derived FO-structure are constructed as follows:*

- *All linear precedence statements of the initial FO-structure that do not concern the node A_1, are copied to the derived FO-structure.*

- *For each linear precedence statement in the initial FO-structure that concerns A_1, two copies of the linear precedence statement are added to the derived FO-structure where the address of A_1 is substituted with the addresses of the root node A_2 and the foot node A_3 of the auxiliary FO-structure respectively.*
- *All linear precedence statements of the auxiliary FO-structure are added to the derived FO-structure.*
- *In addition to this, for each node B in the auxiliary FO-structure which is marked with an inheritance constraint a copy of all linear precedence statements that concern the root node A_2, where the address of A_2 is substituted by the address of B, is added to the derived FO-structure (B inherits all precedence rules).*
- *If the node A_1 in the initial FO-structure at which adjunction takes place is marked with the integrity constraint then the root node*[2] *of the auxiliary FO-structure in the derived FO-structure is also marked with the integrity constraint.*

Note that the resulting FO-structure is again an initial structure, but not an elementary structure in the set I of G.

Definition 4.1.4 *An* **admissible ordering** *of a (derived) FO-structure t is a total ordering of all nodes of t that are not in a dominance relation. All the linear precedence statements of t must hold in this total ordering.*

The **structure set** $T(G)$ *of an FO-TAG is the set of all initial FO-structures and all derived FO-structures.*

The **language** $L(G)$ *of an FO-TAG is the set of the frontiers*[3] *of* admissible orderings *of structures in the structure set $T(G)$.*

The FO-TAG formalism can easily be extended to include a substitution operation and feature structures.

4.1.4 FO-TAG Handles Scrambling

As an example for the FO-TAG formalism, consider sentences (2a) and (2b) of section 2.4, repeated here for the convenience of the reader:

[2] It is not clear if this should also be true for the foot node. The linguistic motivation behind our definition is that adjunction to the topmost node of a category that is marked with Δ should not constrain the integrity of subcategories but only of the whole category. If this is needed, the auxiliary structure can always contain additional integrity constraints.

[3] Or more precisely the (totally ordered) sequence of terminal symbols on the leaf-nodes.

(2a) ... daß der Detektiv$_i$ dem Klienten [PRO$_i$ den Verdächtigen
... that the detective (nom) the client (dat) the suspect (acc)
des Verbrechens zu überführen] versprochen hat.
the crime (gen) to indict promised has

(2b) ... daß [des Verbrechens]$_k$ der Detektiv$_i$ [den Verdächtigen]$_j$
... that the crime (gen) the detective (nom) the suspect (acc)
dem Klienten [PRO$_i$ t_j t_k zu überführen] versprochen hat.
the client (dat) to indict promised has

"... that the detective has promised the client to indict the suspect of the crime."

The initial structures along with the LP rules and integrity restrictions are shown in figure 4.3.

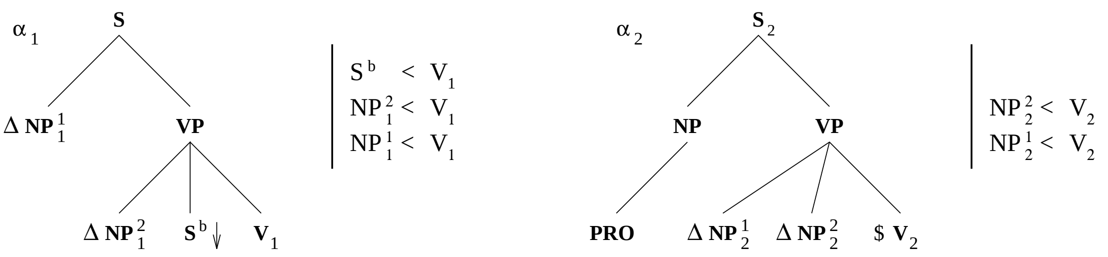

Figure 4.3: The initial structures for deriving sentences (2a) and (2b) in the FO-TAG formalism.

Adjunction yields (among other admissible orderings) the structure shown in Figure 4.4. This ordering is exactly the one of sentence (2b).

As the example in figures 4.3 and 4.4 shows, FO-TAGs can handle the case of scrambling of two arguments of the embedded clause between the arguments of the matrix clause. This is the case of sentence (3b) in section 2.4.1 which standard TAGs cannot handle with the co-occurrence restrictions. FO-TAGs can also handle the case of unbounded scrambling of arguments of embedded sentences with one nominal argument each. See figure 4.5 for a grammar that can derive the indexed string $w = N_3 N_1 N_5 N_2 N_4 V_5 V_4 V_3 V_2 V_1$.

It is obvious that any indexed string of the scrambling language SCR as defined in section 3.2 can be derived by the grammar in figure 4.5.

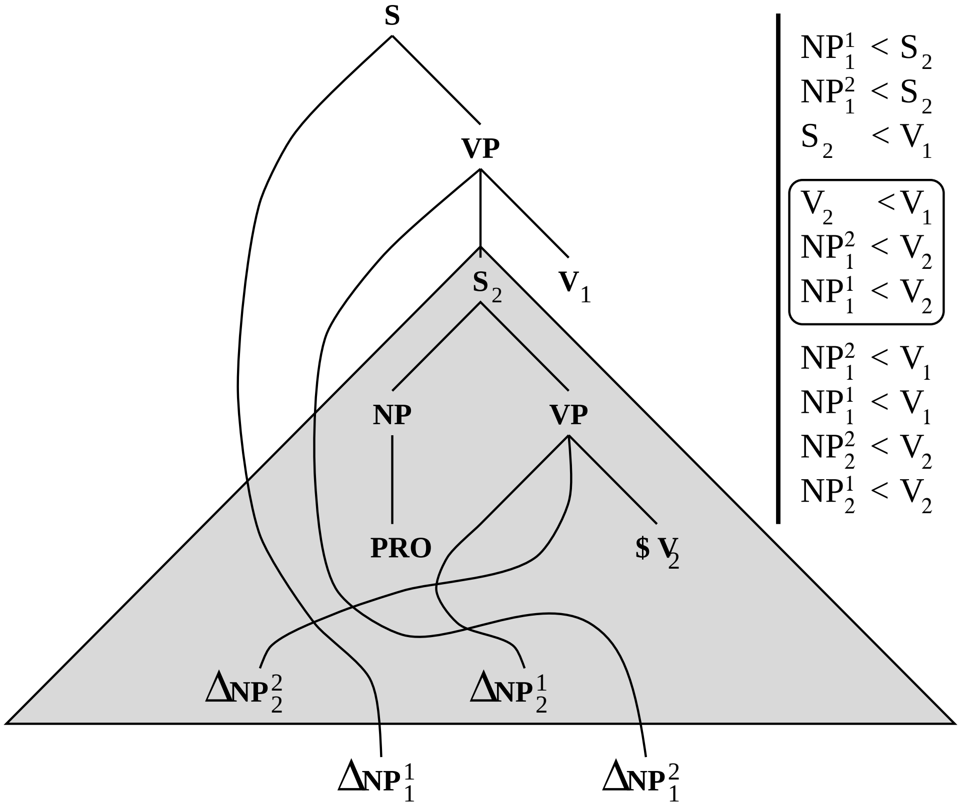

Figure 4.4: Sentence (2b) in FO-TAG. Result of substituting α_2 into α_1. Note the boxed precedence rules that node V_2 inherited from S_2.

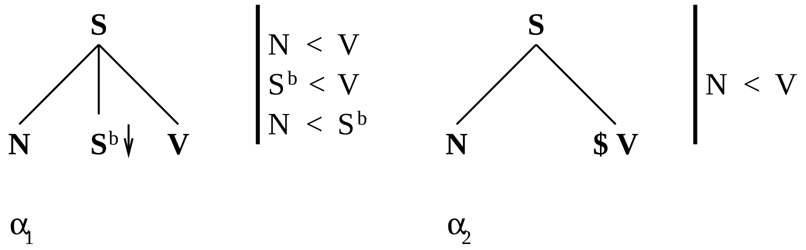

Figure 4.5: FO-TAG grammar that derives the indexed string $w = N_3N_1N_5N_2N_4V_5V_4V_3V_2V_1$.

4.1.5 Formal Properties of FO-TAG

Merely by showing that FO-TAGs can handle the problematic cases of scrambling, nothing is said about the weak or strong generative power of the FO-TAG formalism. Two things need to be checked: First, is TAL a (proper) subset of FO-TAL? (We don't want to loose any possibilities that the original TAG formalism offered.) And second, how much stronger than the original TAG formalism is the FO-TAG formalism? (We don't want to increase the power too much, e.g. a Turing equivalent formalism is not interesting.) It appears that the generative power of the FO-TAG formalism lies between the original TAG formalism and the MC-TAG formalisms, so FO-TAG seems to be a good candidate for linguistic purposes.

Semilinearity

As the property of being semilinear is independent of the order in which terminal symbols appear in the strings of a language, it is clear that FO-TAG are semilinear and also have the constant-growth property (see e.g. [Wei88]) since if one ignores linear precedence, with respect to dominance structures TAG and FO-TAG are identical.

Generative Power of FO-TAG

Already, the generative power of LD/LP-TAGs seems to be greater than the power of regular TAGs. Adding the inheritance and integrity constraints only increases the power. Still, the power of this formalism is – surprisingly – beneath the power of MC-TAGs. Thus FO-TAGs fulfill the desiderata for a slight expansion of the original TAG formalism that can also handle the scrambling phenomena, but keeps as many of the properties of the original formalism as possible. Note that relaxing the linear precedence relations in the trees is only necessary for scrambling (of NPs) and all other linear precedence relations in a TAG that is changed into a FO-TAG in order to accommodate for scrambling may stay unchanged.

TAL $\subseteq$ FO-TAL

Any TAG can be rewritten as an FO-TAG in which the nodes of every elementary structure are completely ordered and marked either with the inheritance constraint or with the integrity constraint. Then the FO-TAG generates as its admissible orderings exactly the trees of the TAG. So for every TAG there is a strongly equivalent FO-TAG.

Note that if every node is marked with the inheritance constraint, the FO-TAG is equivalent to an LD/LP-TAG which has inheritance by default, so LD/LP-TAL also includes TAL. As FO-TAG is more powerful than LD/LP-TAG[4], by transitivity FO-TAG is more powerful than TAG. In the next section we show this directly.

TAL $\neq$ FO-TAL

FO-TAGs can generate the language

[4]This is actually further work, see section 8.2.

MIX-5 := $\{\ w \mid w \in (a+b+c+d+e)^*, \#a = \#b = \#c = \#d = \#e = n, n \geq 0\ \}$.

Figure 4.6 shows the elementary FO-structures of a grammar that generates MIX-5. Note that there are no precedence rules and no inheritance or integrity constraints.

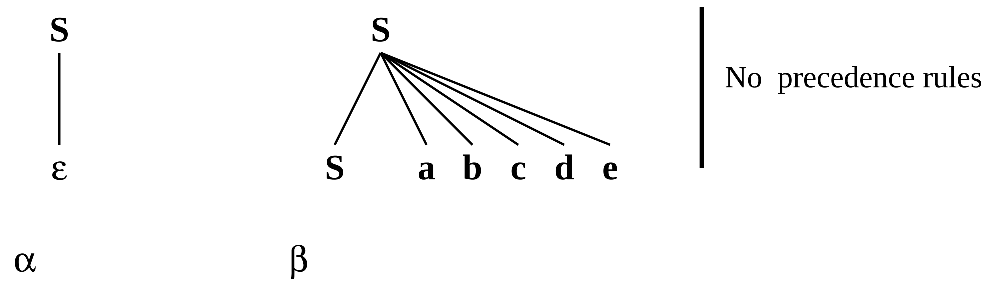

Figure 4.6: An FO-TAG that derives MIX-5.

TALs are closed under intersection with regular languages. But intersection with the regular language $a^*b^*c^*d^*e^*$ yields $a^nb^nc^nd^ne^n$ (L_5 of subsection 4.1.5) which is not a TAL. Therefore MIX-5 cannot be a TAL and TAL $\neq$ FO-TAL.

Corollary: FO-TAL properly includes TAL.

FO-TAL $\subset$ MC-TAL ?

This remains an open problem. A viable approach to proof this inclusion would be to show the inclusion of FO-TAG in MC-TAG-DL (assuming that MC-TAG = MC-TAG-DL). The dominance links of MC-TAG-DL allow ordering of parts of elementary structures and could be used to simulate the precedence rules of an FO-TAG.

FO-TAL $\neq$ MC-TAL

The crucial example here is the language $L_5 = \{w|w = a^nb^nc^nd^ne^n, n \geq 0\}$.

The local MC-TAG in figure 4.7 derives L_5.

No FO-TAG can derive this language. This can be seen from the following argument.

In any FO-TAG that would derive L_5, all auxiliary FO-structures have to have the same number of a's, b's, .. , e's. A node in a tree β at which adjunction takes place recursively can *not* have the integrity constraint because the leaves of one auxiliary structure may *not* stay together, they *have* to be interspersed with other nodes to assure that all a's are grouped together and the b's follow this group and so on. But this means that the nodes of another tree (which may have terminals) can be

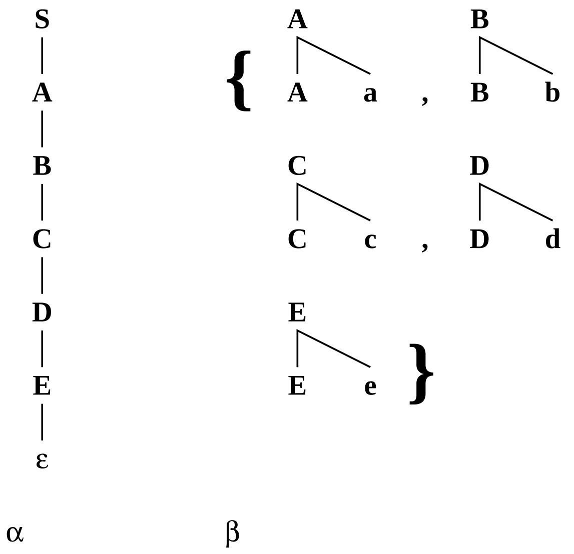

Figure 4.7: A local MC-TAG that derives L_5.

interspersed with the leaves of β in *any* order, thereby destroying the ordering of *all* a's before the b's before the c's and so on.

Corollary: (provided that FO-TAL $\subset$ MC-TAL can be proven) MC-TAL properly includes FO-TAL.

4.2 A Multi-Component TAG (MC-TAG) Approach

The second new formalism, called MC-TAG-DL ("MC-TAGs with dominance links") is based on MC-TAGs (see section 1.3). I will first discuss the motivation for this approach and then define the extensions that are motivated by the long-distance scrambling data and finally show that the resulting formalism can capture the LDS phenomenon.

4.2.1 Relaxing Immediate Dominance

One way of looking at long-distance scrambling is to think of it as a relaxation of the requirement that a verb and its arguments form a contiguous string in the sentence. Structurally, this translates into a relaxation of the immediate dominance (ID) relation of those nodes along the backbone of the clausal tree that dominate the verb's arguments. In a standard TAG, the ID relation between a mother and a daughter node is not necessarily an *immediate* dominance relation because of the possibility of adjoining at the daughter node. The extension to the TAG formalism proposed in this section will allow some of the ID relations of the auxiliary tree to be relaxed when it is adjoined. This can be seen as splitting up the auxiliary tree into parts, while still keeping a dominance constraint between the parts. We define a Relaxed-Dominance Tree (RD-tree) to be an auxiliary tree in which some of the ID relations between nodes are marked as "relaxed" or mediate[5] dominance links. In graphical representations, the mediate dominance link will be indicated by dashed lines. We could then define a grammatical formalism based on RD-trees, by extending the definition of adjunction used in simple TAGs to handle the adjunction of RD-trees. This would be done in a way already defined for MC-TAGs in section 1.3: all parts of the RD-tree are simultaneously adjoined. The fact that we are actually adjoining one tree with relaxed dominance relations rather than an arbitrary tree set means that certain dominance relations hold between the trees in the set. Since in fact the two definitions, RD-TAG and *MC-TAG with dominance links* (MC-TAG-DL) are different ways of looking at the same type of formalism, we will henceforth consider the MC-TAG point of view, since it is already a well known formalism.

In the following section we will consider some issues that enter into the definition of the formalism, then give a formal definition and an example, and finally consider some formal properties of the system.

4.2.2 Definitional Parameters for MC-TAG-DL

There are two issues that need to be addressed in the formal definition: the characterization of "dominance link" and the definition of adjunction.

---

[5]Here, "mediate" is taken as denoting the transitive closure of the ID relation, so that a node immediately dominating another node also mediately dominates it. Note that a node does not dominate itself.

Dominance Links

We want a dominance link between two trees β_1 and β_2 to mean that after adjunction of both trees into some other tree the foot node of β_1 dominates the root node of β_2. There are two possible approaches:

1. We may require that each tree dominates at most one other tree, and that each tree is dominated by at most one other tree, and that every tree dominates or is dominated by some other tree. Furthermore, we could stipulate that if β_1 dominates β_2, then the label on the foot node of β_1 must be the same as the label of the root node of β_2. This definition corresponds most naturally to the intuition of relaxing immediate dominance links in a regular TAG tree.

2. However, we could also (i) allow a tree to dominate more than one other tree and (ii) not require that nodes in different trees which are linked by a dominance relation have the same label. However, (i) means that this version of MC-TAG-DL cannot be expressed as a TAG with relaxed-dominance trees, since then the original tree would have to have more than one foot node.

For the first, less liberal approach the generative power is still unclear (see also section 8.2). For the second, more liberal approach, it seems that an MC-TAG-DL formalism will strongly and weakly include the corresponding (i.e. benign, local or non-local) MC-TAG formalism. Here, "corresponding" means that adjunction is defined in the same way for both systems. Furthermore, linguistic analyses have been proposed (for parasitic gaps [RB90]) that make use of multiple tree dominance. We therefore choose the more liberal approach for our definition.

Definitions of Adjunction in MC-TAG

In addition to the issue of dominance links, there is a second definitional parameter that enters into any definition of MC-TAG-type formalisms, namely the definition of adjunction. Weir [Wei88, p.32] identifies four types (see also section 1.3). The first type ("tree-local," where derived auxiliary tree sequences are adjoined into a single tree) is weakly and strongly equivalent to TAG. In the second type ("set-local"), derived auxiliary tree sets are adjoined into elementary tree sets (or an initial tree), while in the third type ("non-local") elementary auxiliary tree sets are adjoined into derived trees. (The fourth type, in which derived auxiliary tree sets are adjoined into a derived tree, appears to be strongly equivalent to the third type, and we will not discuss it further.) The question arises as to which of theses variants handle scrambling.

We will now argue that tree-local MC-TAG with co-occurrence constraints cannot generate the indexed strings of L_4. In order to do so, we will strengthen the co-occurrence constraint. In arguing the case of simple TAGs, we merely required that each elementary structure contain both a V and an N, and no other terminal (see [BR90]). In a lexicalized, linguistically motivated grammar this means that each clausal structure is anchored by a verb, and subcategorizes for an NP (implemented as a substitution node) as its only overt nominal argument. However, in the context of recursively nested clauses each verb also subcategorizes for a clause (S). Therefore, for the abstract language L_4 we will extend the co-occurrence restriction in the following manner: each elementary structure must contain exactly one N and one V, and, if a structure has index i,[6] the structure into which it is adjoined and the structure which is adjoined into it must have the indices $i-1$ or $i+1$. Following the definition of adjunction in tree-local MC-TAG, this means that in the case of recursively embedded clauses either clause 1 is adjoined into clause 2, which is then adjoined into clause 3, and so on until adjunction into clause n terminates the process (this direction mimics the adjunction of the matrix clause into the subordinate clause in the standard analysis of wh-movement [Kro88]), or, conversely, a derivation starts with adjunction of clause n and terminates by adjunction into clause 1.

Let G_{MC} be an tree-local MC-TAG grammar, and k the size of its largest tree set. Consider the indexed string $w = N_{4k}N_1N_{4k-1}N_2 \dots N_{2k}N_{2k+1}V_{4k}V_{4k-1} \dots V_1$. Assume without loss of generality that we start by adjoining clause $4k$ into clause $4k-1$. It is obvious that we can adjoin the tree containing V_{4k} into the tree containing V_{4k-1}, and the resulting derived tree into the tree containing V_{4k-2}, and so on. We will thus restrict our attention to the N_i. We may adjoin the tree containing N_{4k} to the tree containing N_{4k-1}, since subsequent adjunction to the tree for the matrix clause can still yield the contiguous string $N_{4k}N_1N_{4k-1}$. However, we cannot adjoin the next N, N_{4k-2}: though we can at a later point adjoin the resulting derived tree to the tree containing N_2 to get a derived tree whose yield is $N_{4k}N_{4k-1}N_2N_{4k-2}$, we cannot then adjoin it to the tree of the matrix clause, since the footnode will necessarily be immediately next to N_2. Therefore, after adjunction to the initial tree of the matrix clause, the N_1 will either precede N_{4k}, or be in a position between N_{4k-1} and N_2, or be in a position between N_2 and N_{4k-2}, or be in a position after N_{4k-2}, none of which are correct positions. A similar argument holds for any other attempt to adjoin any derived auxiliary tree containing two N_i terminal nodes into another elementary tree. We conclude that we will need at least one tree in each tree set for two N_i; since there are at most k trees per tree set but $2k$ pairs of N_i, we cannot derive the string, given the extended co-occurrence restriction defined above.

[6] As explained in [BR90], we can assume without loss of generality that the matrix clause has index 1, and if a clause has index i, then the immediately embedded clause has index $i+1$.

4.2.3 Formal Definition of MC-TAG-DL

Definition 4.2.1 (Dominance link)

A **dominance link** *between two trees* β_1 *and* β_2 *of the same tree sequence is a constraint on their multi-component adjunction: it is obeyed if after (simultaneous) adjunction of* β_1 *and* β_2 *into the same tree, the foot node of* β_1 *dominates the root node of* β_2.

Note that the definition above supposes a type 3 (i.e. non-local) definition of adjunction.

Definition 4.2.2 (MC-TAG-DL)

A **Multicomponent TAG with Dominance Links** *(MC-TAG-DL) is a five-tuple* $G = (N, T, S, I, A)$ *where*

- N *is a nonempty, finite set of* nonterminal symbols,
- T *is a nonempty, finite set of* terminal symbols,
- N *and* T *are disjoint sets,*
- S *is a distinguished element of* N, *the* starting symbol,
- I *is a nonempty, finite set of* initial tree sequences *and*
- A *is a finite set of* auxiliary tree sequences.

Within each tree sequence, dominance links between trees may be defined. Adjunction is defined to be the adjunction of an elementary auxiliary tree set into a derived tree such that the resulting structure adheres to the dominance links.

Sequence Internal Adjoining

[Lee92, Lee93] proposes to allow adjoining steps within a tree sequence prior to adjoining the (resulting trees) into a derived structure. Clearly this does not add to the formal power of the formalism since all (finitely many) possible combinations of sequence internal adjoining could be compiled out. This approach allows to generalize the representation of a free word order grammar considerably and is also used in the appendix.

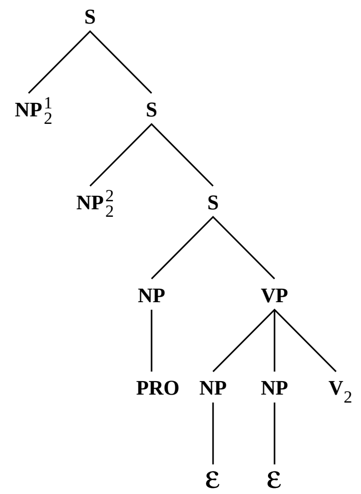

Figure 4.8: Initial tree for MC-TAG-DL derivation

4.2.4 MC-TAG-DL Handles Scrambling

We now show that the formalism defined above does indeed handle the data we presented in Section . We will first illustrate this by showing a derivation for sentence (2b) (see Section 2.4.1) Figure 4.8 shows the initial tree. Figures 4.9 and 4.10 show the auxiliary trees as an RD-tree and an MC-TAG-DL tree set, respectively. Figure 4.11 shows the tree after adjunction.

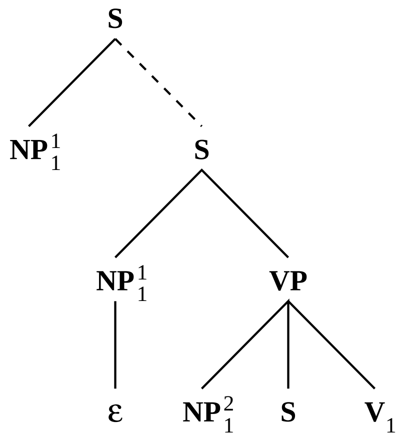

Figure 4.9: Splitting an elementary tree into two parts.

We will now show that the language $L_4 = \{\sigma(N_1, \ldots, N_k)V_k \ldots V_1 \mid k \in \mathsf{N}$ and σ a permutation$\}$, which in [BJR91] (see also [BR90]) we showed could not be generated by a TAG meeting the co-occurrence restrictions, can indeed be generated by an MC-TAG-DL grammar meeting the co-occurrence restrictions. The grammar, G_4, consists of two tree sets, the initial tree in figure 4.12, and the auxiliary tree set in figure 4.13.

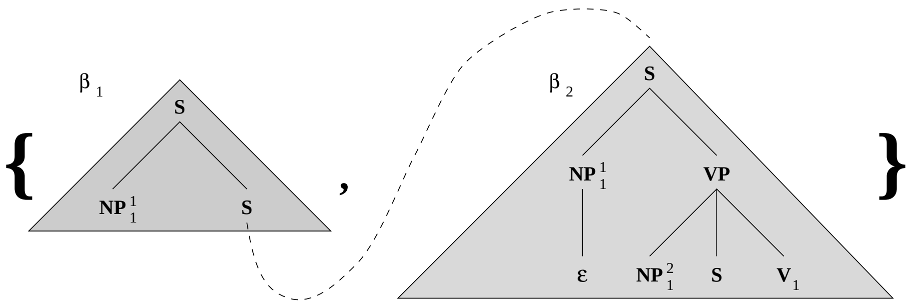

Figure 4.10: The same tree as an MC-TAG-DL tree sequence.

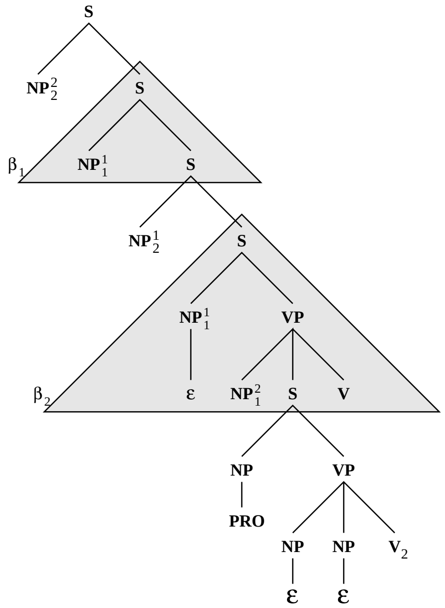

Figure 4.11: Adjunction of an MC-TAG tree set

Features, not shown here, assure that the bulk of the clause is only adjoined in the S node below the VP node.

It can easily be seen that any indexed string in L_4 can be generated by G_4: any step in the derivation will adjoin the tree(s) for the next more deeply embedded clause. At each such step, nodes labelled S are available between all arguments of the already adjoined clauses, and all of these nodes dominate the S node at which the body of the clause must be adjoined.

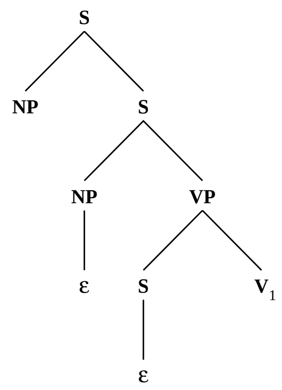

Figure 4.12: Initial tree of G_4

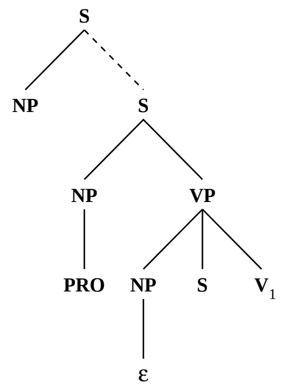

Figure 4.13: Auxiliary tree of G_4

4.2.5 Formal Properties of MC-TAG-DL

General Points

The formal properties of the systems defined above are largely unclear. Changing a definition slightly may lead to a great change in generative power. Since we do not yet have sufficient motivation to choose a particular formalism for linguistic analysis, it is not expedient to spend much time on the study of one of the versions. We therefore sum up here some observations, and leave to further work the elaboration of these points.

- For all MC- type TAG variants, we have the weak and strong inclusion of TAG, simply because a TAG grammar is always also a valid grammar in the other formalisms.
- The MC-TAG-DL formalism, as defined above, uses the non-local definition of adjunction, and, as argued above, does handle scrambling. It can easily

be seen that non-local-MC-TAG (without dominance links) can also generate all indexed strings from L_4. However, it is unclear whether the formalism is powerful enough to avoid "scrambling" of NPs into relative clauses attached to other clauses' arguments. From a linguistic point of view, such movement is impossible. Because of the unbounded number of such relative clauses, and the unbounded number of arguments that could be scrambled into them, it is not possible to pursue a solution involving features.

Mild Context Sensitivity

In [BJR91], we claimed:

> Weir [Wei88] has also shown that MC-TAGs are equivalent to the Linear Context Free Rewriting Systems (LCFRS) which are the best known characterization of the MCSG [Mildly Context-Sensitive Grammar] formalism. It is obvious that this version of MC-TAG with dominance constraints can produce all the possible versions of scrambled embedded sentences for any level of embedding. Further, it can be shown that an MC-TAG with such dominance constraints is no more powerful than an MC-TAG without dominance constraints. Hence, we conclude that scrambling is captured by an MCSG formalism.

The question whether scrambling is or is not captured by an MCSG is important because it assures us that the more powerful formalism can still be parsed in polynomial time.

It is clear from Weir's definition of LCFRS that non-local MC-TAG is not an LCFRS, because there is no way of representing its derivation tree as a local set. However, the fact that a formalism is not an LCFRS does not imply anything, since none of Weir's results are "if and only if" statements. There even seems to be a modified version of non-local MC-TAG-DL that can still be parsed in polynomial time (see [Ram92]).

Even though non-local MC-TAG are not an LCFRS, they still derive only semilinear languages and have the constant-growth property since in every derivation step a fixed number of terminals (one in a lexicalized grammar) are added to the string.

Although the data from LDS show no clear boundary on the amount of scrambling that is permissible in German, there is a sharp decline in acceptability when more then two or three elements scramble. Now, if for practical purposes a fixed boundary is set then the 'set-local' definition of adjunction in MC-TAG-DL is sufficient for describing such a constrained LDS phenomenon. The obvious advantage is the immediate availability of a polynomial time parsing algorithm.

Chapter 5

Relating MC-TAG-DL with a Unification-Based Formalism

In this chapter[1], I compare the proposed TAG-based analyses of the LDS phenomenon of the last chapter with two different analyses in Head-Driven Phrase-Structure Grammars (HPSG, see [PS87, PS91]). HPSG is a so-called unification-based grammar formalism. Unification, the basic operation in HPSG, is used to derive the phrase-structure of sentences from a set of so-called immediate dominance schemata which can be seen as (schemata for) context-free rules[2]. So unlike TAGs (and MC-TAG-DL), HPSG does not have the extended domain of locality in its basic phrase-structure deriving operations. Instead, special features (such as SUBCAT) are used to reflect dependencies. It is interesting to see, how the mechanisms which could be used in HPSG for handling the free word order of the LDS phenomenon parallel the mechanisms in a TAG-based framework, as proposed in the previous chapter.

In [Kas92], Kasper proposes to compile an HPSG grammar into a lexicalized TAG grammar. The possible benefits of such a construction for the TAG framework would be an automatic construction of TAG tree families from the information encoded in an HPSG grammar. The benefit for the HPSG framework might be realized in an efficiencent parser based on the resulting TAG grammar.

As section 5.4 shows, it turns out that the work of [Kas92] can be extended to compile the HPSG mechanisms for handling unbounded dependencies such as the LDS phenomenon into an MC-TAG-DL grammar. Clearly, a regular TAG can not be

[1]This chapter is partly based on [BR91], a joint paper with O. Rambow.

[2]This characterization of HPSG is not completely accurate as the ID schemata do not specify the number of components (this information is given in the lexicon) and in some variants the order of constituents is not given in the ID schemata. HPSG also resembles unification-based variants of categorial grammar.

sufficient as the target formalism. It is important to note that in this compilation, both extension to the TAG formalism, i.e. the multi-component aspect as well as the dominance links are crucial to accomplish the compilation.

5.1 Possible Analyses in HPSG

Within the framework of HPSG, two different analyses have been proposed for unbounded dependency constructions[3] (udc) such as the LDS phenomenon, based on the "interleave constituents" function (discussed in the following section) and on the NONLOCAL feature (discussed in section 5.1.2). Interestingly, both analyses correspond to the two different approaches which we have explored within the TAG framework (see Sections 4.1 and 4.2).

5.1.1 Interleave Constituents

In formalisms that state linear precedence (LP) rules separately from structural rules, such as HPSG, an obvious approach to scrambling is to define less constraining LP rules, while maintaining the structural description. However, in HPSG, LP rules cannot relate nodes that are not sisters. Thus, only clause-internal scrambling among objects can be handled with LP rules, while scrambling across the subject cannot.

Within the HPSG framework, a proposal has been made to allow for the effects of word-order variations that go beyond the domain of locality of a context-free rule, such as scrambling: the "interleave constituents" function [PS87, p.189]. This function breaks up and rearranges the PHONOLOGY values of daughter signs and returns the PHONOLOGY value for the parent sign. However, in our investigation of scrambling in the context of the TAG formalism, a problem for LP-based approaches became apparent: German, like most scrambling languages, is not completely unconstrained in its word-order freedom. In particular, there are islands for scrambling, into which or from which it is not possible to move anything. In German, such islands include NPs, relative clauses and finite clauses.

The problem that this poses for LP-based approaches to scrambling is that this is really a structural constraint, rather than a constraint on the order of strings. As the function "interleave-constituents" operates on strings, the PHONOLOGY feature needs to be augmented with pointers into the structural representation. Reape has

[3] Unboundend dependency constructions include all phenomena where one element can appear at an unbounded distance from a dependent position in the sentence, e.g. the wh-extraction. Naturally, the LDS phenomenon is an instance of an unbounded dependency construction.

worked out such an approach [Rea91] which also seems to be able to give an analysis of scrambling in German. Since, in this thesis, the MC-TAG-DL approach is proposed as the best way to handle the LDS phenomenon, I will not give a detailed discussion of this approach and its similarity to an LP-based approach in the TAG framework (FO-TAGs, see section 4.1 and also [Ram91]).

5.1.2 The NONLOCAL Feature

The NONLOCAL feature has been adopted within the HPSG framework as a mechanism to handle unbounded dependency constructions (such as topicalization in English) structurally [PS91, Ch.4]. A constituent can be realized by a trace, and the NONLOCAL feature contains the information about which constituents have not been realized. This has the effect of "moving" constituents. For topicalization, this information is kept in the NONLOCAL | SLASH feature.

It has been proposed that the value of the SLASH feature can be a set. This proposal predicts that doubly unbounded dependencies (such as long-distance scrambling in German) should indeed be possible in natural language.

5.2 An HPSG Analysis of Scrambling Using the NONLOCAL Feature

In this section, we will discuss analyses of scrambling in HPSG based on the NONLOCAL features. We will first present a "naive" approach based on the SLASH feature, and then argue for a more complex approach based on a new NONLOCAL feature.

Figure 5.1 shows a possible analysis of sentence (1b)[4]. Two traces are used to link the "moved" NPs to the verbs whose arguments they are. These traces are lexical items as described in chapter 4 of [PS91]. Case assignment and semantic facts are handled through structure sharing. VP_1 inherits the value of feature INHER | SLASH from its trace daughter, in accordance with the Nonlocal Feature Principle (NFP):

Nonlocal Feature Principle (NFP): For each nonlocal feature, the INHERITED value on the mother is the union of the INHERITED values on the daughters minus the TO-BIND value on the head daughter.

VP_2 in turn inherits the INHER | SLASH values from both of its daughters, VP_1 and the second trace. The new value of INHER | SLASH is the set containing the two

[4] We follow the standard HPSG analysis that German non-finite clauses, lacking overt subjects, are rooted in VP.

indices. Note that the order of introduction of the indices is in fact irrelevant since the feature is set-valued. The INHER | SLASH feature is inherited by VP_3, and in turn by S_1. Here, we use an analogue of Schema 6 [PS91,Ch.4,p.9] to introduce one of the indices into the TO-BIND | SLASH feature.

Schema 6:

X	$\rightarrow$	$\boxed{1}$	,	S[fin, INHER|SLASH {$\boxed{1}$, ...}, TO-BIND|SLASH {$\boxed{1}$}]
		Filler		Head

This leads to the discharging of the index from the INHER | SLASH feature in S_2 by the NP node *den Mittelstand* in accordance with the NFP. The remaining index is similarly discharged at S_3 by NP *den Ländern.* Note that the two traces obey the Trace Principle, since they are subcategorized for by a substantive head (in fact, strictly so):

Trace Principle: Every trace must be subcategorized by a substantive head.

Furthermore, the SLASH Inheritance Principle is obeyed, since *entschlossen ist* subcategorizes (strictly) for VP_2, *zu helfen* for VP_1 and NP_2, and *zu fördern* for NP_1:

SLASH **Inheritance Principle:** Every member of the INHER|SLASH set on a headed constituent must be inherited from (i.e. belong to the INHER|SLASH set of) a daughter which is either (a) (strictly) subcategorized by a substantive head, or (b) the head.

In order to obtain the correct phrase structure, we must introduce a new rule for the licensing of filler–head structures. The rule proposed for English topicalization (Schema 6) is insufficient since it assumes that the filler will be generated at the root of an S node headed by a tensed verb. While this is true in sentence (1a), it is not true in general as sentence (1b) shows:

(1a) ...daß der Minister [den Mittelstand]$_j$ den Ländern t_j
...that the minister medium businesses the states
zu fördern zu helfen entschlossen ist.
to foster to help is decided

(1b) ...daß [den Ländern]$_i$ [den Mittelstand]$_j$ der Minister t_i t_j
...that the states medium businesses the minister
zu fördern zu helfen entschlossen ist.
to foster to help is decided
...that the minister is decided to help the states to foster medium businesses.

Here, the scrambled NP *den Mittelstand* must be dominated by a VP headed by a

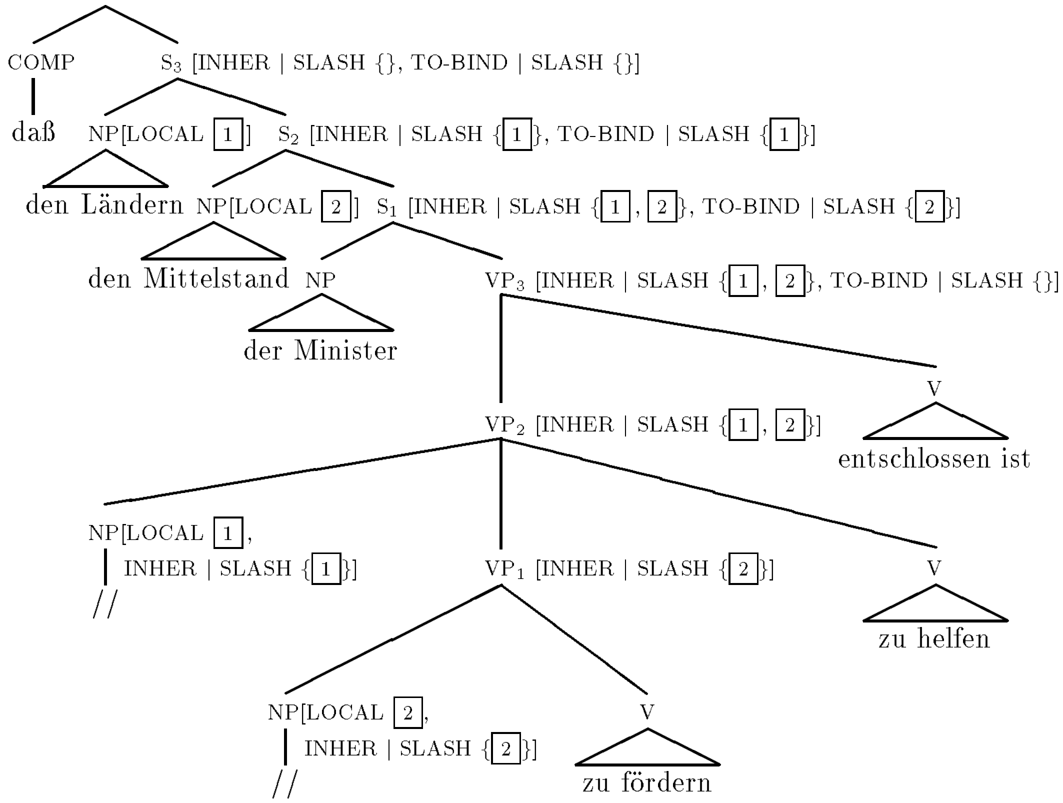

Figure 5.1: Analysis of sentence (1b).

non-finite verb[5].

Furthermore, since scrambling out of tensed clauses is impossible, we need a rule that embedded S nodes must have an empty SLASH feature, such as (1) below. (Since non-finite clauses in German never have overt subjects, they do not have an S node in HPSG analyses.)

$$(1)\quad \begin{bmatrix} \text{DTRS} & \begin{bmatrix} \text{LOCAL} & \begin{bmatrix} \text{CAT} & \begin{bmatrix} \text{HEAD} & \text{verb} \\ \text{SUBCAT} & <> \end{bmatrix} \end{bmatrix} \\ \text{NONLOCAL} \mid \text{INHER} \mid \text{SLASH} & \{\,\} \end{bmatrix} \end{bmatrix}$$

So far, we have assumed that both topicalization and scrambling are handled by the SLASH feature. Under this assumption, the rule above will predict that topicalization is not possible out of tensed clauses. This is indeed correct in standard German; however, in some Southern German dialects, English-style long-distance topicalization out of tensed clauses is possible. Thus, in order to handle both scrambling and

[5]Alternatively, we could assume that NP *der Minister* has also scrambled. However, there seems to be no linguistic motivation for requiring scrambled constituents to be the daughters of nodes headed by finite verbs, especially at the cost of string-vacuous scrambling.

topicalization with the SLASH feature, (1) must be relaxed to allow one element in the SLASH feature (namely, the topicalized one) to be passed out of tensed clauses. However, now it is difficult to exclude sentences such as (2), in which one topicalized element has passed the boundary of the tensed clause but is then discharged in the middle of a higher clause (as is licit for scrambled constituents), rather than in sentence-initial position (as is mandatory for topicalized constituents). The problem is that the discharge rules operate indiscriminately for any element of the SLASH set.

(2) *Der Hans hat dem Peter [das Buch]$_i$ zu sagen,
Hans has Peter the book to say
daß er mir t_i geben wird, versprochen.
that he me give will promised
Intended meaning:
Hans has promised Peter to say that he will give me the book

A second, more subtle problem arises when we use the SLASH feature for both scrambling and topicalization. This problem is related to the binding properties of the dislocated constituents. In topicalization and wh-movement constructions, it is well known that the filler often behaves as if it were still in the position of the gap. On the other hand, passivization and subject raising, which in GB have been given movement analyses ("NP-movement"), do not display such binding behavior. The question arises as to the binding properties of scrambled constituents. The issue has recently received much attention within the GB literature. While originally it was assumed that scrambling patterns with wh-movement [Sai89], more recent work argues that scrambling shows characteristics of both wh- and NP-movement [Mah89, Web89], or is in fact a unitary NP-movement phenomenon [LS91]. While the details of these arguments need not concern us here, let us consider some of the relevant data (from [San90]; her (29) and (30))[6].

(3) Heute habe ich versprochen, die Gäste einander vorzustellen
today have I promised the guests each other to introduce
Today, I have promised to introduce the guests to each other

(4) *Heute habe ich einander$_i$ versprochen, die Gäste t$_i$ vorzustellen.

(5) Einander$_i$ habe ich versprochen, die Gäste t$_i$ vorzustellen.

If scrambling and topicalization were the same type of movement, the contrast between (4) and (5) would be unexpected. Maintaining a record of both scrambling and

[6]We use extraposition only to mark long-distance scrambling into the matrix clause. It does not affect our analysis.

topicalization in the same feature (SLASH) would make it impossible to distinguish between them for the sake of binding. We conclude that separate structures are in fact necessary.

One option would be to introduce a new NONLOCAL | INHER feature, say, SCRAMBLE, and then replicate the analysis shown in Figure 1 above using this new feature. The feature SCRAMBLE would have to be set-valued in order to account for the doubly unbounded nature of scrambling; the SLASH feature could now be a bounded cache, since only one element can undergo topicalization.

However, sentence (4) above shows that the analysis given for scrambling by the mechanism described above, be it in the SLASH feature or a new SCRAMBLE feature is problematic: according to HPSG Binding Theory (as formulated in [PS91]), there is no way to avoid that the anaphor *einander* is bound by *die Gäste* (which must be avoided in sentence 4 above), no matter where it has been dislocated to. While one solution may be to extend HPSG binding theory in some way, such an extension would likely violate the spirit of HPSG binding theory by introducing potentially rather complex structural notions. Moreover, in HPSG other instances of what are in GB classified as NP-movement, namely passive-movement and raising, are not handled by SLASH-style features at all. Instead, lexical rules transform the SUBCAT lists of the verbs involved in the constructions. We will therefore sketch a similar approach for the case of scrambling: in sentence (4) above, argument *einander* actually needs to be removed from the SUBCAT list of *vorstellen* if it long-distance scrambles, since it can no longer participate in binding with any of the verb's other arguments. However, scrambling does not merely destroy binding possibilities, as is shown in (6):

(6) ?Heute habe ich die Gäste$_j$ einander$_i$ versprochen t$_j$ t$_i$ vorzustellen

(6) shows that scrambling does not prevent constituents from entering into binding relationships: when both constituents from the lower clause undergo scrambling into the same higher clause, binding between them is preserved. Thus, we propose to analyze scrambling by moving arguments from SUBCAT lists to the SUBCAT lists of higher verbs, using a set-valued SCRAMBLE feature. A lexical rule transfers arguments from the SUBCAT list to the SCRAMBLE feature. The discharge schema (originally Schema 6) is replaced by a rule which transfers elements from the SCRAMBLE feature to the SUBCAT list of a verb.

Note that this analysis in some respect resembles the familiar clause-union analysis of scrambling [Eve75, Net91]. However, we do not in fact form a single verb complex with a single unified SUBCAT list, since in such an analysis it would be difficult to account for the shifting binding possibilities exhibited in the sentences above within the framework of HPSG binding theory[7].

5.3 Correlation to the MC-TAG-DL Analysis

Although in general the formal power of the formal system underlying HPSG is unconstrained, showing parallels to constrained grammar formalisms, such as TAG and its variants, gives some insight to the formal complexitiy of one particular aspect of HPSG, namely how it handles unbounded dependency constructions. In this section, we will attempt to relate the SLASH-feature analyses of scrambling with the analysis in the TAG framework that we have proposed in chapter 4.

Recall, that in MC-TAG-DL, the elementary structures are tree sequences. The trees of one sequence can be linked by dominance constraints. The elements of a tree sequence can be adjoined at different nodes of a target tree, provided that the dominance constraints are met and all trees of the sequence are adjoined simultaneously.

This is very similar to the SLASH-feature approach in HPSG:

- Both approaches make direct use of the phrase structure tree to account for possible positions of the moved argument and for possible word orderings.
- The Nonlocal Feature Principle of HPSG implicitly forces a moved constituent to c-command its trace. The structure of the tree sequences in our MC-TAG-DL analysis of long distance scrambling implies exactly the same constraint.

An interesting question arises as to the structure of the SLASH feature, which in fact parallels the question about the "local" vs. "non-local" distinction within MC-TAG-DL. Recall that an MC-TAG-DL is called "set-local" if the trees of one sequence are adjoined into (perhaps different) trees of exactly one other elementary sequence. In a non-local MC-TAG-DL, adjunction is allowed into trees from different elementary sequences. The difference is important because set-local MC-TAG-DL are polynomially parsable, while this is an open question for non-local MC-TAG-DL. It is not clear, however, whether set-local MC-TAG-DLs are descriptively adequate; if long-distance scrambling is in fact unconstrained as described above, they are not.

It has been shown that TAGs are equivalent to Linear Indexed Grammars (LIGs), which is a context-free grammar with a stack valued feature that can be shared

[7]Of course, it would be possible to account for the binding facts by altering the relative obliqueness of arguments as they scramble. However, this approach seems linguistically ill motivated and ad hoc.

between the parent node and exactly one daughter. SLASH features clearly resemble the stack of an LIG. The exact definition of the structure of the feature will affect the formal power of the system. With a stack-valued SLASH feature, the resulting system is formally equivalent to a TAG, which has been shown in chapter 3 to be descriptively inadequate for long-distance scrambling.

In [Ram92], Owen Rambow investigates varieties of LIGs with the aim of defining LIG variants equivalent to the two types of MC-TAG-DL. A context-free formalism with a set-valued feature such as SLASH and non-local MC-TAG-DL seem to be equivalent. Even though the formal implications of a set-valued SLASH feature are currently not well understood, the data on long-distance scrambling shows that a set-valued feature structure is indeed needed in HPSG.

5.4 Compiling HPSG to MC-TAG-DL

In this section, I will further demonstrate how the relevant fragment of HPSG (the principles and schemata dealing with unbounded dependencies) can be compiled into the MC-TAG-DL formalism. It turns out that both extensions, i.e. tree sequences and dominance links are essential in this construction.

To keep matters simple, I will assume an implementation of the LDS phenomenon in HPSG by a SCRAMBLE feature (as discussed above) which is constraint by a SCRAMBLE inheritance principle that is identical to the SLASH inheritance principle and that there is a version of schema 6 for the SCRAMBLE feature. Clearly, this allows us to give an HPSG analysis of the LDS phenomenon. In figure 5.1 only the SLASH features need to be renamed to SCRAMBLE.

The compilation process keeps two different data structures. One is a partial HPSG structure which is derived by the application of schemata and principals. The second is the actual result of the compilation process, i.e. an elementary tree sequence of an MC-TAG-DL.

A compilation process would begin with lexical entries and then repeatedly apply schemata and principles in order to derive elementary trees, or rather tree sequences. If such a process was started with the lexical entry for "fördern" (to foster) then applying Schema 2 could result in the structure shown in figure 5.2.

Schema 2: [SUBCAT $<\boxed{1}>$] $\rightarrow$ HEAD[word], C^*
I.e. if the mother node is an almost-saturated phrase (SUBCAT list with one element) then the head daughter is a lexical sign (word) and there are additional complements (C^*) for the other element on the SUBCAT list of the head daughter.

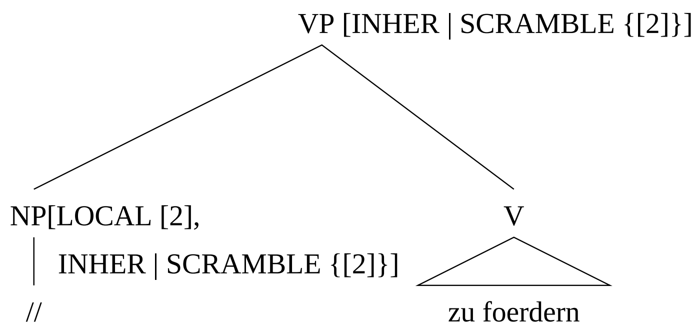

Figure 5.2: Applying Schema 2 to "fördern" (to foster).

The results of other principles which also apply, need not concern us since they will merely affect the feature structures and not the phrase structure. So far, the complete HPSG structure is copied as one tree t_1 of the resulting elementary tree sequence.

At this point, one of the schemata which can apply is again Schema 2, this time using the structure of figure 5.2 as a complement. The SCRAMBLE inheritance principle will ensure that the SCRAMBLE value of VP_1 is inherited by VP_2. From this point on, Schema 2 may apply repeatedly, using the structure constructed so far as a complement. An actual compilation algorithm would need to keep control of this recursion.

The compilation process can detect this application of Schema 2 with the previous structure as a complement and will begin to create a new tree (which is empty so far) in the elementary tree sequence, which will dominate the tree t_1 that has been constructed so far. Recursive applications of Schema 2 in the above mentioned manner will not result in any additions to the compilation result. The motivation for ignoring the additional HPSG structure is the design of the MC-TAG-DL framework. It is supposed to capture recursive structures within one elementary structure and clearly, the adding of structure by applying Schema 2 is a recursive step which is outside of the domain of locality of the verb "fördern" and thus not included into the resulting elementary tree sequence. However, the HPSG structure constructed so far is not saturated and the process can therefore not stop.

At any point the scrambled NP of "fördern" can also be discharged from the SCRAMBLE feature, resulting in a structure as shown partially in figure 5.3, see also figure 5.1.

At this point, the mother node is saturated and the compilation process can stop. All information which is contributed by structures that have not been added to the elementary tree sequence (e.g. additional elements in the SCRAMBLE feature, e.g. see node S_3 of figure 5.1) can be deleted. The local tree at which the discharge has taken place (shown in figure 5.3) should be added to the resulting elementary tree sequence

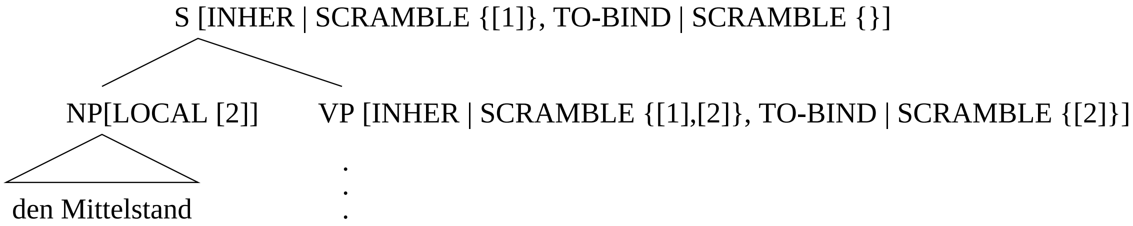

Figure 5.3: Discharging the NP from the SCRAMBLE feature.

as a second tree t_2. However, the tree t_2 is not an auxiliary tree since its root node bears an S and its foot node bears an VP. There are several possible ways to solve this:

- One might argue that this is due to the particular HPSG analysis and change the HPSG structure in a way to make t_2 become an auxiliary tree, e.g. by adding another S node immediately above the VP node. Then t_2 could include only the new S node as its footnode. But then this S node would have to be added to *all* the elementary tree sequences which are compiled from the structure that includes the VP node. This is clearly not feasible since one of the main goals of the compilation process is to keep the HPSG analysis.

- As the extended formalism also includes the substitution operation, one can simply mark the VP node as a substitution node. In this way the tree t_2 of the compiled tree sequence can combine with VP nodes of other tree sequences to create exactly the structure of the HPSG analysis.

- A third possiblity is the inclusion of node labels into the feature structures of the resulting MC-TAG-DL, in the sense that all nodes are labeled with the same dummy label. This ensures that all trees are auxiliary trees and also allows to exactly reflect the HPSG analysis in the resulting MC-TAG-DL grammar[8].

In any case, a dominance link connects t_2 to t_1. The resulting elementary tree sequence is shown in figure 5.4.

This is exactly the type of tree sequence which was used in section 4.2 to capture the LDS phenomena. Verbs with two objects would result in tree sequences with three elements, one additional tree would reflect the discharge of the second object.

Both, the multi-component aspect and the dominance links are important to the compilation process. First, the tree t_2 does not immediately dominate t_1 in the HPSG analysis. This can only be reflected in the resulting elementary tree sequence

[8]This idea is due to K. Vijay-Shanker, pers. comm.

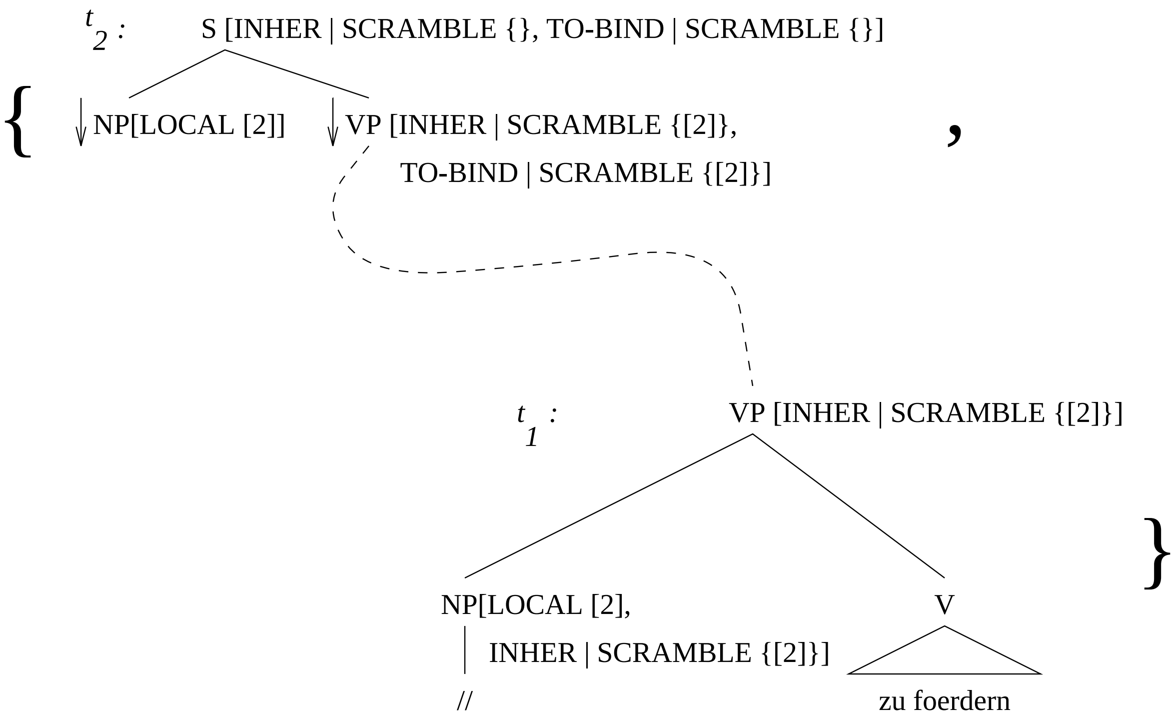

Figure 5.4: Resulting elementary tree sequence with two trees.

by keeping t_1 and t_2 as seperate trees. Keeping them as one tree would imply that an LDS analysis with regular TAGs is possible. Second, the tree t_2 of the resulting elementary tree sequence appears in the HPSG analysis only in a position where it dominates t_1 which is reflected by the dominance link between t_1 and t_2.

Chapter 6

HyTAG: Compact representation of MC-TAG-DL with Metarules

In this chapter, metarules are introduced as a method to represent a given MC-TAG-DL or TAG grammar in a compact and efficient way. Metarules allow to factor out redundancies which appear in natural language grammars.

Recall that the main building blocks of a TAG are elementary trees. In a lexicalized TAG, there is one lexical item associated with each tree and for each lexical item there will be a number of different trees, representing the different types of structures (e.g. sentences) a lexical item can appear in. Between these trees there are a number of systematic similarities. These similarities need to be expressed in some way. It turns out that these similarities express linguistic relations between phrases within one elementary structure – an advantage of the extended domain of locality of TAGs (and MC-TAG-DL).

In some linguistic theories, these similarities have been expressed either as transformations which apply to intermediate, generated structures (i.e. in Transformational Grammar) or they have been described as transformations on the elementary structures of the grammar (i.e. the lexical rules of HPSG or the meta-rules of GPSG). Although the similarities are sometimes relational in nature, they have usually been formulated as (directed) rules. In this chapter, I will develop and discuss such rules (which I will also call metarules) for the TAG framework and especially for the MC-TAG-DL formalism.

There are also relations between elementary trees which are hierarchical in nature, so some hierarchical representation language (of a similar nature as those used in object-oriented languages) is necessary. A language for hierarchical descriptions within the MC-TAG-DL formalism will be described in chapter 7.

As chapter 3 has shown, an augmented version of TAGs is needed for a number of phenomena in free word order languages. However, for a description of more fixed word order languages as English, the original definition of TAGs may well turn out to be sufficient. Also, the definitions for TAGs are somewhat simpler than those for MC-TAG-DL. Therefore, I will introduce both metarules and the inheritance network first only for the TAG formalism. Based on this, the necessary adaptions for the MC-TAG-DL formalism are quite easy to formulate and follow immediately.

6.1 Metarules for regular TAGs

In order to express the relations between the various forms which a phrase headed by some word can have, we introduce a variant of lexical rules for a TAG-based grammar. We will call them metarules (like the metarules in GPSG) and define them first as an extension to the TAG formalism (in section 6.1.5) and finally extend the definitions to the MC-TAG-DL formalism (in section 6.2).

Metarules will be used for two purposes. As a practical benefit, they allow for a significantly shorter description of grammars for natural languages. On the other hand they can capture a number of linguistic generalizations (i.e. lexical rules) that cannot be expressed in the original TAG framework.

Formally, metarules are introduced as a set of rules that transform a given initial set of elementary trees into an expanded (but still finite[1]) set of elementary trees. Thus they can be looked at completely independently from processing considerations (either generation or parsing). However, the existence of this rules can be exploited by practical systems as well as by linguistic and cognitive theories.

6.1.1 Motivation and Introduction

Two current Lexicalized Tree Adjoining Grammars for English (the LTAG project, see [ABCS90]) and French (see [Abe88]) use the concept of *tree-families* to group together all the elementary trees "that share the same subcategorization type" [ABCS90]. For example (see also Figure 6.1), all variants of clauses (or more general: phrases) that are headed by a verb that subcategorizes for two noun phrases such as *eat* are grouped

[1] If such rules can produce infinitely many new elementary structures then the formal properties (e.g. the generative power) of the formalism can change. This is discussed in some detail in section 6.1.3.

together in a tree-family that is called *Tn0Vn1*[2]. This includes variations like, e.g., Wh-questions, relative clauses, topicalized and passive sentences.

The same variations of the basic declarative sentence (which in different formalisms and linguistic theories are called lexical, syntactic or transformational rules respectively) occur in almost all other tree-families as well. In the current implementations of the grammars, these trees are explicitly included in all tree-families. This not only increases the number of trees significantly but also – and more importantly – misses a linguistic generalization: the existence of (lexical and syntactic) rules that can be used to describe such variations as passive, topicalization etc. across the tree-families.

The term metarule stems from the fact that they are rules on a meta-level, i.e. they are not used to construct the (phrase-)structure but are applied to the phrase structure rules themselves. They were first introduced by Van Wijngaarden in the formal description of the programming language ALGOL 68 [vWMPK69]. Since then, a number of different formalisms use metarules for capturing generalizations.

It depends on the kind of basic formalism, what kind of objects the metarules apply to. Usually, metarules are stated as rules about the phrase generating entities, these are rules in simple CFGs, in a way the lexical entries in HPSG and would be the elementary trees in a TAGs. In linguistic frameworks their first appearance was in GPSG [GKPS85], where they were strongly influenced by the transformational rules of early transformational grammar which date back to [Cho57]. Transformational rules are applied to intermediate, but in a sense complete structures, they apply *after* the completion of a first phrase structure construction process. Metarules, on the other hand apply to the phrase structure generating entities, and thus apply *before* any phrase structure is derived.

Metarules can be defined as a set of rules that take grammar-rules[3] as input and generate new rules as output. A metarule consists of an *input-pattern* and an *output-pattern*. The input-pattern may contain so-called *meta-variables* which describe those parts of a grammar-rule which are not affected if the metarule is applied. If a grammar-rule (or an elementary tree) matches the input-pattern, then the application of the metarule creates a new grammar-rule (or elementary tree) which could then be added to the grammar. The new rule is created by filling the metavariables in the output-pattern with the parts of the original grammar-rule that are bound to the same variable in the input-pattern.

[2] *T* stands for *tree familiy*, *n0* and *n1* stand for *noun phrase* and *V* stands for *verb*; *V* is capitalized because it is the so-called *anchor* of the tree.

[3] For simplicity the term *rules* is used here with a very general meaning. It stands for the elements of the grammar that generate the sentences. In a CFG these are the CF rules, in a GPSG the ID-rules and in a TAG the elementary trees.

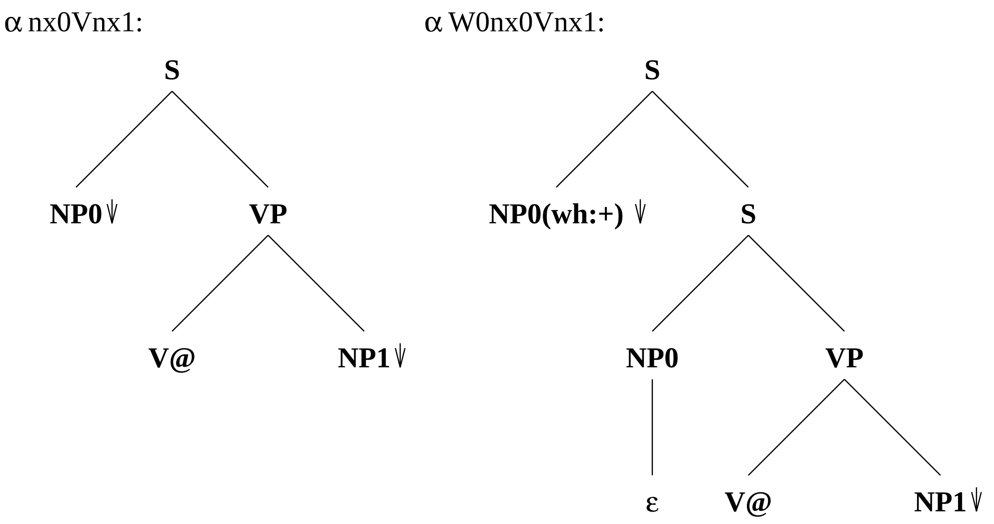

Figure 6.1: Two trees from the tree-family Tn0Vn1. Both are selected for by verbs like *eat*. αnx0Vnx1 generates sentences like "Peter eats a cookie."; αW0nx0Vnx1 generates sentences like "Who eats the cake?".

6.1.2 An Example

An example of a possible metarule in the LTAG for English is used to clarify the concept of metarules in general as well as specifically for TAGs. Almost all tree-families in the LTAG for English include a tree for a wh-question on the subject of the sentence. Figure 6.1 shows the basic tree and the wh-tree for the tree-family Tn0Vn1 which is selected for by verbs like *eat*. The tree αn0Vn1 can be used to generate a sentence like *"John eats the cake."* and the tree αW0n0Vn1 can be used to generate *"Who eats the cake?"*.

Another tree-family that includes a tree for wh-questions (on the subject) is αn0V that is selected for by a verb like *sleep*. Figure 6.2 shows the corresponding trees for this tree-family.

Although the trees αW0n0Vn1 and αW0n0V are not identical, they share the same differences to the corresponding basic tree in their tree-family. Both have an additional S-node that is the root and both have a left branch that has an NP-node with the feature [wh: +]. Both have the basic tree as a right branch from the new root node which has the subject NP-node changed from a substitution node to a node with the empty word as its leaf. This generalization holds for almost all other tree-families in the LTAG for English. To express it, a metarule can be used. Without giving a formal definition at this point (see section 6.1.5), a metarule for wh-movement of the subject is given in figure 6.3.

By using metarules in this style, the number of trees that have to be stated in an LTAG

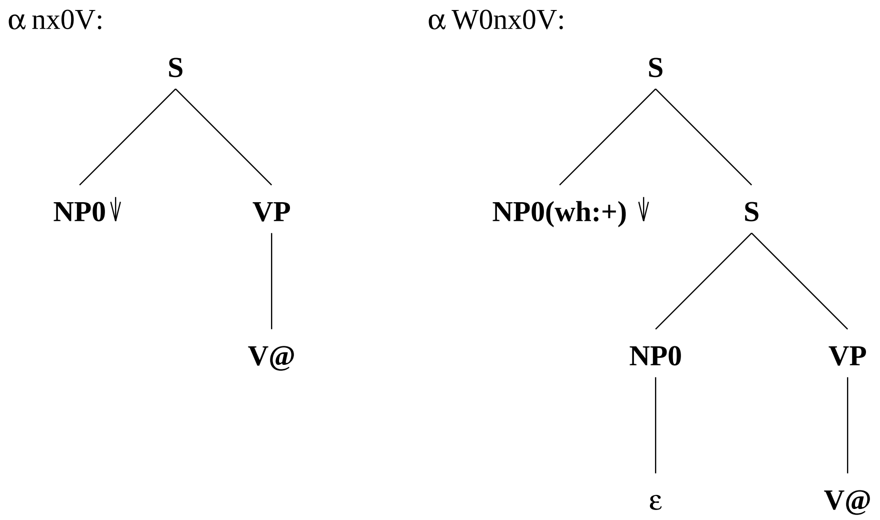

Figure 6.2: Two trees from the tree-family Tn0V. Both trees are selected for by verbs like *sleep*. αnx0V might generate "Mary sleeps."; αW0nx0V generates sentences like "Who sleeps?".

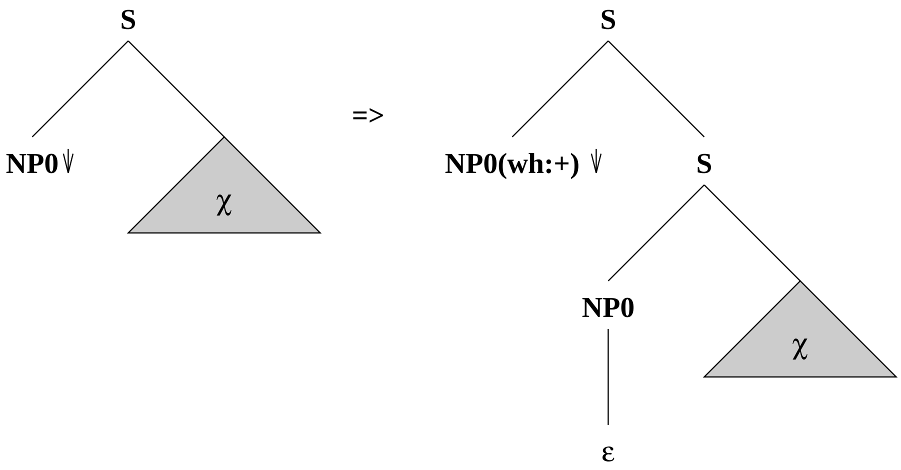

Figure 6.3: A metarule for Wh-movement of the subject.

can be reduced considerably. Ideally, for every tree-family only one representative tree, which could be called the *basic* tree, has to be given, all the other trees can be derived by the application of metarules.

6.1.3 Problems: The generative power of metarules

The introduction of metarules into a grammar formalism brings a number of problems with it. The most important one is the generative power of the extended formalism. In a wide number of different formalisms such as transformational grammar, Van Wijngaarden grammars, ID/LP grammars, Categorial Grammars and Head grammars the formalism that includes recursive metarules has a greater generative power than the original formalism. In fact the extended formalisms can generate every recursively enumerable language which makes the word-recognition problem undecidable.

In all cases, the problem arises from the fact that recursive application of metarules can generate infinitely many grammar-rules (in the same way that the grammar-rules generate infinitely many sentences). This will therefore be the key idea to bound the expressive power of a TAG formalism that is extended with metarules.

A way around this problem (without giving up the concept of metarules entirely) is to simply dismiss recursive applications of metarules. This has been proposed in GPSG [GKPS85] (and has been further refined in RGPSG [Ris87]). In [GKPS85] the application of metarules on the ID rules of the grammar is defined as being restricted to at most one application of each metarule to a given ID rule of the grammar. More precisely, in the process of deriving a new ID rule from an initial ID rule by applying a metarule and (repeatedly) applying another metarule to the result, each metarule may be applied only once. This restriction that is called the *finite closure* has the advantage that recursive applications of metarules are ruled out so that the metarules only generate a finite number of new rules and therefore the generative power of the grammar formalism is unchanged. However this restriction appears motivated only by the concern for the generative power of the formalism and no other (linguistic) motivation is given. Based on experience with GPSG grammars Ristad [Ris87] proposed the concept of *biclosure*, meaning that at most two metarules may be applied to an ID-rule. But at least for the LTAG for English, two is not sufficient. (E.g. a combination of dative movement, passive and topicalization.) Instead of picking an arbitrary finite bound on the number of applications, we will here give a principle based limit to the application of metarules (see Section 6.1.4 for details).

Probably the first usage of rules on a meta-level has been made in the W-Grammars of Van Wijngaarden [vWMPK69], the proof for the undecidability of the W-Grammars can be found in [CU77]. In [BJE86], an approach to use a slightly modified (but also undecidable) version of W-Grammars for natural language processing is developed.

In W-Grammars, the actual phrase structure grammar (much like a CFG, but with potentially infinitely many rules) is first *generated* by the meta-level grammar (called hyper-rules). Another approach is the usage of meta-rules to *extend* a given set of phrase structure rules (as done in GPSG).

But the same undecidability result holds for meta-rules in a CFG (and therefore for metarules in GPSG), see [UP83]. Even the restriction to just one meta-variable does not change the result.

In [Bec89], I have introduced a definition of recursive metarules for TAGs which does not increase the generative power of the resulting formalism. However, this result depends on a restricted form of tree patterns in the metarules. The descriptive power of this restricted formalism is weak and not completely satisfying for linguistic purposes.

6.1.4 Bounding metarules by principles

The new definition of metarules for TAGs developed in this chapter is constrained by two desiderata. One is the need for an adequate description of a certain set of linguistic phenomena. The other is a desire for bounding the recursive application of metarules in order to retain the restricted expressive power of the TAG formalism. These restrictions will be based on two (very closely connected) principles of TAGs.

In the framework of TAGs a restriction can be given which is motivated by principles of TAG as well as by linguistic considerations. The resulting restriction is similar to the finite closure restriction (see section 6.1.3), only better motivated.

(1) One of the features of TAGs is that they *factor the recursion* in the described language in a very compact way, namely by adjunction of auxiliary trees. So any recursion in the language (also in a natural language) should be expressed in terms of auxiliary trees[4].

    So metarules should not be used for describing recursion; instead, auxiliary trees should be used[5].

(2) The factoring of recursion that TAGs provide leads to the notion of the *extended domain of locality*. This term describes the phenomenon that TAGs provide

[4]This does not contradict the fact that the definition of LTAGs includes also a second recursive operation, that of substitution. Substitution can always be expressed in terms of adjunction and simply allows for more adequate elementary trees.

[5]But like substitution, recursive metarules that can be transformed into auxiliary trees may turn out to be useful for the description of some phenomena.

an easy way of describing the predicate-argument structure of verbs in one elementary structure, namely the elementary trees. One of the principles in writing TAGs for natural languages in this way is to keep the elementary trees as close to such predicate-argument structures as possible (and describe the construction of more complex sentences by adjoining). These are exactly the considerations that lead to the notion of co-occurrence restrictions (see Section 3.2).

So the maximum structure that an elementary tree should describe is limited by the principles and basic features of TAGs that guide their construction.

This leads to a restriction on the application of metarules that can informally be stated as follows:

Restricted application of metarules: *The resulting tree from the application of a metarule to an elementary tree is itself a valid elementary tree if and only if it is smaller (in a yet to be defined measure) than a given limit of the grammar.*

This of course implies that the application of metarules will always be finite. But this version of finite closure is not merely stipulated but derived from principles of the formalism and its specific use for representing natural language syntax.

Not all elementary trees in an LTAG for a natural language will consist of only the predicate-argument structure. Exceptions are, e.g., the more complex trees that have been proposed for idioms [AS89]. But still these trees are limited in their size and an idiom that exhibits unbounded size – and hence recursion – should probably again be decomposed into auxiliary trees.

6.1.5 Definitions

The following definitions apply to a regular TAG (possibly with substitution nodes). Some minor changes have to be made to adapt metarules to an extension with features and to the FO-TAG formalism or to MC-TAG-DL. These are discussed in the next section.

Definition 6.1.1 (Metarule)

A **metarule** *for a given TAG G and a (finite) set MV of* metavariables *has the form*

$$t_1 \Rightarrow t_2$$

where t_1 and t_2 are elementary trees with the exception that any node may be a metavariable. Every metavariable that occurs in t_2 also occurs in t_1.

In this definition, the occurrence of metavariables could be further restricted. E.g. a metavariable could be restricted to exactly one occurrence in the input-pattern and at most one occurrence in the output-pattern. This would result in *linear* metarules and exclude copying of metavariables which seems to be a reasonable restriction. The only phenomenon where a need for copying metavariables in t_2 would arise is coordination. But as coordination is a phenomena beyond the domain of locality found in elementary trees, it can not and should not be handled by metarules. On the other hand *erasing* metarules (where metavariables that occur in t_1 do not occur in t_2 as well) cannot be ruled out, they are needed for rules like the passive-metarule (which can delete the subject NP of the input-tree).

However, the generative power of the formalism would not be changed by omitting such restrictions because the finite application of any kind of metarule can always be computed first and then the resulting set of elementary trees can be used as a pure TAG without metarules.

Before defining the application of a metarule first the notion of *substitution* that will be used needs to be defined. Note that a metavariable can match a set of trees.

Definition 6.1.2 (Substitution of a meta-variable)

The result of the substitution *of a meta-variable χ in a tree α with a set T of trees is constructed as follows:*

- *χ is deleted from α.*
- *All trees t_i in T are added to the mother node of χ as new daughters.*
- *If χ had daughters in alpha then every such daughter must be added as a daughter to some node of some tree t_i in T. The order of the daughters of χ has to be preserved.*

The metarule in figure 6.3 can be used to derive the tree αW0nx0V from αnx0V in figure 6.2. The metavariable χ matches the subtree under (and including) the VP node. In the output pattern of the metarule, χ is then substituted with this subtree, resulting in the tree αW0nx0V of figure 6.2.

Note that metavariables can be used to describe any contiguous part of a tree, especially a set of subtrees of the interior of a given tree. Note that therefore, the result of a substitution of a metavariable is not necessarily unique. If the metavariable has daughters then addition of these daughters to the substituted tree(s) will usually be possible in more than one place.

Convention 6.1.1 (Drawing metavariables)

In order to make the input- and output-pattern in metarules more intuitive and to distinguish the metavariables from other nodes, metavariables are not written (drawn) like other nodes. The arc that connects a mother-node with a daughter that is a metavariable is omitted and replaced by a triangle. The name of the metavariable is written within the triangle. Figure 6.4 shows how a metavariable χ is drawn according to this convention.

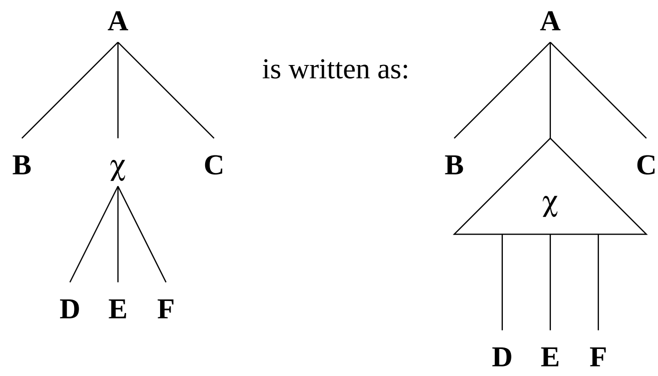

Figure 6.4: Convention for drawing metavariables.

Definition 6.1.3 (Application of a metarule)

Given a boundary M on the size of the output-trees of metarules[6]*, a metarule γ of the form $t_1 \Rightarrow t_2$ can be applied to an elementary tree α if and only if*

(i) There is a substitution σ of the metavariables in t_1 with (possibly empty) sets of trees such that $\sigma(t_1)$ and α become identical. The substituted sets of trees are said to be bound to the metavariables*. α is said to be* matched *by the input-pattern t_1.*

$\sigma(t_2)$ the result *of the application of the metarule is a tree β that is derived from t_2 by substituting all tree set variables with the tree sets that they are bound to. β is said to be* derived from α by the application of the metarule γ*. Note that the substitution and therefore the result of the application of the metarule is not unique.*

(ii) The resulting tree is smaller than the boundary M.

[6]E.g., given as a predicate on output-trees that evaluates to true if and only if the tree is smaller than the given limit.

The only constraint on the boundary M is that there are only finitely many trees which are smaller than M.

In a linguistic application of metarules at least some of the nodes in the output-pattern correspond to some node in the input-pattern. One might want to express this correspondence by some means (e.g. links or co-indexing) but formally this is not necessary. See subsection 6.1.6 for further discussion of this point.

Definition 6.1.4 (MR-TAG)

An MR-TAG *G is a quadruple (G', MV, MR, M) where G' is a TAG, MV is a finite set of* metavariables*, MR is a finite set of* metarules *and M is the boundary on the size of the output of metarules.*

We will now define the *tree set of an MR-TAG*, the set of all trees that can be derived by an MR-TAG and then from that define the *language of an MR-TAG*. The first step is the definition of all trees that can be derived by the application of metarules (the *set of metarule-derived elementary trees*).

Definition 6.1.5 (Metarule-derived elementary trees)

Given an MR-TAG G, the set of metarule-derived elementary trees *is the recursively defined set of all trees that can be derived by repeated application of metarules:*

- *All elementary trees of G are* metarule-derived elementary trees *(abbreviated as* MR-derived elementary trees*).*
- *All trees that are derived from MR-derived elementary trees by the application of a metarule are MR-derived elementary trees*[7]*.*

Now the tree set of an MR-TAG is defined analogous to the tree set of a TAG.

Definition 6.1.6 (Tree Set of a MR-TAG)

The tree set *T(G) of a MR-TAG G=(G', MV, MR, M) is the tree set of the TAG G" that is constructed from G' by substituting the elementary trees in G' by the set of MR-derived elementary trees of G.*

[7] This definition is recursive. Note, however, that the definition of *application of a metarule* rules out trees that are larger than the given boundary M. Therefore, the set of MR-derived elementary trees is always finite

One might want to allow the application of metarules also on derived trees (these are trees that are generated by adjoining and substitution). This could be achieved by adding the following to the original definition of T(G):

- *All trees that are derived from derived trees by the application of a metarule are also derived trees.*

However, even if such a definition is safe by means of the given boundary M, it violates the principle (ii) for the use of metarules which claims that metarules should only be used to generate additional elementary trees. Such a use of metarules would be similar to the transformational rules of transformational grammar.

Definition 6.1.7 (Language of a MR-TAG, MR-TAL)

The language L(G) *of an MR-TAG G is the set of all strings that are on the leaves of a tree of the tree set T(G):*

$$L(G) := \{w \mid w = \mathrm{leaf-word}(t),\ t \in T(G)\}$$

MR-TAL *is the set of all languages that can be generated by an MR-TAG:*

$$MR-TAL := \{L \mid L = L(G),\ G \text{ is } \mathrm{MR-TAG}\}$$

Adaption to FO-TAGs

Although chapter 4 concluded with the proposal to use the MC-TAG-DL formalism for an adequate description of natural language and noted some disadvantages of the FO-TAG formalism, a sufficiently refined FO-TAG based formalism might overcome these disadvantages. Therefore, I will briefly discuss in this section how the definitions for metarules can be adapted to the FO-TAG formalism.

In the FO-TAG formalism (see section 4.1), the input- and output-patterns of a metarule no longer can describe elementary trees. Instead, they have to describe elementary structures (namely a dominance structure and a set of linear precedence rules). So a pattern in a metarule then consists of a dominance structure with meta-variables that is matched in a way similar to the matching of TAG trees as defined in definition 6.1.3. In addition, a pattern must also describe the precedence rules. The patterns may contain precedence rules with variables for nodes that are handled in the same way that the metavariables for subtrees are. It seems convenient to define matching for metarules in FO-TAGs in the following way.

- A precedence rule matches a (precedence rule) pattern if and only if there is a substitution of the variables in the pattern so that the rule and the pattern (after substitution) are equal.

- The precedence rules of an elementary pattern match the precedence pattern of an input pattern if and only if for every precedence rule pattern there is a precedence rule that matches it. This way a pattern can easily under-specify the precedence rules of an elementary structure that is to match it. In addition to the precedence rules that are generated from the output pattern, all those precedence rules of the input elementary structure that are not matched by metarule will also be included in the resulting elementary structure.

- An elementary structure matches an input pattern if and only if the dominance structure matches the dominance pattern and the precedence rules match the precedence patterns.

The last condition is trivial and straightforward, but the second one is motivated by the expectation that no metarule may delete parts of the precedence rules. All precedence rules that are not matched by the metarule can be seen as being bound to an invisible, implicit variable that also appears in the output pattern. Other variables for precedence rules would only be needed to describe phenomena like copying rules (which has no meaning), deleting rules (which is expected not to be needed) or changing the order (which also has no meaning for precedence rules).

Adaption to feature-structures

In a TAG with feature-structures (see section 1.1.3), each node on an elementary tree may have one (or two in the approach of [VS87]) feature-structures attached to it. They may either be represented as DAGs or as sets of feature-equations (see [VS87]). For external representations (used e.g. for editing a tree) usually feature-equations are chosen. Also for the applicability of metarules features play an important role. (E.g. a metarule that generates the corresponding wh-question to an elementary tree may not be applied if the input-tree already has the feature [wh:+].) On the other hand not all feature-values are important. So also for feature-structures a description with patterns is necessary.

The syntax for feature-equations (given here in a BNF) is fairly simple:

- equation ::= path = path | path = value

- path ::= node-name feature*

So there have to be two kinds of additional meta-variables for matching feature-equations: Path-variables that match any (sub-) path and value-variables that match any value. This set of variables may be expanded further, since some complex (and probably not very useful) constraints can not be expressed with them (e.g. "The node has to have only equations which contain the feature agr in their path.").

So now the syntax for feature-equations can be expanded to a syntax for equation-patterns:

- equation-pattern ::= path-pattern = path-pattern | path-pattern = value-pattern
- path-pattern ::= node-name-pattern (feature + path-variable)$^{+}$
- value-pattern ::= value | value-variable

Matching of an equation and application of a metarule is then defined as in definitions 6.1.2, and 6.1.3.

6.1.6 Relating Input and Output of a Metarule

In [Abe91], Abeillé discusses the use of links between (the nodes of) trees that are related by syntactic or lexical rules. She proposes a variation of synchronous TAGs which she calls auto-synchronous TAGs to connect such related trees. When one expresses such syntactic or lexical rules explicitly with metarules then it might be useful to generate such links automatically with the metarules. Some minor additions to the definition of metarules will allow this easily:

- Allow links from a node in the input-pattern to a node in the output-pattern in the definition of a metarule.
- Define that the links are added to the result of the application of a metarule as links to the tree which the metarule was applied to.
- When a meta-variable was linked to some node, every root of the tree set that is substituted for the meta-variable gets a copy of the link.

Adding such links to the grammar has various benefits. For one thing, it gives a more explicit representation of the history of an elementary tree that is derived by metarules[8]. Such links can be useful e.g. for generating paraphrases which again could be important for machine translation with synchronous TAGs: The translation lexicon needs only to state the most preferable translation, other paraphrases can be deduced. Generating paraphrases is also useful in natural language generation.

[8]Otherwise such a history could be coded into the name of the tree.

6.2 Metarules for MC-TAG-DL

In this section, I will extend the definitions for metarules of the last section to the MC-TAG-DL framework. Most definitions need little adaption, however it is not obvious how the appearance of multiple trees within an elementary structure (i.e. within a tree sequence) of an MC-TAG-DL should be handled.

In an MC-TAG-DL with metarules, the input- and output-patterns of a metarule have to describe tree sequences instead of single trees. Therefore, a pattern has to consist of a sequence of single tree patterns (as defined above for regular TAGs). Also, it needs to be decided whether there must be one pattern for every tree in a tree sequence or if there is an (explicit or implicit) variable which can match multiple trees. For example, if the sequence-pattern has two tree patterns and a given tree sequence has four trees, two of which match the tree patterns of the sequence-pattern, will this count as a match?

Another issue are the dominance links. Certainly, it must be possible to describe a tree sequence-pattern which contains no information about dominance links, i.e. allows all combinations of these.

This leads to the following definitions:

Definition 6.2.1 (Metarule for an MC-TAG-DL) *A* **metarule** *for a given MC-TAG-DL G and a finite set MV of* metavariables *has the form*

$$t_1 \Rightarrow t_2$$

where t_1 and t_2 are tree sequences with the exception that any node may be a metavariable. Every metavariable which occurs in t_2 also occurs in t_1.

Figure 6.6 gives an example for a metarule in an MC-TAG-DL. It shows a version of the passive metarule for German. It applies to tree sequences as the sequence for the german verb *geben* in figure A.13 of the appendix, repeated here as 6.5. The result of the application of the metarule is the tree sequence for the corresponding passive phrase (e.g. *... daß der Ball den Kindern gegeben wird.*), shown in figure 6.7. Note that the specific features used in this example are meant only to clarify the example and not to represent a specific linguistic analysis.

Substitution of a metavariable is defined exactly as in definition 6.1.2 and the convention 6.1.1 for drawing metavariables also applies for tree sequence metarules. Also the definition 6.1.3 for the (result of the) application of metarule applies to tree sequences almost identically, the only exception being that the substituted tree sequence input

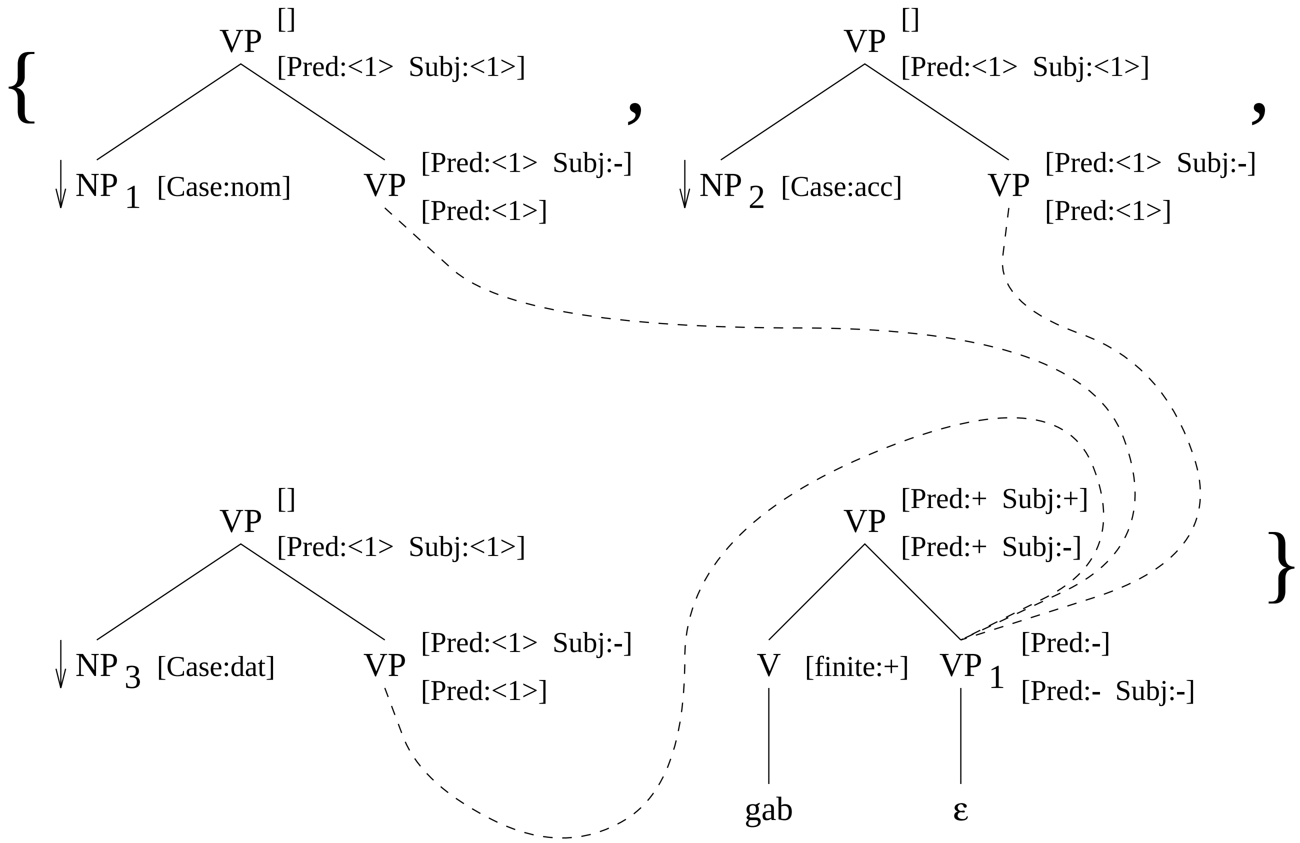

Figure 6.5: Elementary tree sequence for active form of verbs like "geben."

pattern must be identical to the tree sequence and the result of the application is of course also a tree sequence. Also the boundary M must apply to a tree sequence instead of a tree.

Also the definitions for *MC-TAG-DL with metarules*, the definitions for derived elementary tree sequences, tree-set and language of such a grammar apply almost identically and are therefore not repeated here.

6.3 Discussion

The two principles for bounding the application of metarules given in section 6.1.4 are further supported by linguistic motivations. The linguistic generalizations that metarules are designed to describe are the so-called lexical and/or syntactical rules. Both kind of rules have properties that restrict the application of metarules. First, they are not applied in an unbounded recursive manner and second they apply only on a bounded domain of locality. This reflects exactly the two restrictions of section 6.1.4

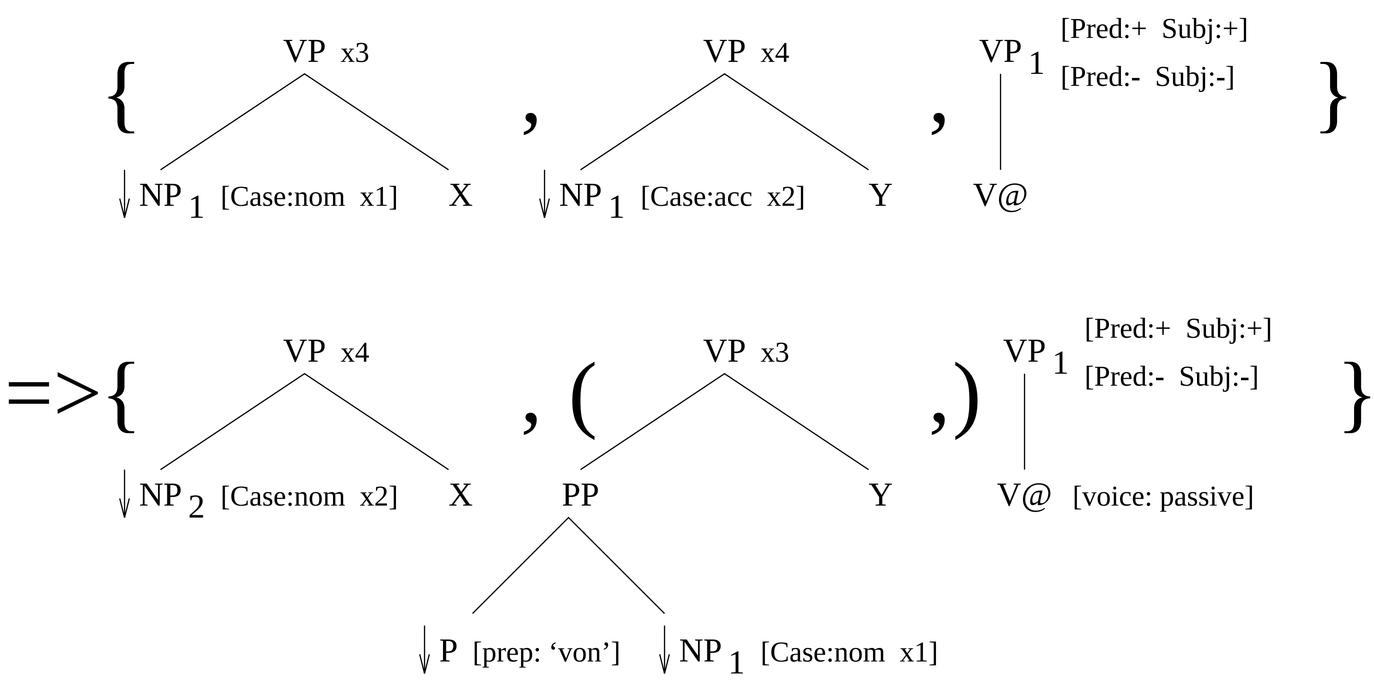

Figure 6.6: The metarule for passive in German.

For some phenomena like passive it is an open question whether they should be described as lexical or syntactic rules. By using metarules to describe both kinds of rules uniformly, a generalization is achieved that makes such a distinction unnecessary.

6.3.1 Selection of Metarules by Lexical Entries

A problem that arises with lexical and syntactic rules (described as metarules) and the organization of trees in tree-families is that not all verbs with the same predicate-argument structure select for exactly the same elementary trees. As Abeillé in [Abe90] points out, there are a number of phenomena that have to be handled in the lexicon. Especially for idioms not all of the matching lexical/syntactical rules can be applied. But this also holds for verbs like, e.g., *rumor* which can only be used in passive voice. So although *rumor* would select for the tree family Tn0VS it only selects the passive trees of this tree-family.

Therefore a way of representing such information in the lexicon must be given. One way to do this is to state for each entry in the (syntactic) lexicon and for each tree-family in this entry, which metarules are applicable for this entry in this tree-family.

Only experiments with a large grammar will reveal if such a simple approach is sufficient, it might turn out that some entries allow only certain *combinations* of metarules. This is a line of further research.

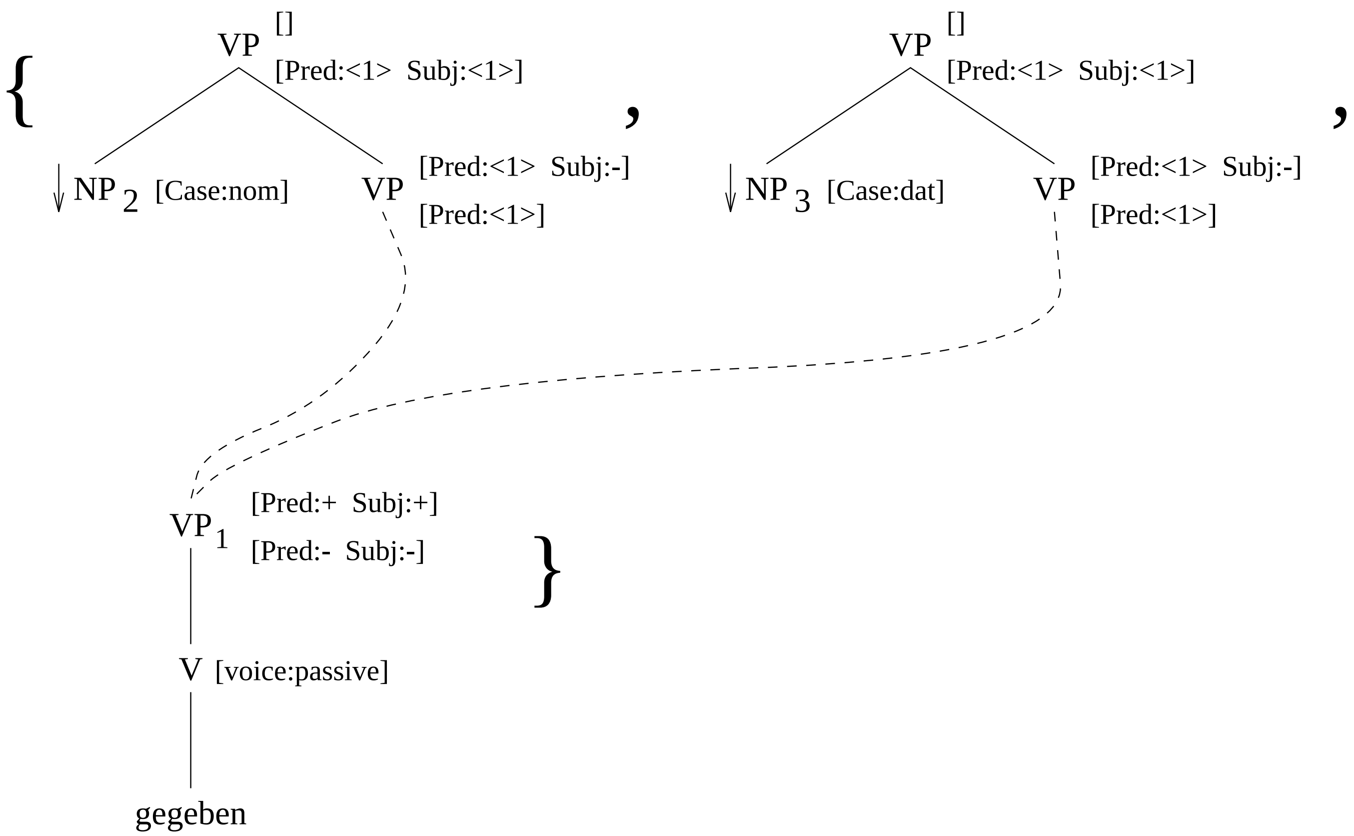

Figure 6.7: Elementary tree sequence for passive form of verbs like “geben” as the result of applying the passive-metarule.

Chapter 7

Hierarchical Lexicon for HyTAG

7.1 The Inheritance Network

With the introduction of metarules, the set of elementary structures of the grammar that need to be given explicitly shrinks to a rather small set of 'basic' elementary structures which could be called the 'core of the grammar.'

In this chapter, the concept of an "inheritance network" is introduced as an attempt to give also this 'core grammar' some structure. Naturally, this is also an additional approach to compactifying the lexicon. Unlike metarules, the inheritance network mechanism is not quite as strong with respect to the amount of redundancy that can be factored out. But this is to be expected since it represents the already highly compactified core of the grammar. The combination of a lexical hierarchy with lexical rules is used in a number of formalism for natural language, often influenced by the work of [FPW85, Fli87]. In the TAG framework such a proposal has been made for regular TAGs in [VSS92] (see also the work on lexical rules in [Abe90, Abe91]).

In the complete framework, all lexical entries select elementary structures from the inheritance network. And all metarules are originally applied to elementary structures from the inheritance network. So any (linguistic or other) principles that guide the construction of the elementary structures must already apply to the structures in the inheritance network.

An inheritance network, usually represented as a graph, can be viewed as a set of nodes (each representing some object, here the elementary structures), which is partially ordered by an inheritance relation. If the structural description of some elementary structure A is a proper subset of another elementary structure B then the structure B can be described more compact by saying "Structure B is like structure A plus some additional structure."

There are two possible ways of adding an inheritance network to the (MC-TAG-DL with metarules) formalism which I have developed so far:

(i) Define the entire grammar in terms of an inheritance network. I.e. every elementary tree sequence is seen as a node in the inheritance network. In this view, metarules transform one element of the inheritance network into another element of the inheritance network. As there is no inheritance relation between e.g. the elementary trees for verb-initial and verb-final clauses, the complete inheritance network would consist of a number of unconnected (except by metarules) subnetworks.

(ii) Define only the core of the grammar as an inheritance network. All other elementary structures (i.e. those derived by metarules) are not part of the inheritance network. The advantage is that a grammar writer really has to look only at the core of the grammar and the inheritance network stays relatively small. The drawback, however, is that (inheritance) relations between derived elementary structures are lost. Say, two derived elementary structures A and B are both derived by the same metarule from two elementary structures A' and B' of the inheritance network where one inherits from the other. Then there is still an inheritance relation between the two derived elementary structures A and B but this is not immediately available since the derived elementary structures are not part of an inheritance network anymore. However, the inheritance relation can easily be reconstructed.

The view adopted in the HyTAG framework follows the second approach. The definition is based on the concept of "basic elementary structures", the core inheritance network. This saves the advantages in actually constructing a grammar.

In a way, the reconstruction that is mentioned in (ii) can be included into the definition by keeping track of inheritance relations of (metarule-)derived elementary structures. This has advantages especially for text generation: All relations between different elementary structures are directly available from all (including derived) elementary structures. This is of importance for choosing between possible syntactic realizations of semantic concepts.

To clarify the concept of inheritance network, the definitions are first given for regular TAGs which makes them somewhat simpler and in a second step a few refinements which are necessary for the adaption to MC-TAG-DL are defined (see section 7.3).

Unlike classes in an object-oriented programming language, the classes in the inheritance network for a grammar should not be seen as objects which have local functions but the classes of the grammar are rather only data-structures[1]. So inheritance is only used for the data-structure. To make such an approach more viable, some deviation

[1] From a strictly object-oriented point of view the simple access functions are (the only) methods available for the object.

from the classic inheritance networks in object-oriented programming languages is needed. On the other hand, the objects which represent the elementary structures could be augmented with functions which realize the communication with other such objects, e.g. in order to generate a sentence with these objects. A similar approach is used in the TAG-GEN part of the WIP project [HFS91].

One way to achieve the reduction of the objects to a simple data structure is the introduction of just one simple data type. This approach is used in Unification based inheritance networks. In a unification based inheritance network, the basic data structure is a DAG (directed acyclic graph) and only final nodes of the graph may contain other data types (e.g. pronunciation patterns; for an example see the feature-structures of chapter 5).

Roughly, a refined class has some private specifications plus the inherited ones; all are unified to yield the full specifications of this class. This leads to some problems if some of the additional specifications are along a 'path' of some inherited features. In these cases, the private specifications need to refer to the inherited ones in some way. One way to achieve this is to give names to all classes and use these names in the specializations. This is exactly what I will use for the inheritance network of elementary trees.

The data-structure of an elementary structure consists mainly of the tree structure. (Or even more elementary of a set of nodes; the dominance relations can also be viewed as information which is associated with the respective nodes.) All other information,e.g. top- and bottom-features, adjoining constraints, substitution marker and dominance constraints, is then related to specific nodes.

Each elementary structure is identified by a unique name. These names are then used to specify the inherited structures. This is redundant in cases of single inheritance, but in principle, this is necessary in cases of multiple inheritance. (The actual definition of the sort of multiple inheritance which will be allowed in this system also makes these names redundant.)

7.2 Definitions

The basic concept for the definitions in this chapter are the individual names of nodes in each elementary structure of the inheritance network. These names allow for an easy reference to inherited information in elementary structures which further specify this inherited information.

Based on the definition of the "core inheritance network" and the actual elementary tree which is represented by a node in the inheritance network, I then define how the inheritance network and metarules are combined and how the complete grammar (the "full set of elementary trees") is constructed. Section 7.3 gives the adaptions necessary for an inheritance network in an MC-TAG-DL grammar.

Definition 7.2.1 (Core Inheritance Network)

A **Core Inheritance Network** *IN is a set of nodes (called "encoded trees") of the following form:*

A **node of an inheritance network** *is a pair ET*[2] *= (S,A) where S the set of* super-nodes *is a possibly empty set of nodes from IN and*

A the set of additional structure *is a set of triples (N,DS,LP) where*

> *N is a node from "the tree represented by the super nodes" (see definition 7.2.2 below),*
>
> *DS the* additional dominance structure *is a tree and*
>
> *LP,* the additional precedence information, *is a term of the form "L < DS < R" where*
>
> > *L is either the label "left" or a daughter node N' of N or an additional dominance structure DS' from another additional structure and*
> >
> > *R is either the label "right" or a daughter node N" of N or an additional dominance structure DS" from another additional structure.*

Definition 7.2.2 (Represented Tree of a Node)

R(ET), the elementary **tree represented by a node** *ET = (S,A) of an inheritance network IN is constructed as follows:*

In a first step the elementary trees represented by all nodes from S, the set of super nodes, are unified[3]*.*

The next step is the addition of all additional structures (N,DS,LP) from A where the tree DS becomes a subtree of the encoded tree ET as daughter of node N. The

[2]ET stands for "encoded tree" as the node of the inheritance network actually represents an elementary tree of a grammar; also it is thus easier to distinguish from the nodes of the trees.

[3]Note that there always has to be at least one root node (i.e. a node without super nodes) in the inheritance network in order to make this definition meaningful. The result of this unification need not be one single tree. However, if it consists of several trees then the next step must result in the combination into one tree. If, for whatever reason, one wishes to exclude multiple inheritance, the set S can be defined to contain at most one element.

position of DS among the other daughters of N is determined by the linear precedence information LP as follows:

If "left" < DS *is in LP then DS becomes the leftmost daughter;*
if DS' < DS *is in LP then DS becomes a daughter to the right of DS';*
if N' < DS *is in LP then DS becomes a daughter to the right of N'.*

If DS < "right" *is in LP then DS becomes the rightmost daughter;*
if DS < DS" *is in LP then DS becomes a daughter to the left of DS";*
if DS < N" *is in LP then DS becomes a daughter to the left of N".*

Only if the result is an elementary tree (as defined for TAGs), R(ET) is defined as this tree.

Definition 7.2.3 (Set of elementary trees of the inheritance network)

ET(IN), the **set of all elementary trees of an inheritance network** *IN is defined as the set of the elementary trees R(ET) represented by the encoded trees ET of the inheritance network IN:*

$$ET(IN) := \{R(ET) \mid ET \in IN\}$$

At this point, a formal definition of "TAGs with inheritance" could follow. But as discussed above, the inheritance network and metarules will be defined as one integrated framework and we will immediately proceed to a definition of "TAGs with metarules and inheritance." The important part is the integration of metarules into the inheritance network:

Definition 7.2.4 (TAG inheritance network with metarules)

A **TAG inheritance network with metarules** *INM is a pair INM = (I, M) where I is a core inheritance network as defined above (definition 7.2.1) and M is a set of metarules as defined in section 6.1.5.*

The set ET(INM), the **full set of elementary trees** *of an inheritance network with metarules is defined recursively as follows:*

- *All 'elementary trees of the inheritance network' are in the full set of elementary trees: ET(I) $\subseteq$ ET(INM).*
- *All elementary trees that are the result of the application of some metarule*[4]

[4] Note that "application of a metarule" is only defined if the resulting tree is "smaller" than the given boundary M. Thus the set of elementary trees of the inheritance network is still finite.

$m \in M$ to some elementary tree γ in the full set of elementary trees are also in the full set of elementary trees: $\gamma \in ET(INM)$ and $m \in M \Rightarrow m(\gamma) \in ET(INM)$.

Definition 7.2.5 (TAG with metarules and inheritance)

A **TAG with metarules and inheritance** *is a 4-tuple (N, T, S, INM) where N is a set of non-terminal symbols, T is a set of terminal symbols, S is the start symbol from N and INM is a TAG inheritance network with metarules as defined in definition 7.2.4 above.*

The definition of all other terms (e.g. tree-set, language) is identical to the definitions of TAGs, where the set of elementary trees of a TAG is replaced by ET(INM), the 'full set of elementary trees' of the inheritance network.

Figure 7.1 shows a simple example of an inheritance network for English. The graphical representation is used to make the inheritance relations more prominent, the elementary trees represented by the nodes of the inheritance network are explicitly given to the right.

7.3 Adaption for MC-TAG-DL

All ideas and concepts for inheritance networks apply identically to MC-TAG-DL, the actual form of the elementary structures (trees in TAG vs. tree sequences in MC-TAG-DL) is mainly independent from the definition of the inheritance network. The only part of the definitions above that needs some revision is the introduction of tree structure in addition to the inherited structure. In an MC-TAG-DL, a new tree sequence may have additional elements in its sequence and for additional structures (subtrees) within given (inherited) elements of the sequence, the element to which the structure belongs needs to be identified.

Definition 7.3.1 *A* **core inheritance network for an MC-TAG-DL** *is defined as in definition 7.2.1, with the following modifications:*

Additional structures become 4-tuples (N,DS,LP,DL), with an additional slot for dominance links.

If in an additional structure (N,DS,LP,DL) the entries for N and LP are empty, the dominance structure DS is to be added as another element into the represented tree sequence.

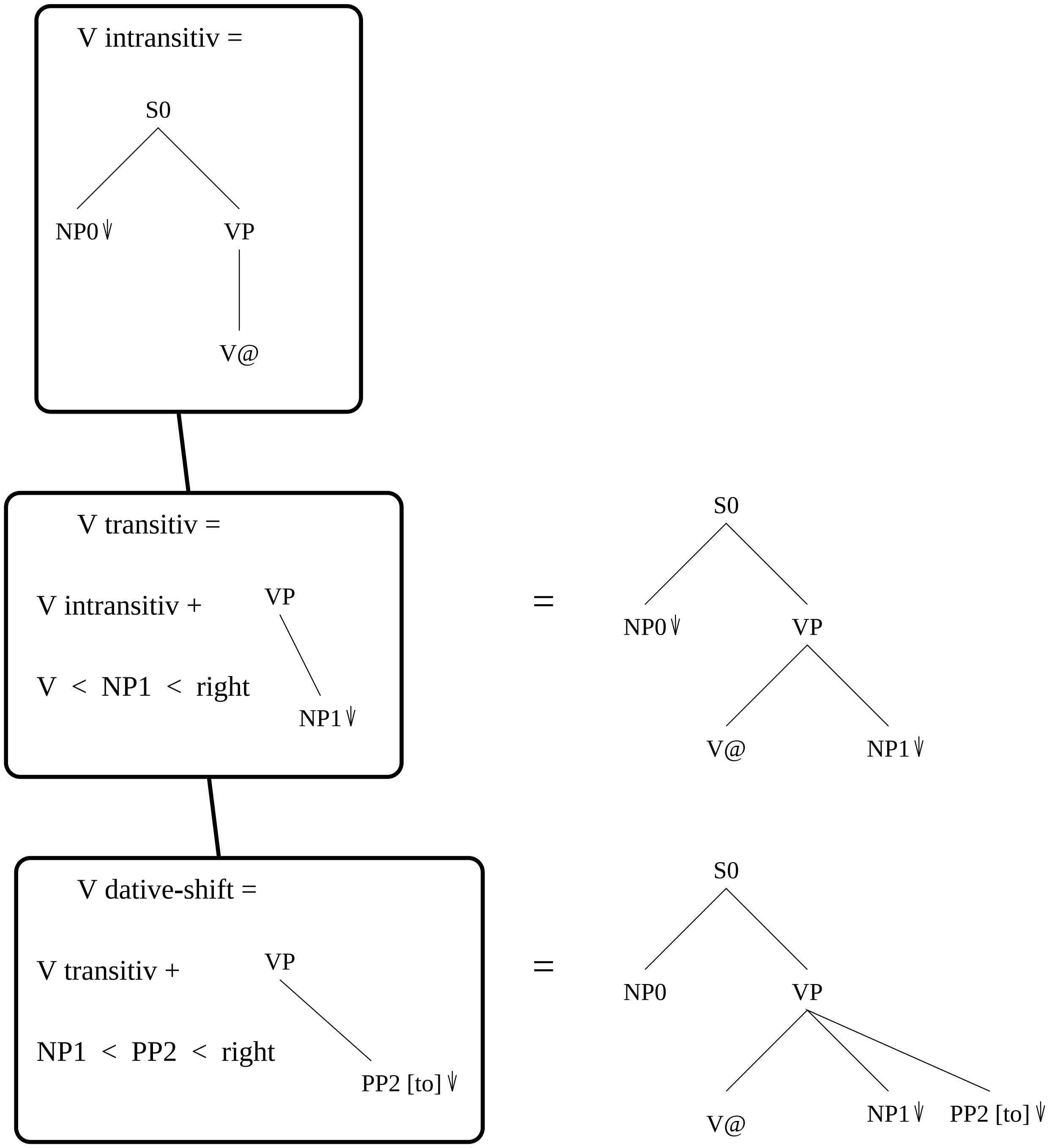

Figure 7.1: Part of a simple inheritance network for English.

Definition 7.3.2 *An* **MC-TAG-DL inheritance network with metarules** *INM is a pair INM = (I, M) where I is a core inheritance network as defined above (definition 7.2.1) and M is a set of (MC-TAG-DL–)metarules as defined in section 6.2.*

The set ES(INM), the **full set of elementary structures** *of an inheritance network with metarules is defined recursively as follows:*

- *All 'elementary structures of the inheritance network' are in the full set of elementary structures: ES(I) $\subseteq$ ES(INM).*
- *All elementary structures that are the result of the application of some metarule m in M to some elementary structure γ in the full set of elementary structures are also in the full set of elementary structures: $\gamma \in$ ES(INM) and $m \in M \Rightarrow m(\gamma) \in$ ES(INM).*

Finally, we can define HyTAG as the integration of MC-TAG-DL, metarules and an inheritance network:

Definition 7.3.3 (HyTAG)

A **HyTAG** *is an* MC-TAG-DL with metarules and inheritance *which is a 4-tuple (N, T, S, INM) where N is a set of non-terminal symbols, T is a set of terminal symbols, S is the start symbol from N and INM is an MC-TAG-DL inheritance network with metarules as defined above.*

As with TAGs, the definition of all other terms (e.g. tree-set and language of a HyTAG, i.e. an "MC-TAG-DL with metarules and inheritance") is identical to the definitions for MC-TAG-DL, where the set of elementary structures of an MC-TAG-DL is replaced by ES(INM), the "full set of elementary structures," of the inheritance network.

Note, that the TAG and the MC-TAG-DL formalism do not pose any restrictions on the shape of the elementary trees of a grammar and thus on the resulting syntactic structures for analysed sentences. But linguistic theories have a clear view on the possible shapes of a phrase structure tree. So, for a complete system there needs to be a special module which expresses these constraints on the shapes of the elementary trees. In an inheritance network and metarule based framework, these constraints need only be stated on the comparatively small set of trees in the inheritance network and the tree patterns of the metarules.

7.4 Metarules vs. Inheritance Network

In the last two chapters, I have introduced two quite different mechanisms to give a compact representation of an MC-TAG-DL grammar. Both mechanisms have been proposed for rather orthogonal phenomena. It appears that there are two different dimensions along which an MC-TAG-DL grammar can be compacted. However, one might want to reconsider the usage of one mechanism for phenomena in both dimensions.

One approach could be to drop the metarules.

Without metarules

The "full set of elementary structures" (also called the "full grammar") as defined in definition 7.3.2 consists of a multitude of different elementary trees. A realization of this set as a hierarchy (or rather a set of hierachies) would result in a huge inheritance net which has one node for every elementary structure of the full grammar. As metarules are not merely adding to the structure of an elementary tree sequence but rather changing the structure, non of the regularities described by metarules can be expressed as inheritance relations. But the aim of the whole process of expressing the grammar in a heterogeneous framework is that of yielding a more compact representation. So the only feasible alternative (with the aim of compacting the representation) is to 'invent' a new means of compactifying a hierarchical representation of objects (elementary trees). Now, how could such a means look like? Certainly, the metarules that have been defined in this thesis are one possible alternative. I will briefly try to give a general dicussion of other possibilities.

One approach is a fundamental change in the design of the objects. The tree would no longer be represented directly, but rather the object would include the base tree plus functions for the application of all applicable metarules. This could be realized as follows:

All metarules are represented as objects from which other objects can inherit functions that change the tree. These functions would have to be inherited in a cumulative way, multiple inheritance would take place in this rather unusual way. As there are only finitely many metarules, there could of course be a slot for every possible metarule but that would loose (or at least cloud) the generalization that there can be an arbitrary number of such metarules.

No matter what the exact definition (realization) would be, in all cases, the metarules would have to be defined as some kind of function that changes a tree. So I propose to make this difference as explicit as possible by giving the phenomena a name of its own and also a mechanism of its own, i.e. the metarules as defined in chapter 6.

But also the inheritance mechanism changes the elementary tree sequences. Even though it does so in a strictly monotonic behaviour (which is the basic distinction between the two phenomena), this change of the elementary tree sequences can also be described by metarules. So a more promising approach seems to be:

Without an inheritance network

The obvious advantage of representing the hierarchical framework in terms of metarules would be a simpler formalism.

In principle, this can be done easily. But it would basically amount to a reinvention of (some aspects of) hierarchical representations. Assume that phenomena like 'also having an accusative object' were represented by metarules. Let us call such rules 'hierarchical metarules.'

As there is a mechanism for allowing or disallowing the application of metarules for individual lexical entries, for each entry there would be a list of the hierarchical metarules that define exactly the place this lexical entry would have in a hierarchical representation.

There is no strong reason (as in the case of representing metarules within the hierarchical framework) not to use such hierarchical metarules. But as the phenomena that is described with the hierarchical framework is of a different kind, and it also has already successfully been represented in this way (see the discussion in the beginning of this chapter), it seems advisable to keep this representation. It also makes the distinction between the non-monotonic operations of metarules and the (basically – at least as far as the tree structure is concerned) monotonic operations of inheritance more clear.

This leads to the conclusion that both mechanisms, i.e. metarules and the inheritance network are important and one can not be replaced by the other.

Chapter 8

Conclusion and Further Work

8.1 Summary

In this thesis, I have developed HyTAG, an architecture for a compact and efficient representation of syntactic knowledge in natural language processing. The system is based on MC-TAG-DL, a tree rewriting system which generates phrase structures as syntactic descriptions of natural language sentences. The MC-TAG-DL formalism is developed as a result of the study of certain natural language phenomena, in particular the Long-Distance Scrambling (LDS) phenomenon in free word order languages such as German and Korean which show the inadequacy of TAGs for their description. In order to prove this formally, the notion of derivational generative capacity is developed. The elementary structures of the MC-TAG-DL formalism which represent the grammar are elementary tree sequences, for which a compact and efficient representation is developed. This architecture is based on metarules and inheritance networks. Both mechanisms factor out redundancies of the grammar as well as capture linguistic generalizations. Not only can a grammar be written and maintained better in such a compact representation, it also becomes easier to ensure the accordance of a grammar to linguistic (or other) principles which constrain the phrase structures.

The first chapter reviews all the relevant formal systems and includes a new automaton model for TAGs and its generalization which is a candidate for an automaton model for LCFRS.

The next chapter first reviews arguments about natural language phenomena which are beyond the descriptive power of TAGs. In sections 2.3 and 2.4 new data about the Long-Distance Scrambling (LDS) phenomena in free word order languages is presented and analyzed. The following chapter 3 gives a formal characterization of the LDS phenomenon and proof the inadequacy of the class of LCFRS to describe the

LDS phenomenon. This becomes possible only with the introduction of a new means of characterizing a formalism, namely the notion of derivational generative capacity.

From this negative result the need arises to develop a new formalism that can adequately capture the LDS phenomenon while preferably keeping the desirable properties of TAGs. Chapter 4 gives two extensions of the TAG formalism which meet this requirement, FO-TAG and MC-TAG-DL. The latter turns out to be superior.

Chapter 5 investigates the relation between the MC-TAG-DL analysis of LDS and the analysis of unbounded dependencies in HPSG, an unification based formalism. It also discusses how the MC-TAG-DL formalism can be used as a target language for compiling HPSG.

Based on the developed formalism, HyTAG an architecture which can represent a grammar in the MC-TAG-DL formalism compact and efficiently is developed. Chapter 6 introduces metarules which can substantially reduce the redundancies in an MC-TAG-DL grammar with rules which also allow to capture linguistic generalizations. In section 6.1.4 I show how the problem of the unbounded generative power of metarules in other formalisms can be solved in a well motivated way in the MC-TAG-DL framework (and the TAG framework).

Chapter 7 introduces inheritance networks as a further means for a more compact representation of the elementary tree sequences and defines HyTAG as the integration of metarules, inheritance networks and the MC-TAG-DL formalism.

The appendix finally gives a small fragment of the syntax of German to show the benefits of the developed architecture.

Main Results

To wrap up, this thesis presents the following results:

2-SA, a new automaton model for TAGs is introduced; n-SA the generalization of 2-SA are a candidate for an automaton model for LCFRS. Long-Distance Scrambling (LDS), a phenomenon of Free Word Order Languages is investigated. The notion of "derivational generative capacity", a new method of characterizing the adequacy of syntactic formalism is introduced. We show that this formalization of LDS which we develop is beyond the derivational generative capacity of LCFRS and therefore beyond TAGs. We present FO-TAG and MC-TAG-DL, two new extensions to the TAG formalism which can adequately capture the LDS phenomenon and propose to use MC-TAG-DL. We demonstrate how the handling of LDS in an unification-based formalism (HPSG) can be compiled into MC-TAG-DL. MC-TAG-DL is proposed as the basic formalism for HyTAG, a formalism for a hybrid representation of syntactic knowledge. Two major aspects of HyTAG are metarules and an inheritance network which allow for a compact and efficient representation of syntactic knowledge. We define both for TAGs and for MC-TAG-DL. We give a well motivated defintion of metarules that unlike other approaches does not add to the formal power of the resulting formalism. Finally, a short fragment of German syntax demonstrates the benefits of the HyTAG system.

8.2 Further Work

Starting from the results of this thesis there are a number of interesting issues that remain open for further work. A lot of these concern not the area of syntactic descriptions itself but rather areas which interface with syntax in natural language processing. Also, there a number of formal problems and NLP issues within the field of syntactic respresentions.

A very interesting problem is the extension of the concepts of the MC-TAG-DL formalism to a rewriting system that is polynomial parseable. Such a system, called Unordered Vector Grammar (UVG) is currently investigated by Owen Rambow, results are to appear in the final version of [Ram92].

Another interesting topic is the new automaton model, n-SA introduced in this thesis. It remains to be shown that the version presented here is an exact model for the class of LCFRS languages. Another interesting topic is whether there is a further constrained version of n-SA which is an automaton model for Paren Grammars, a subclass of LCFRS. It seems that the restricted (i.e. nested) type of composition operation which Paren Grammars allow corresponds closely to the more or less nested ordering of information on the additional stacks of an n-SA.

As a parallel to the work in this thesis on the relation between the MC-TAG-DL formalism and the HPSG analysis of unbounded dependencies, it would be interesting to investigate the relationship between the FO-TAG formalism and especially its use to capture the LDS phenomenon and other unbounded dependencies and the "clause union" formalism which has been proposed within the HPSG formalism. This could also lead to results about the formal properties of clause union.

From a linguistic point of view, it would be interesting to work out how linguistic principles that guide the construction of the elementary tree sequences of an MC-TAG-DL grammar can be incorporated into the formalism. One obvious candidate are the principles of GB theory. Some principles, e.g. X-Bar theory can be incorporated quite easily and it becomes clear that the compact representation of a grammar in the HyTAG formalism is a great advantage. The principles need to be checked only for the comparatively small set of elementary tree sequences which are represented by the inheritance network and for the metarules. This ensures that the whole (and potentially large) set of elementary tree sequences which are represented by the HyTAG gramamar adhere to the principles. Other principles like the factoring of recursion and the co-occurrence restrictions are part of the HyTAG design, only the linguistic details (e.g. basing the co-occurrence restrictions on some part of the linguistic theory) need to be worked out.

The relation between lexicon and grammar has only been sketched in this thesis, it seems to be worthwhile to look at the possibility to incorporate a morphological component into the system which transforms the actual words (full forms) into their stems and the corresponding morphological information. This morphological information includes not only information which should be represented in the feature structures (e.g. agreement information) but also information regarding the phrase structure (e.g. finite verb vs. infinitive form). Therefore, such a morphological component will have to guide the selection process of elementary tree sequences from the inheritance network as well the selection of metarules.

The semantic component is the second interface besides the morphological component. The modularity and highly differentiated structure of a grammar developed in the HyTAG formalism should also be a benefit for the semantic component, e.g. the type of metarules that have been applied will correspond to certain semantic information.

Appendix A

A Fragment of German Syntax

This appendix defines a small fragment of German syntax in order to illustrate the benefits of the HyTAG system. It is not inteded to give a general linguistic analysis of German syntax. The syntactic structures of the example tree sequences are kept as simple as possible to show how the various components of HyTAG interact.

A.1 The Inheritance Network

Figure A.1 gives a sketch of the inheritance network for verbs which is defined in the following. This might seem extremely small, but it subsumes at least all the various subcategorizations shown in figure A.2.

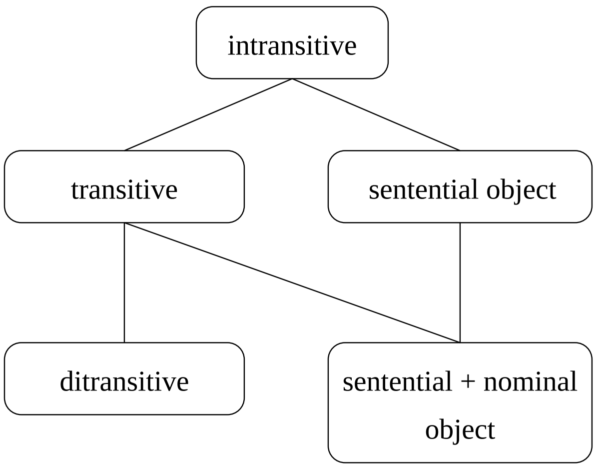

Figure A.1: Fragment of the inheritance network for German.

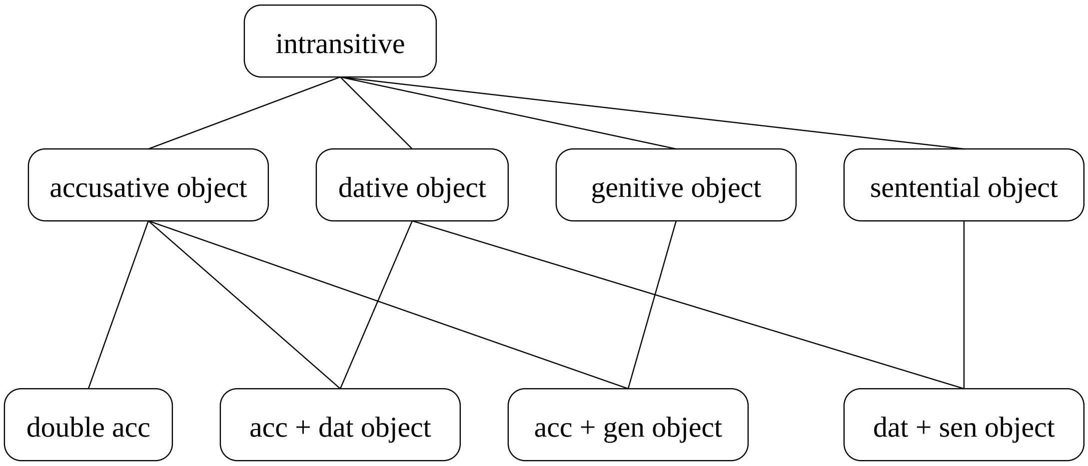

Figure A.2: Some verb classes subsumed by the inheritance network of figure A.1.

Figure A.3 shows the tree sequence for "V intransitive". It is selected by verbs like "schlafen" (sleep) or "denken" (think). The use of the "Pred" and "Subj" features is taken from [Ram92].

Figure A.4 defines the tree sequence for "V transitive". Note that the node NP_2 is not marked with case information. This is supplied by the lexical entries. By this mechanism, otherwise seperate nodes for verbs with dative, accusative or genitive objects can be collapsed. The node VP_1 is not a new node but repeated here from "V intransitive" to give a convenient way of defining the dominance link. "V transitive" is selected by verbs like "schieben" (push), "reparieren" (repair) and "gedenken" (commemorate).

Figure A.5 defines the tree sequence for "V ditransitive". The entry for "V ditransitive" is almost identical to the one for "V transitive", but note that it introduces a new tree, indicated here by naming the new NP node NP_3. "V ditransitive" is selected by verbs like "lehren" (teach) "geben" (give) and "überführen" (indict).

Figure A.6 defines the tree sequence for "V sentential". To keep this example grammar simple, I assume that sentential objects are substited. No linguistic claim is intended by this approach. As [Ram92] discusses, an approach where the matrix clause is adjoined into the subordinate clause should be preferred. The feature "PRO" is used to select only clauses with a PRO subject. These can be derived with the metarule of figure A.11. With the extended domain of locality that the tree sequences exhibit, feature equations could easily be stated between the subject of the matrix clause and the PRO node of the subordinate clause.

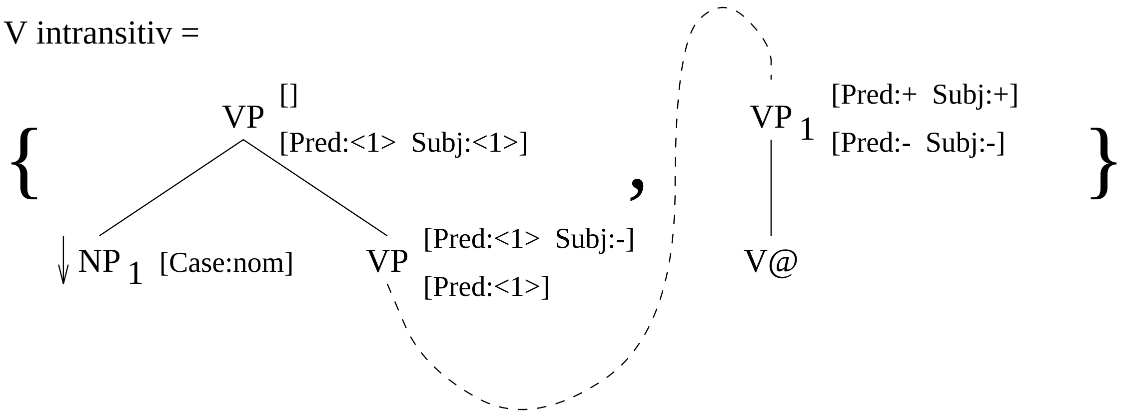

Figure A.3: Intransitive Verbs

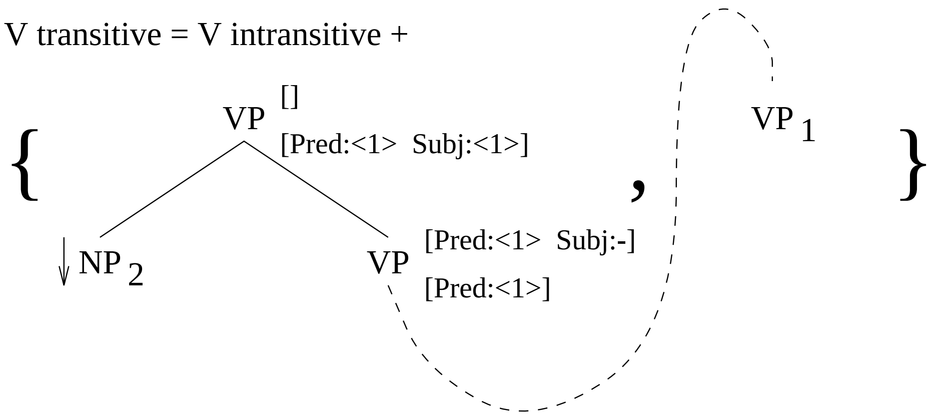

Figure A.4: Transitive Verbs

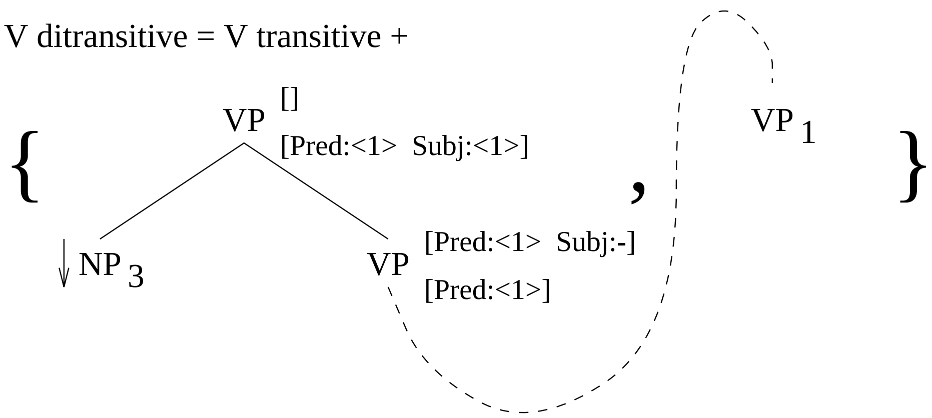

Figure A.5: Ditransitive Verbs

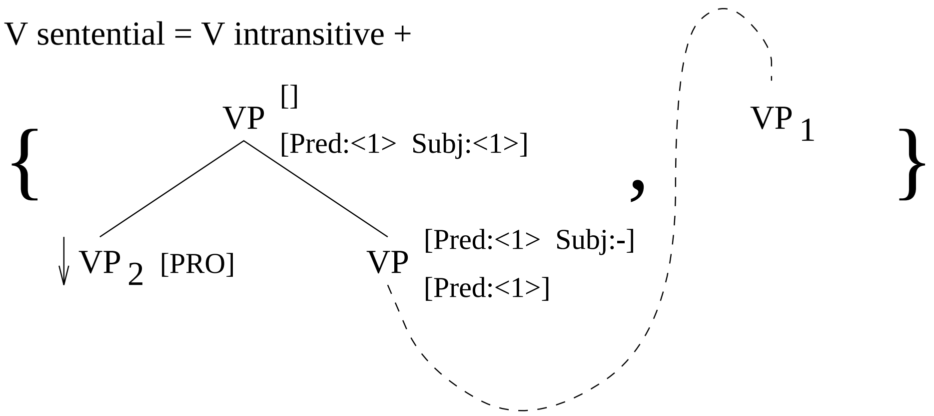

Figure A.6: Verbs taking a sentential object

Verbs which have a sentential and a nominal object select the tree sequence defined by "V sent&nom object" which can be defined as:

V sent&nom object := V sentential + V transitive

While the tree sequences defined in the inheritance network above are sufficient for verb-final clauses, there needs to be a way to create the verb second forms. If the finite verb is an auxiliary, the verb second clause can be derived by adjoining an auxiliary tree like the one shown in figure A.7.

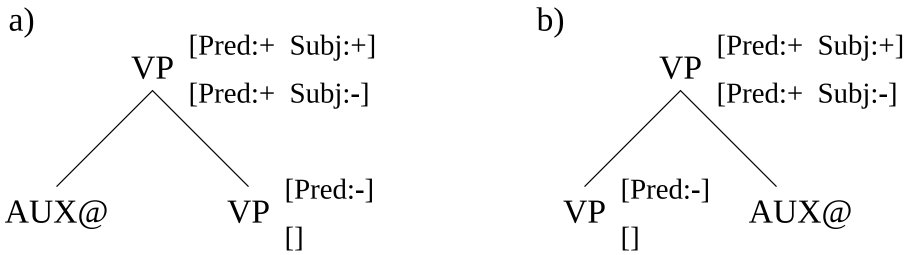

Figure A.7: Auxiliary tree for auxiliary verbs in a) verb second position and b) verb final position.

A.2 Metarules

If the verb is finite, the effects of adjoining an auxiliary at the verb second position must be achieved in some other way. One could express this by a metarule[1], as shown in figure A.8.

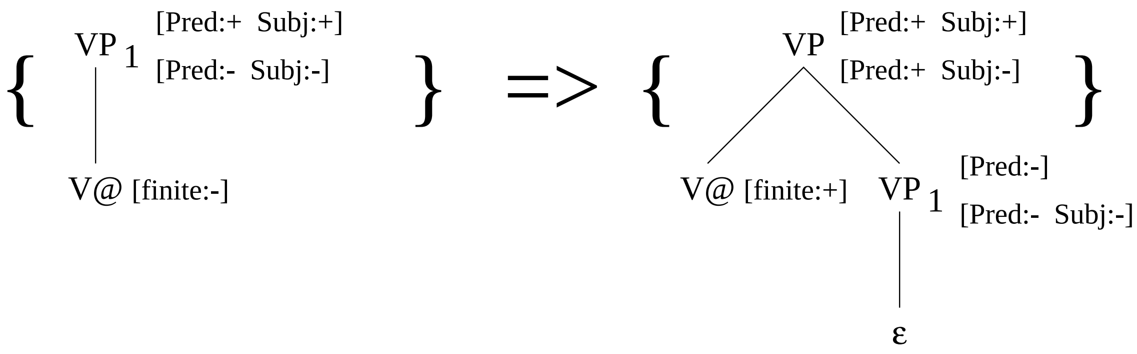

Figure A.8: Metarule for verb second clauses.

Functions which change the morphology or phonolgy assciated with the verb (as e.g. in HPSG) could be included in this metarule.

The passive metarule is shown in figure A.9. X and Y are metavariables, matching any node. x1 to x4 are supposed to indicate that all feature equations including the respective node must be matched and copied over to the resulting tree sequence. The parentheses around the PP tree are used mark it as optional. Keeping strictly with the defined formalism, there need to be two passive metarules, one with and one without the PP tree. The feature "[voice: passive]" is used to indicate that the features must ensure that the verb "werden" or "sein" is adjoined which form the passive in german.

Figure A.10 shows the metarule which derives relative clauses. Note that the resulting node NP_2 can only be substituted by pronomials (indicated by [pronom: +]). The [Subj:+] in the bottom feature of the VP dominating NP_2 ensures that no other argument can be adjoined at this node.

Figure A.11 shows the metarule which creates a PRO from an nominative argument. This can also be achieved by including an initial NP tree with PRO as its anchor, but this contradicts the idea of the grammar being lexicalized, i.e. having at least one overt lexical item in each elementary tree sequence.

[1] There is no claim intended about the linguistic adequacy of such an approach. However, this is a good example for a lexical rule.

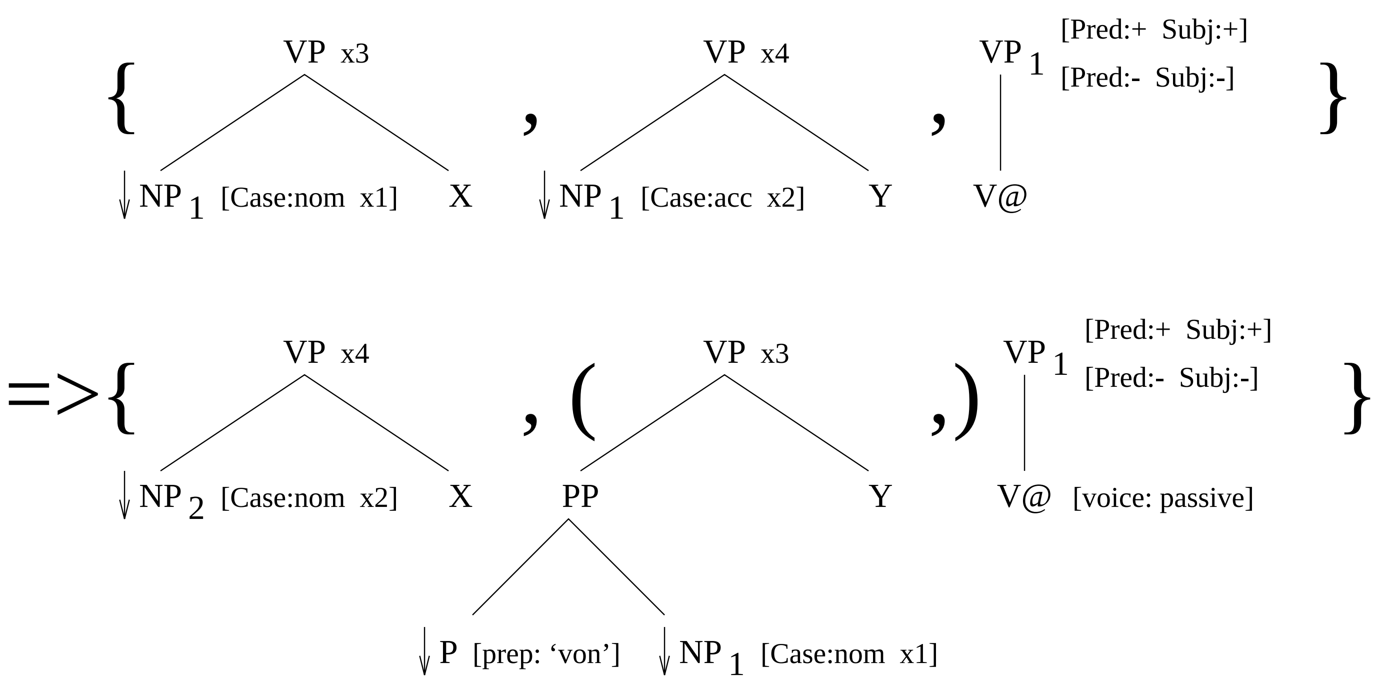

Figure A.9: The metarule for passive.

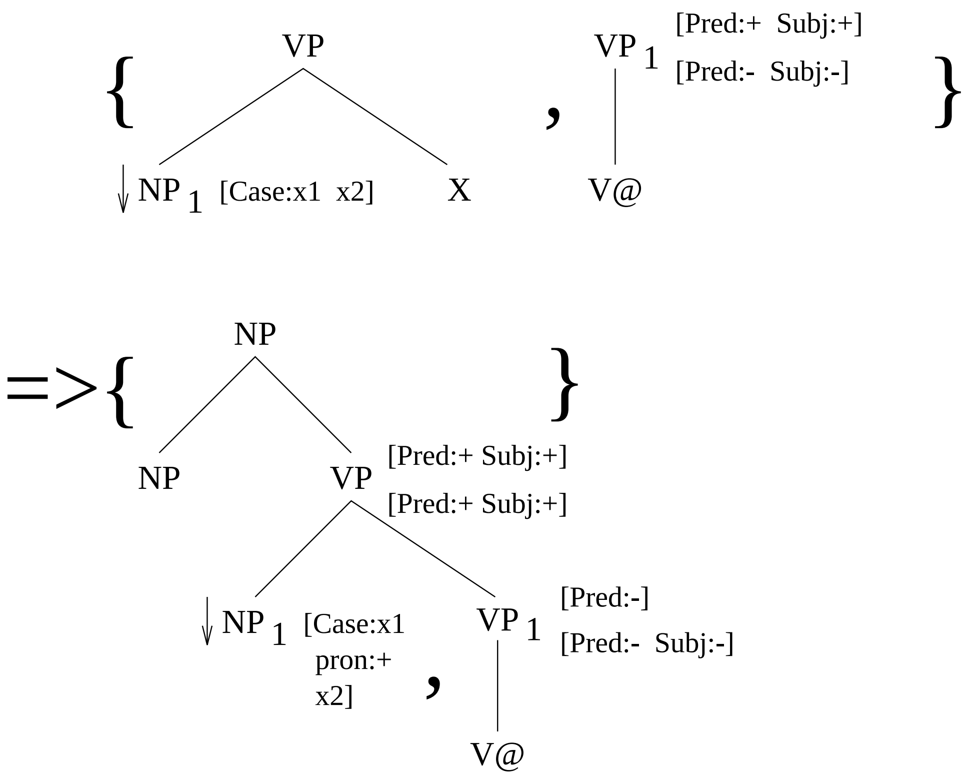

Figure A.10: The metarule for relative clauses.

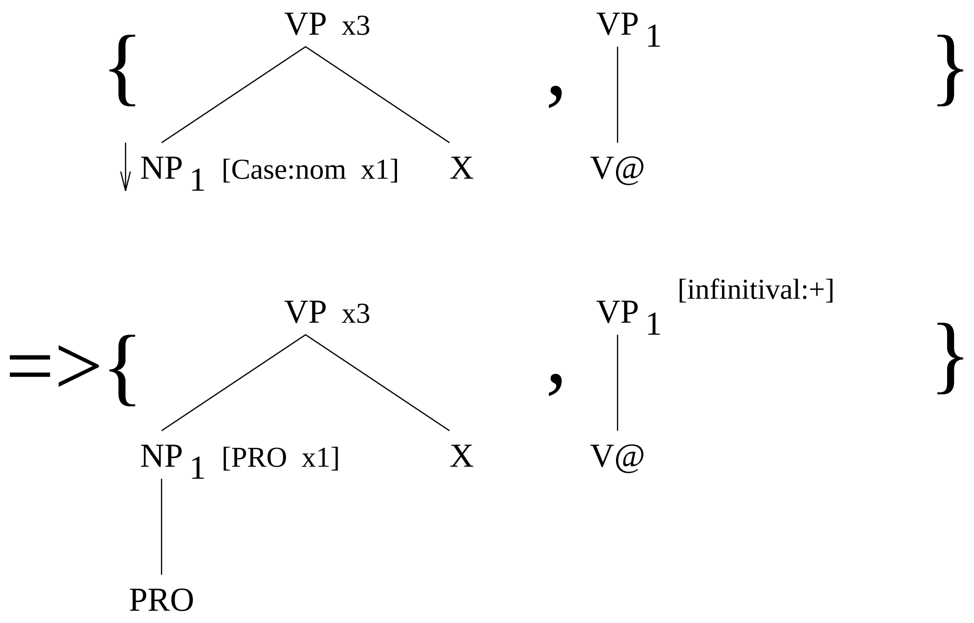

Figure A.11: Metarule for PRO clauses.

A.3 Lexical entries

The following short list includes the verbs used in the example sentences of section A.4 and the nodes of the inheritance network they select.

geben (give):	V ditransitive
reparieren (repair):	V transitive
versprechen (promise):	V sentential or V sent&nom object
haben (have):	auxiliary

All other words are nominals and appear as NPs in the example sentences. The only exception is the adverbial "gestern" (yesterday) which selects the auxiliary tree shown in figure A.12.

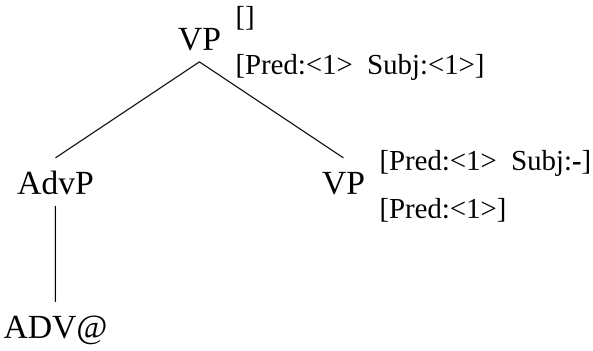

Figure A.12: Auxiliary tree vor adverbials.

A.4 Example Sentences and Analyses

This section gives a number of example sentences, including example sentences of the previous chapters. Also, an example sentence is given which illustrates the application of metarules. Note that we assume that prior to adjoining all trees of a sequence into a derived structure, sequence internal adjunctions can take place, see also section 4.2.3.

(1) Der Lehrer gab gestern den Kindern die Bücher.
"Yesterday, the teacher gave the children the books."

The first step in the derivation of sentence (1) is the selection of the node "V ditransitive" from the inheritance network. Then the verb second metarule is applied. The resulting tree sequence is shown in figure A.13.

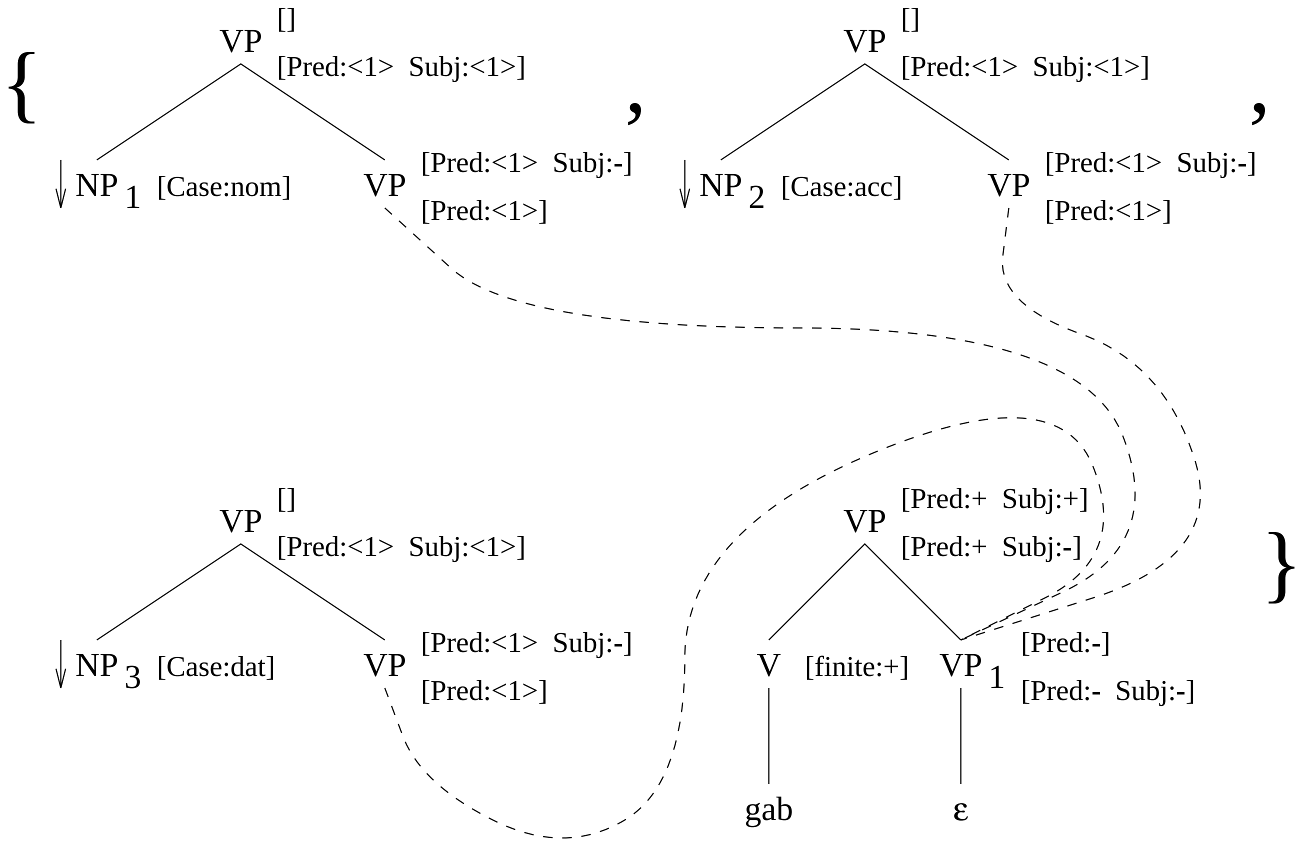

Figure A.13: Tree sequence for "gab" (gave).

Then the arguments of "geben" are adjoined and finally the adverbial is also adjoined. At this point all top and bottom feature structures can be unified. The resulting phrase structure for sentence (1) is shown in figure A.14. Obviously, sentence (1a) can be derived in the same way, as the arguments of "geben" can be adjoined in any order.

(1a) Die Bücher gab gestern den Kindern der Lehrer.

(2) ... daß ich dem Kunden den Kühlschrank zu reparieren versprochen habe.
"... that I have promised the client to repair the refrigerator."

(2a) ... daß ich den Kühlschrank dem Kunden zu reparieren versprochen habe.

Figure A.15 shows the tree sequence for the verb "versprechen" (promise) which is "V sent&nom object" taken directly from the inheritance network. The tree sequence for the verb "reparieren" is shown in figure A.16. It is derived from "V transitive" and the application of the PRO metarule. Figure A.17 shows the resulting phrase structure for the scrambled sentence (2a) which also includes the COMP, which is

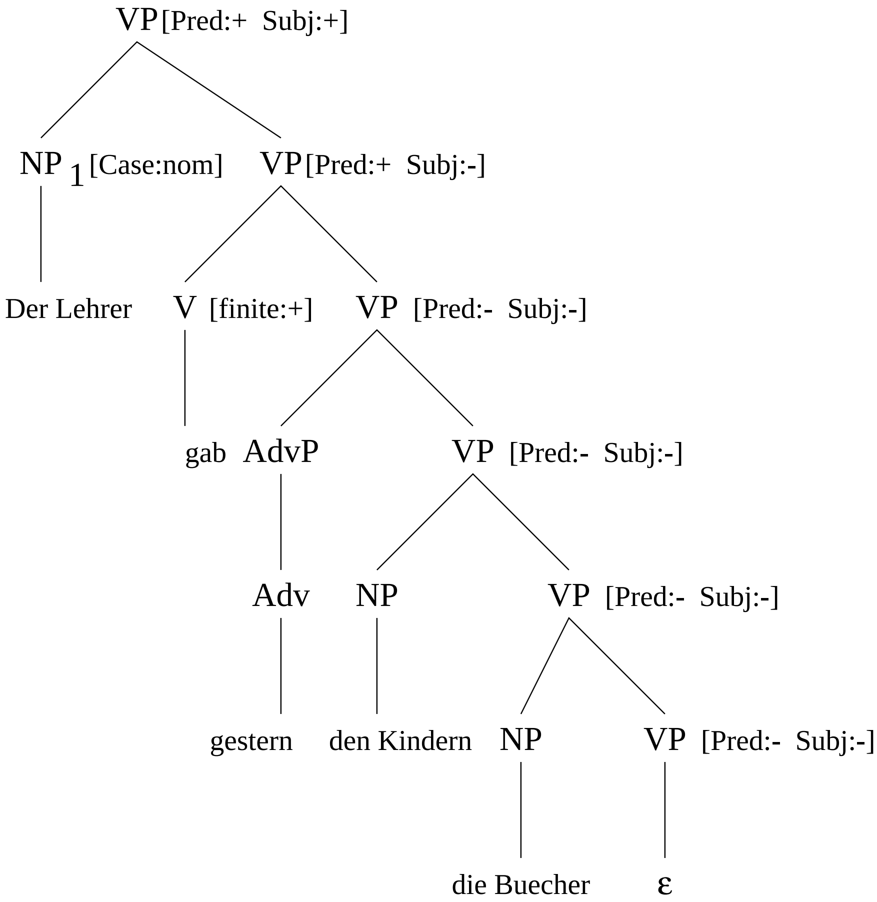

Figure A.14: Phrase structure for sentence (1).

adjoined into the derived structure. Of course, sentence (2) can also be derived.

(3) Die Kinder, denen die Bücher gegeben werden.
"The children who are given the books."

This phrase is derived from the elementary tree sequence V ditrans by first applying the passive metarule and subsequently applying the relative clause metarule. Figure A.18 shows the result of the application of the passive metarule (shown in figure A.9) to the tree sequence for "geben" (as shown in figure A.13). Figure A.19 shows the result of the subsequent application of the relative clause metarule (shown in figure A.10). The final result after tree sequence-internal adjunction is shown in figure A.20.

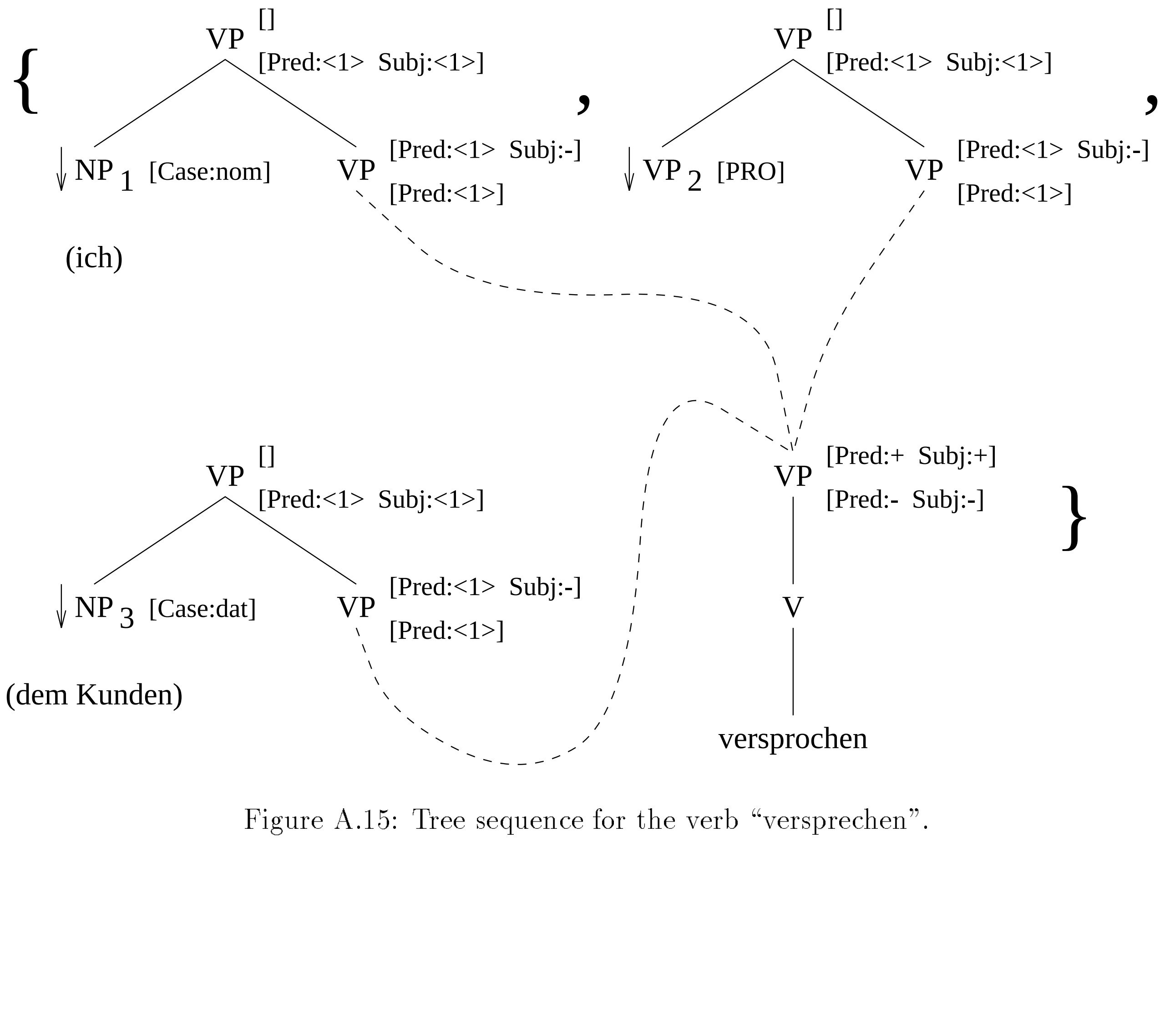

Figure A.15: Tree sequence for the verb "versprechen".

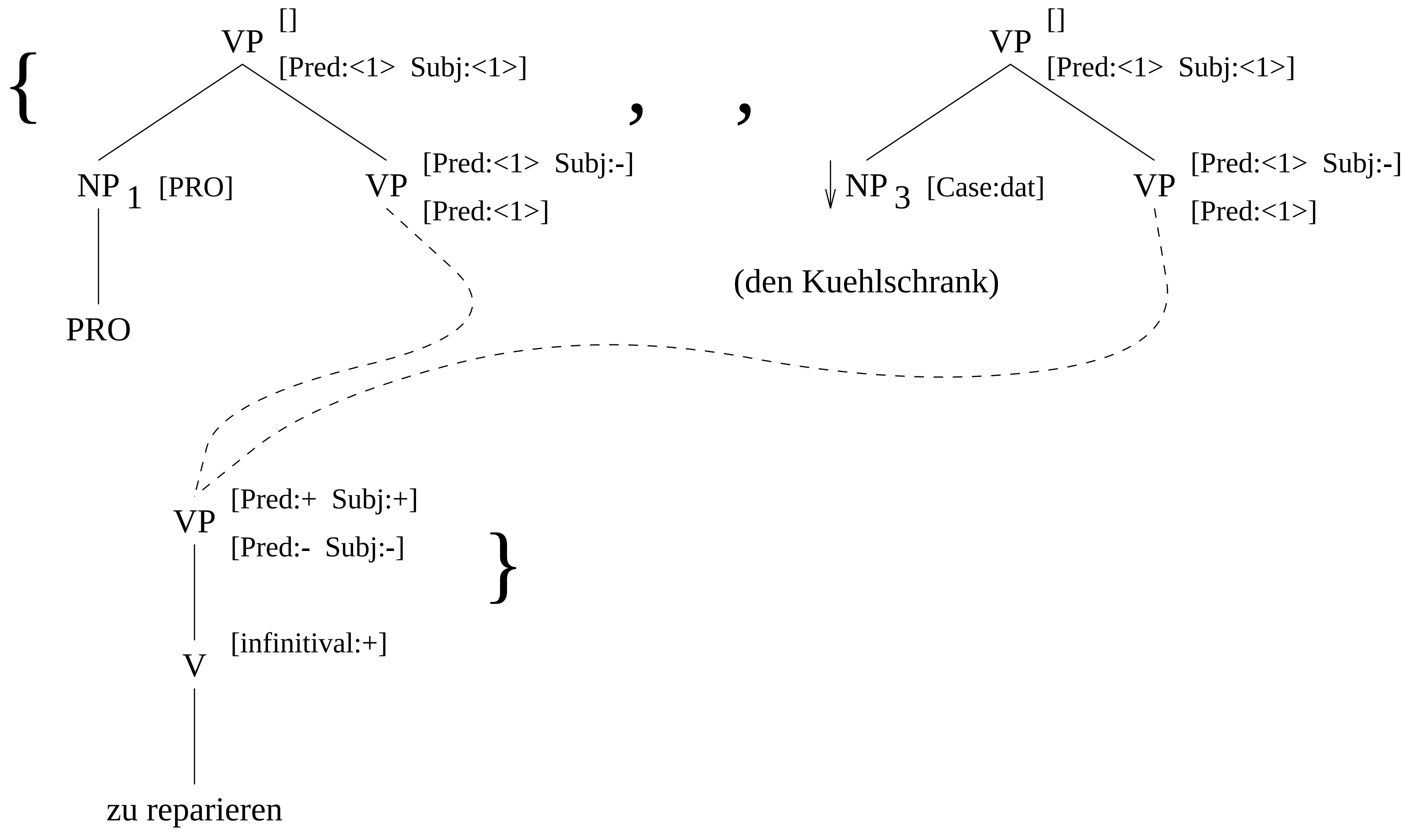

Figure A.16: Tree seqence for "reparieren" with PRO.

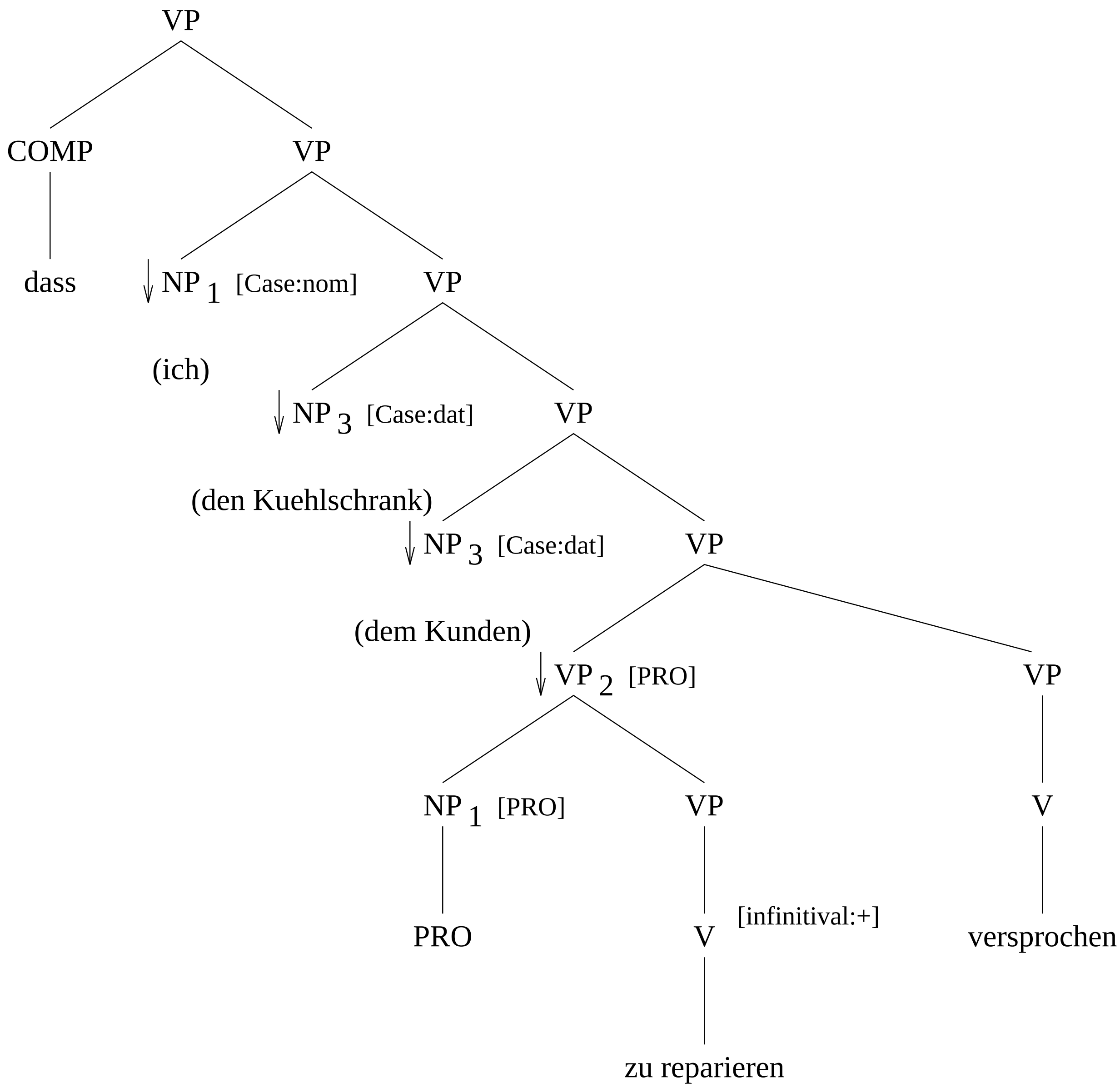

Figure A.17: Phrase structure for sentence (2a).

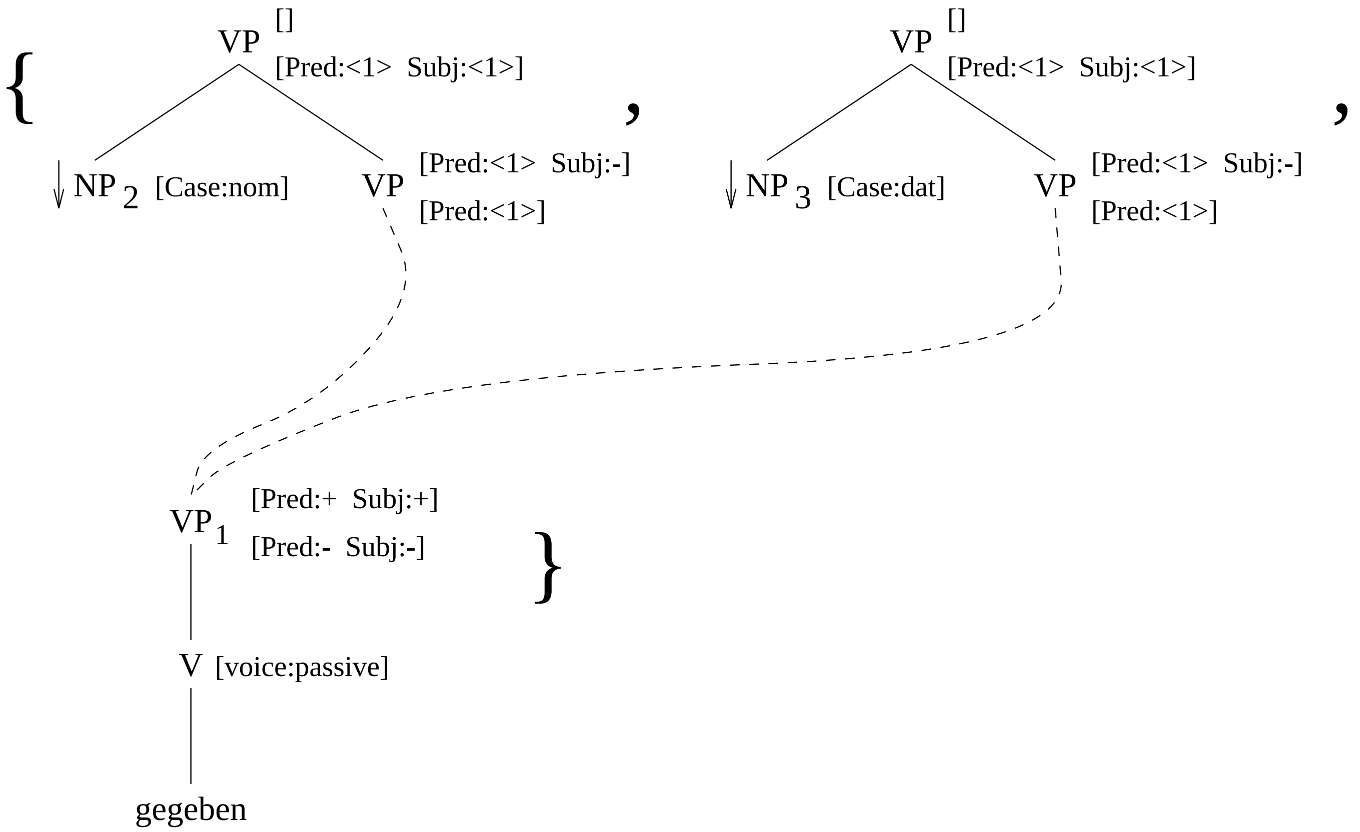

Figure A.18: Tree sequence for "geben" in passive voice.

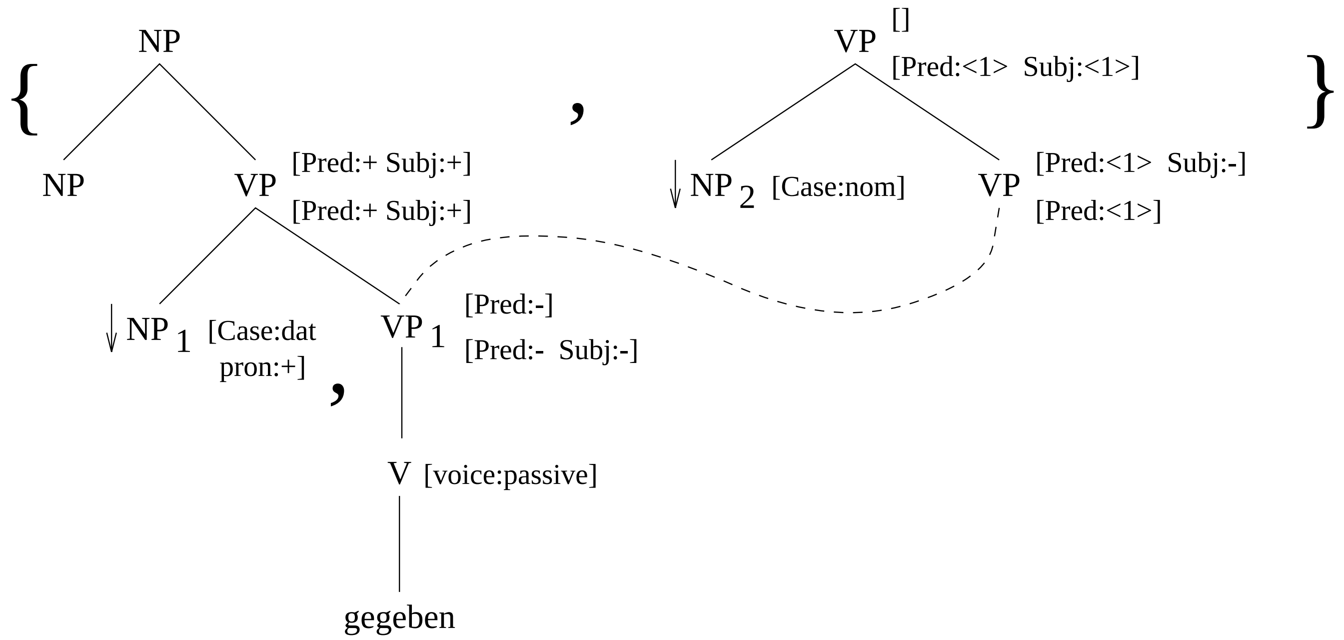

Figure A.19: Tree sequence for passive "geben" in a relative clause.

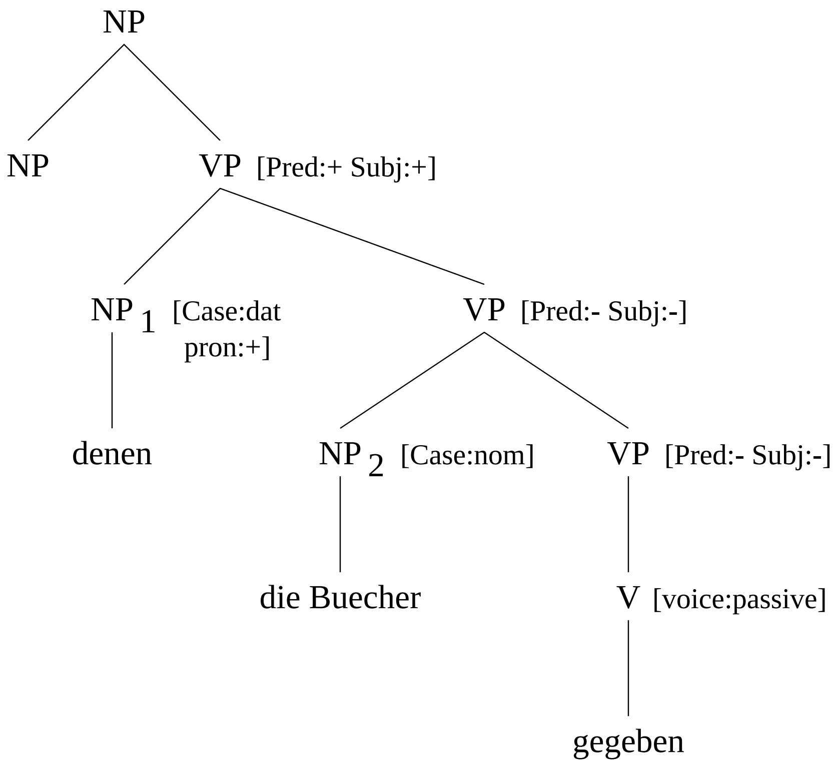

Figure A.20: Phrase structure for passive "geben" in a relative clause.

Bibliography

[ABCS90] Anne Abeillé, Kathleen Bishop, Sharon Cote, and Yves Schabes. A Lexicalized Tree Adjoining Grammar for English. Technical Report MS-CIS-90-24, Department of Computer and Information Science, University of Pennsylvania, April 1990.

[Abe88] Anne Abeillé. A Lexicalized Tree Adjoining Grammar for French: the general framework. Technical Report MS-CIS-88-64, University of Pennsylvania, 1988. To appear in Machine Translation, 1990.

[Abe90] Anne Abeillé. Lexical constraints on syntactic rules in a Tree Adjoining Grammar. In 28^{th} *Meeting of the Association for Computational Linguistics (ACL'90)*, 1990.

[Abe91] Anne Abeillé. *Une grammaire lexicalisée d'arbres adjoints pour le français*. PhD thesis, Université Paris 7. UFR de Linguistique. Laboratoire d'Automatique Documentaire et Linguistique., Paris. France., Janvier 1991.

[Aho69] A. V. Aho. Nested stack automata. *J. ACM*, 15:383–406, 1969.

[AS89] Anne Abeillé and Yves Schabes. Parsing idioms in Tree Adjoining Grammars. In *Fourth Conference of the European Chapter of the Association for Computational Linguistics (EACL'89)*, Manchester, 1989.

[Bec89] Tilman Becker. Metaregeln für Tree Adjoining Grammars. Master's thesis, RWTH Aachen, 1989.

[Bec90] Tilman Becker. Tag-languages $\subset$ TAG-with-constraints-languages. Unpublished Manuscript, 1990.

[Bec92a] Tilman Becker. A new automaton model for TAGs: 2-SA. In *Proceedings of the TAG+ workshop 1992*, University of Pennsylvania, Philadelphia, 1992.

[Bec92b] Tilman Becker. A new automaton model for TAGs: 2-SA, 1992. Submitted to the Journal Computational Intelligence.

[BJE86] B. R. Bryant, D. Johnson, and B. Edupuganty. Formal specification of natural language syntax using two-level grammar. In *Proceedings of the 11th Conference on Computational Liniguistics*, pages 527–532, Bonn, 1986.

[BJR91] Tilman Becker, Aravind Joshi, and Owen Rambow. Long Distance Scrambling and Tree Adjoining Grammars. In *Proceedings of the 5th Conference of the European Chapter*. ACL, 1991.

[BKPZ83] J.W. Bresnan, R.M. Kaplan, S. Peters, and A. Zaenen. Cross-serial dependencies in Dutch. *Linguistic Inquiry*, 13:613–635, 1983.

[BPSH91] Béla Buschauer, Peter Poller, Anne Schauder, and Karin Harbusch. Tree Adjoining Grammars mit Unifikation. Technical report, DFKI, Saarbrücken, 1991.

[BR90] Tilman Becker and Owen Rambow. Formal aspects of Long Distance Scrambling. Unpublished Paper, University of Pennsylvania, 1990.

[BR91] Tilman Becker and Owen Rambow. Long-Distance Scrambling, TAGs, and HPSG. Presented at the Workshop on *HPSG and German Word Order*, Saarbrücken, 1991.

[BRN92] Tilman Becker, Owen Rambow, and Michael Niv. The derivational generative power, or, scrambling is beyond LCFRS. Technical report, University of Pennsylvania, 1992. A version of this paper was presented at MOL3, Austin, Texas, November 1992.

[Cho57] Noam Chomsky. *Syntactic Structures*. Mouton, The Hague, 1957.

[CU77] J. C. Cleaveland and R. C. Uzgalis. *Grammars for Programming Languages*. Elsevier North-Holland, New York, 1977.

[Eve75] Arnold Evers. *The transformational cycle in Dutch and German*. PhD thesis, University of Utrecht, 1975. Distributed by the Indiana University Linguistics Club.

[Fli87] Daniel P. Flickinger. *Lexical Rules in the Hierarchical Lexicon*. PhD thesis, Stanford University, 1987.

[FPW85] D. Flickinger, C. Pollard, and T. Wasow. Structure-sharing in lexical representation. In *Proc. of the 23th ACL*, pages 262–267, Chicago, IL, 1985.

[GKPS85] Gerald G. Gazdar, Ewan Klein, Geoff K. Pullum, and Ivan A. Sag. *Generalized Phrase Structure Grammars*. Blackwell Publishing, Oxford, 1985. Also published by Harvard University Press, Cambridge, MA.

[Gua92] Yonggang Guan. *Klammergrammatiken, Netzgrammatiken und Interpretationen von Netzen*. PhD thesis, Universität des Saarlandes, Saarbrücken, 1992.

[Har89] Karin Harbusch. *Ein effizientes Parsingverfahren für TAGs*. PhD thesis, Universität des Saarlandes, Saarbrücken, 1989.

[HFS91] K. Harbusch, W. Finkler, and A. Schauder. Incremental syntax generation with Tree Adjoining Grammars. In W. Brauer and D. Hernandez, editors, *Verteilte Künstliche Intelligenz und kooperatives Arbeiten: 4. Internationaler GI-Kongreß Wissensbasierte Systeme, Proc.*, pages 363–374. Springer, Berlin, Heidelberg, 1991.

[HU79] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, 1979.

[JLT75] Aravind K. Joshi, L. S. Levy, and M. Takahashi. Tree Adjunct Grammars. *J. Comput. Syst. Sci.*, 10(1), 1975.

[Jos85] Aravind K. Joshi. How much context-sensitivity is necessary for characterizing structural descriptions—Tree Adjoining Grammars. In D. Dowty, L. Karttunen, and A. Zwicky, editors, *Natural Language Processing—Theoretical, Computational and Psychological Perspectives*. Cambridge University Press, New York, 1985. Originally presented in a Workshop on Natural Language Parsing at Ohio State University, Columbus, Ohio, May 1983.

[Jos87a] Aravind K. Joshi. An introduction to Tree Adjoining Grammars. In A. Manaster-Ramer, editor, *Mathematics of Language*. John Benjamins, Amsterdam, 1987.

[Jos87b] Aravind K. Joshi. Word-order variation in natural language generation. Technical report, University of Pennsylvania, 1987.

[Jos89] Aravind K. Joshi. Processing crossed and nested dependencies: an automaton perspective on the psycholinguistic results. *Language and Cognitive Processes*, 1989.

[JVSW90] Aravind K. Joshi, K. Vijay-Shanker, and David Weir. The convergence of mildly context-sensitive grammar formalisms. Technical Report MS-CIS-90-01, Department of Computer and Information Science, University of Pennsylvania, Philadelphia, 1990.

[Kas87] Michael Kashket. A government-binding based parser for Warlpiri, a free-word language. Technical Report 993, MIT Artificial Intelligence Laboratory, 1987.

[Kas92] Robert Kasper. Compiling Head-Driven Phrase Structure Grammar into Lexicalized Tree Adjoining Grammar. In *Proceedings of the TAG+ workshop 1992*, University of Pennsylvania, Philadelphia, 1992.

[KJ85] A. Kroch and A. K. Joshi. Linguistic relevance of Tree Adjoining Grammars. Technical Report MS-CIS-85-18, Department of Computer and Information Science, University of Pennsylvania, 1985.

[Kro88] A. Kroch. Unbounded dependencies and subjacency in a Tree Adjoining Grammar. In A. Manaster-Ramer, editor, *Mathematics of Language*. John Benjamins, Amsterdam, 1988.

[Lee92] Young-Suk Lee. Scrambling and the Adjoined Argument Hypothesis. Paper presented at the TAG+ Workshop, 1992.

[Lee93] Young-Suk Lee. *Scrambling as case-driven obligatory movement*. PhD thesis, University of Pennsylvania, 1993.

[LS91] Young-Suk Lee and Beatrice Santorini. Resolving Webelhuth's paradox: evidence from German and Korean. In *Proceedings of the Tilburg Workshop on Scrambling*. 1991.

[Mah89] Anoop Mahajan. On the A/A-bar distinction: scrambling and weak crossover and binding in Hindi. Ms., M.I.T., 1989.

[Mil91] Philip H. Miller. Scandinavian extraction phenomena revisited: Weak and strong generative capacity. *Linguistics and Philosophy*, 14:101–113, 1991.

[MR88] Alexis Manaster-Ramer. Book review: *The Formal Complexity Of Natural Language*. *Computational Linguistics*, 14(4):98–103, 1988.

[MZ82] J. Maling and A. Zaenen. A phrase structure account of scandinavian extraction phenomena. In *The Nature of Syntactic Extraction*, pages 229–282. Reidel, Dordrecht, 1982.

[Net91] Klaus Netter. Clause union phenomena and complex predicates in German. Technical Report R1.1.B (Part 1), DYANA, 1991.

[Pol84] C. Pollard. *Generalized phrase structure grammars, head grammars and natural language*. PhD thesis, Stanford University, Stanford, CA, 1984.

[Pol93] Peter Poller. Inkrementelles und direktes Parsing von LD/TLP-TAGs. Master's thesis, Univerität des Saarlandes, Saarbrücken, 1993.

[PS87] Carl Pollard and Ivan A. Sag. *Information-Based Syntax and Semantics. Vol 1: Fundamentals.* CLSI, 1987.

[PS91] Carl Pollard and Ivan A. Sag. 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, 1991.

[Rad91] Daniel Radzinski. Chinese number-names, Tree Adjoining Grammars and mild context-sensitivity. *Computational Linguistics*, 17(3):277–299, 1991.

[Ram91] Owen Rambow. Long-distance scrambling as underspecification: an analysis based on Tree Adjoining Grammars. In *Penn Review of Linguistics*. University of Pennsylvania, Philadelphia, 1991.

[Ram92] Owen Rambow. Natural language syntax and formal systems. Dissertation Proposal, University of Pennsylvania., 1992.

[RB90] Owen Rambow and Tilman Becker. Scrambling and Tree Adjoining Grammars. Unpublished Paper, University of Pennsylvania, 1990.

[RBN92] Owen Rambow, Tilman Becker, and Michael Niv. Scrambling is beyond LCFRS. Paper presented at MOL3, the 3rd Meeting on Mathematics of Language, 1992.

[Rea91] Mike Reape. Clause union and word order variation in germanic. Technical Report R1.1.B, DYANA, University of Edinburgh, Edinburgh, 1991.

[Ris87] E. S. Ristad. Revised Generalized Phrase Structure Grammar. In *Proc. of the 25 th Meeting of the Association for Computational Linguistics (ACL'87)*, pages 243–250, Stanford, CA, 1987.

[RS92] Owen Rambow and Giorgio Satta. A two-dimensional hierarchy for local rewriting systems. Paper under preparation, University of Pennsylvania, 1992.

[Sai89] Mamoru Saito. Scrambling as semantically vacuous a'-movement. In Mark Baltin and Anthony S. Kroch, editors, *Alternative conceptions of phrase structure*, pages 182–200. University of Chicago Press, Chicago, 1989.

[SAJ88] Yves Schabes, Anne Abeillé, and Aravind K. Joshi. Parsing strategies with 'lexicalized' grammars: Application to Tree Adjoining Grammars. In *Proceedings of the* 12^{th} *International Conference on Computational Linguistics (COLING'88)*, Budapest, August 1988.

[San90] Beatrice Santorini. Scrambling and INFL in German. Paper presented at CUNY; Manuscript, University of Pennsylvania, 1990.

[Sch90] Yves Schabes. *Mathematical and Computational Aspects of Lexicalized Grammars.* PhD thesis, University of Pennsylvania, Philadelphia, PA, August 1990. Available as technical report (MS-CIS-90-48, LINC LAB179) from the Department of Computer Science.

[Shi84] Stuart M. Shieber. The design of a computer language for linguistic information. In 22^{nd} *Meeting of the Association for Computational Linguistics (ACL'84)*, Stanford, July 1984.

[Shi85] Stuart B. Shieber. Evidence against the context-freeness of natural language. *Linguistics and Philosophy*, 8:333–343, 1985.

[SJ88] Yves Schabes and Aravind K. Joshi. An Earley-type parsing algorithm for Tree Adjoining Grammars. In 26^{th} *Meeting of the Association for Computational Linguistics (ACL'88)*, Buffalo, June 1988.

[SMFK91] Hiroyuki Seki, Takashi Matsumura, Mamoru Fujii, and Tadao Kasami. On multiple context-free grammars. *Theoretical Computer Science*, 88:191–229, 1991.

[UP83] H. Uszkoreit and S. Peters. On some formal properties of metarules. Technical Report Technical Note 305, SRI International, Stanford, 1983.

[VS87] K. Vijay-Shanker. *A Study of Tree Adjoining Grammars.* PhD thesis, Department of Computer and Information Science, University of Pennsylvania, 1987.

[VSJ85] K. Vijay-Shanker and A. K. Joshi. Some computational properties of Tree Adjoining Grammars. In 23^{rd} *Meeting of the Association for Computational Linguistics*, pages 82–93, 1985.

[VSJ88] K. Vijay-Shanker and A.K. Joshi. Feature structure based Tree Adjoining Grammars. In *Proceedings of the* 12^{th} *International Conference on Computational Linguistics (COLING'88)*, Budapest, August 1988.

[VSJ90] K. Vijay-Shanker and Aravind K. Joshi. Unification based Tree Adjoining Grammars, 1990. University of Pennsylvania, Philadelphia.

[VSS92] K. Vijay-Shanker and Yves Schabes. Structure sharing in Lexicalized Tree-Adjoining Grammars. In 20^{th} *Meeting of the Association for Computational Linguistics (ACL'92)*, 1992.

[VSWJ87] K. Vijay-Shanker, David J. Weir, and Aravind K. Joshi. Characterizing structural descriptions produced by various grammatical formalisms. In 25^{th} *Meeting of the Association for Computational Linguistics (ACL'87)*, 1987.

[vWMPK69] A. van Wijngaarden (ed.), B. J. Mailloux, J. E. L. Peck, and C. H. A. Koster. Report on the algortithmic language ALGOL 68. *Numerische Mathematik*, 14:79–218, 1969.

[Web89] Gert Webelhuth. *Syntactic saturation phenomena and the modern Germanic languages*. PhD thesis, University of Massachussetts, 1989.

[Wei88] David J. Weir. *Characterizing Mildly Context-Sensitive Grammar Formalisms*. PhD thesis, Department of Computer and Information Science, University of Pennsylvania, 1988.

[Zwi63] A. Zwicky. Some languages that are not context-free. In *Quarterly Progress Report of the Research Laboratory of Electronics*, pages 290–293. MIT 70, 1963.