Tree Description Grammars and Underspecified Representations

von

Laura Kallmeyer

Philosophische Dissertation
angenommen von der Neuphilologischen Fakultät
der Universität Tübingen
am 5. Dezember 1998

Tübingen

1999

Gedruckt mit Genehmigung der Neuphilologischen Fakultät der Universität Tübingen

Hauptberichterstatter:	Prof. Dr. Erhard W. Hinrichs
Mitberichterstatter:	Prof. Dr. Uwe Mönnich Prof. Dr. Marga Reis Prof. Dr. Hugo Volger
Dekan:	Prof. Dr. Bernd Engler

Acknowledgements

A project of this scope necessarily owes much to the efforts of many people besides the author. First of all and most importantly I would like to mention my two supervisors Erhard Hinrichs and Uwe Mönnich. Both of them have supported me very much in the course of the writing of this dissertation.

Erhard Hinrichs has contributed a lot of fruitful suggestions and ideas to the project of this thesis, I would like to thank him very much for pointing out to me many interesting questions and problems related to my work. Without his help, in particular in the beginning it would have taken much more time for me to decide on the subject of this thesis and to see the interesting issues related to this subject.

Uwe Mönnich has supported me crucially; he has spent much time trying patiently to understand me (which sometimes was not easy) and bringing together my ideas, his intuitions and the formal details of my work. I would like to thank him very much for his invaluable comments and his encouragement that contributed a lot not only to the writing but also to the finishing of this thesis.

Exceptional thanks are also due to Frank Morawietz with whom I had a lot of interesting and fruitful discussions and who even read a previous version of the whole thesis and made a lot of useful comments. Without these comments, the dissertation would be much less readable and clear, not only to others but also to me. Furthermore, I would like to thank Tom Cornell, who discussed parts of the thesis with me and thereby helped me to see some things more clearly.

During the time I spent writing the thesis, I was member of the Graduiertenkolleg Integriertes Linguistik Studium, and I owe a lot to the Kolleg in several respects: firstly, it was encouraging to be part of a community where most people also had to write a thesis and where people supported each other in such a way that one did not feel left alone with one's project. Secondly, I had a lot of valuable discussions on different aspects of my work with members of the Graduiertenkolleg, particularly with Paul King and Graham Katz, who were both postdocs of the Kolleg. Both read various drafts of various parts of the thesis, and, besides general discussions, gave me a lot of very useful detailed advice. For their help, I would like to thank them both very much.

Furthermore, I am grateful to Hélène Boudault for native speaker judgements. I would also like to thank Claus Hümpel and Andreas Reindl, both no computational linguists, who read parts of drafts of the thesis, and by their comments helped me to write more clearly and intelligibly.

Besides those involved in the writing of the thesis, I would like to thank Marga Reis and Hugo Volger for acting as expert witnesses of my dissertation and for pointing out in their reports several interesting aspects to be considered in the future.

Apart from all the encouragement and support I got in Tübingen, I am very grateful to the Institute for Research in Cognitive Science (IRCS) at the University of Pennsylvania, Philadelphia, and particularly to Aravind Joshi for providing the opportunity to interact with researchers at IRCS and, furthermore, to have my thesis published as technical report at IRCS.

Abstract

Tree Description Grammars and Underspecified Representations

Author: Laura Kallmeyer

Supervisors: Erhard W. Hinrichs and Uwe Mönnich

In this thesis, a new grammar formalism called *(local) Tree Description Grammar (TDG)* is presented that generates tree descriptions. This grammar formalism brings together some of the central ideas in the context of *Tree Adjoining Grammars (TAG)* on the one hand, and approaches to underspecified semantics for scope ambiguities on the other hand.

First a general definition of TDGs is presented, and afterwards a restricted variant called *local TDGs* is proposed. Since the elements of a local TDG are tree descriptions, an extended domain of locality as in TAGs is provided by this formalism. Consequently, local TDGs can be lexicalized, and local dependencies such as filler gap dependencies can be expressed in the descriptions occurring in the grammar.

The tree descriptions generated by local TDGs are such that the dominance relation (i.e. the reflexive and transitive closure of the parent relation) need not be fully specified. Therefore the generation of suitable underspecified representations for scope ambiguities is possible.

The generative capacity of local TDGs is greater than the one of TAGs. Local TDGs are even more powerful than set-local multicomponent TAGs (MC-TAG). However, the generative capacity of local TDGs is restricted in such a way that only semilinear languages are generated. Therefore these languages are of constant growth, a property generally ascribed to natural languages.

Local TDGs of different rank can be distinguished depending on the form of derivation steps that are possible in these grammars. This leads to a hierarchy of local TDGs. For the string languages generated by local TDGs of a certain rank, a pumping lemma is proven that allows to show that local TDGs of rank n can generate a language $L_i := \{a_1^k \cdots a_i^k \mid k \geq 0\}$ iff $i \leq 2n$ holds.

In order to describe the relation between two languages, synchronous local TDGs are introduced. The synchronization with a second local TDG does not increase the generative power of the grammar in the sense that each language generated by a local TDG that is part of a synchronous pair of local TDGs, also can be generated by a single local TDG.

This formalism of synchronous local TDGs is used to describe a syntax-semantics interface for a fragment of French which illustrates the derivation of underspecified representations for scope ambiguities with local TDGs.

Contents

Chapter 1

Introduction

In a grammar, linguistic information can be encoded in different ways. Two broad perspectives on linguistic information can be distinguished: either the information is represented by structures, e.g. trees or attribute-value graphs, or by descriptions of structures, e.g. formulas in some description language. From the first, more traditional point of view, information is considered as a model whereas in the second case information is regarded as a theory. Henceforward, the first approach will be called *structure-based* whereas the second will be called *description-based*.

This thesis adopts the second approach: it brings together ideas from the area of tree-generating formalisms, in particular description-based variants of Tree Adjoining Grammars and proposals for underspecified semantic representations. The goal of the thesis is the definition of a grammar formalism generating tree descriptions that are suitable as underspecified representations for scope ambiguities.

The two general perspectives – structure-based and description-based – are by no means exclusive. Looking at the history of grammar formalisms, not only purely structure-based or purely description-based formalisms can be found but also formalisms that show both aspects. Already in early transformational generative grammar (Chomsky 1957, 1965) both perspectives can be observed since the notion of transformation as a relation between trees is mentioned on the one hand, opposed to structure descriptions and transformation descriptions on the other hand. The contrast between structure and description becomes more explicite with the so-called unification-based or constraint-based grammar formalisms developed in the 80s, for example Kay's functional unification grammar (FUG) and the PATR-II grammar (see Shieber 1986). The two terms *unification* versus *constraint* show the status of these approaches between structure- and description-based since unification is a structural notion whereas a constraint is a logical formula. The starting point for these formalisms was to view a grammar as a set of objects, e.g. feature structures (directed acyclic attribute value graphs) that represent partial linguistic information. With the increasing complexity of these objects the need arose to define a logic that helps to understand the meaning of such objects. Work in this area has been done for example by Shieber (1992) with an application to PATR-II, and by Kasper and Rounds (1990), Johnson (1990), Dawar and Vijay-Shanker (1990) for feature structures in general. In contrast to these constraint-based approaches, more recent formalizations of grammar tend to become more rigorously description-based. This is for example the case for HPSG (Head-Driven Phrase Structure Grammar, see Pollard and Sag

1994). In the logic for HPSG proposed by King (1994), descriptions are formulas denoting fully specified objects. A description in such a logic is either true of false for a specific object. Usually, it is true for more than one object, and in this sense it may be called underspecified. However, there are no objects representing partial information in such a purely description-based approach.

A similar development from structure-based concepts of grammar to more and more description-based formalisms can be observed in the research related to Tree Adjoining Grammars (TAG) (Joshi et al. 1975, Joshi 1987). TAGs originally consist of trees from which larger trees can be generated. Therefore TAGs were introduced as purely structure-based. Vijay-Shanker (1992) proposes to define TAGs in a description-based way. Following these ideas, Rogers and Vijay-Shanker develop a definition of so-called quasi-trees. In their earlier approach (Rogers and Vijay-Shanker 1992), quasi-trees are objects representing partial information and together with these objects, a logic is defined denoting quasi-trees. Therefore this first formalization of quasi-trees is both structure- and description-based, i.e. it is comparable to constraint-based grammar formalisms as mentioned above. In their later formalization (Rogers and Vijay-Shanker 1994), however, the notion of quasi-trees as objects is left aside and the approach is purely description-based. The work presented in this thesis is very much influenced by the work on tree descriptions and TAG done by Rogers and Vijay-Shanker.

In the case of King's logic for HPSG, a grammar is viewed as a single huge description that is true for all grammatical objects. Therefore there is no generation or derivation process involved in this approach. However, the choice of a description-based formalism does not necessarily exclude the notion of generation. Such an approach might be derivation-based in the sense that there might be a set of elementary descriptions, and from these elementary descriptions other descriptions might be derived by combining two descriptions in each derivation step. This is for example the case for the description-based approach to TAGs in Vijay-Shanker 1992.

There are several reasons for preferring a purely description-based approach to a structure-based approach: in a structure-based approach some ordering relation is needed that compares two models with respect to their degree of partiality. In the context of unification-based grammars, this relation is usually called subsumption. A first model subsumes a second model if the second model extends the first or, in other words, if the second represents more information than the first. The problem is that subsumption is not monotonic with respect to classical negation and implication. This problem is avoided if one deals with descriptions instead of structures representing partial information. Even in a structure-based approach such as TAG, a partial order on the objects can be found, namely the relation of "being derivable by adjunction from" (i.e. $\langle \gamma_1, \gamma_2 \rangle$ is in the relation iff γ_2 can be derived from γ_1). Here, as pointed out by Vijay-Shanker (1992), the problem of non-monotonicity arises not only with respect to negation but also with respect to structural relations between nodes in the trees.

A further reason to use descriptions is the necessity of underspecified representations for natural language expressions. With an increasing number of elements that are ambiguous with respect to their scope, the number of possible readings explodes: a sentence with n such elements may (at least theoretically) have $n!$ readings. Poesio (1996) points out that (1) for example has at least 14400 readings.

(1) A politician can fool most voters on most issues most of the time, but no politician can fool all voters on every single issue all of the time.

This shows clearly that from a practical point of view, a disjunction of several analyses of an expression should be replaced by a single underspecified analysis in order to avoid combinatorial explosion. Furthermore, from a cognitive point of view it does not seem plausible that we process sentences such as (1) by first generating all readings and then testing them. Instead, in some cases, the generation of a single underspecified analysis might be cognitively adequate whereas in other cases, we probably generate a default analysis and, if necessary, perform some backtracking operation. The underspecified representation then marks the point one has to go back to when revising the default analysis. In any case, even from a cognitive point of view the use of underspecified representations is necessary.

Since one description may be true for more than one object, the use of descriptions provides underspecification in a very natural way. With appropriate partial structures, underspecified representations also can be obtained but in a less straight-forward way. In this dissertation, I will therefore use descriptions of trees in order to integrate underspecified representations into a tree-generating formalism.

1.1 Tree Adjoining Grammars and tree descriptions

The *Tree Adjoining Grammar (TAG)* formalism, originally presented in Joshi et al. 1975, is a tree-rewriting grammar formalism developed for natural languages. TAGs were proposed in order to provide on the one hand a formalism that should be powerful enough to handle natural languages, in particular more powerful than context-free grammars. On the other hand, this formalism should stay as close as possible to the set of natural languages and therefore only slightly extend the generative capacity of context-free grammars.

This last condition is captured by the notion of so-called *mildly context-sensitive* grammars. Joshi (1985) claims that grammars that are adequate for natural language structures are mildly context-sensitive, i.e. they allow only limited cross-serial dependencies, they generate only languages of constant growth, and they are polynomially parsable. These three conditions hold for TAGs.

TAGs are structure-based since the elementary objects (i.e. the objects the grammar consists of) are trees. From these elementary trees other trees are derived by operations called *adjunction* and *substitution*. Both operations consist of replacing one single node with a new elementary tree. In contrast to the string rewriting rules of context-free grammars, TAGs allow extended domains of locality, i.e. a single "elementary" structure in the grammar may contain more than one node and its daughters. As a consequence of this, TAGs can be lexicalized (which means that each "elementary" tree in the grammar contains a lexical item), and linguistic relations such as argument selection can be stated in the elementary structures of the grammar.

However, there are some natural language phenomena that cannot be adequately analyzed with TAGs. Therefore several extensions of TAGs were proposed, in particular Multi-Component TAGs (MC-TAGs, see Weir 1988 and Kroch and Joshi 1987) were used to handle certain extraposition phenomena, Description Tree Grammars (D-Tree Grammars, see Rambow et al. 1995) were developed in order to have a uniform complementation operation, and

Unordered Vector Grammars with Dominance Links (UVG-DL, see Rambow 1994a,b) were introduced in order to provide an analysis of German scrambling phenomena.

A common property of these TAG variants is that more than one tree is added in each derivation step. Furthermore, in the case of D-Tree Grammars and UVG-DLs, these trees are connected by *dominance* links. More precisely, a dominance link relates a leaf in one of the trees to the root of another tree. These dominance links are supposed to represent the reflexive and transitive closure of the parent (or immediate dominance) relation. In other words, a dominance link between a leaf u_1 and a root node u_2 signifies that either u_1 and u_2 may be identified with each other or there may be arbitrarily many nodes between u_1 and u_2. In this sense, dominance links signify an underspecification of the parent relation in a tree. This shows that the possibility of underspecification in TAG variants is motivated not only by the necessity of underspecified representations but also by considerations concerning the generative capacity of the grammar.

As already mentioned above, the idea to use tree descriptions (so-called *quasi-trees*) for a definition of TAGs stems from Vijay-Shanker (1992). This approach is motivated by the lack of monotonicity in TAGs which can be remedied by the use of quasi-trees. A logic for this description-based approach is presented in Rogers 1994 and Rogers and Vijay-Shanker 1994.

This shift from the more traditional structure-based TAG formalism dealing with objects to description-based TAG variants dealing with descriptions of objects opens new perspectives. Underspecification is a problem if linguistic information is viewed in terms of a single model. But, as already mentioned above, if linguistic information is viewed in terms of descriptions, then underspecification is expected, since descriptions usually have several models. Therefore, a description-based TAG variant might enable us to obtain underspecified representations for sentences such as (1).

Another TAG variant that has influenced the work presented in this dissertation is the formalism of synchronous TAGs proposed by Shieber and Schabes (1990). The idea of this formalism is to relate two TAGs by a synchronization relation and then to perform derivations in the two TAGs in parallel where simultaneous derivation steps are controlled by the synchronization relation. Shieber and Schabes (1990) propose to use such a grammar in order to account for the syntax-semantics interface. Adopting this idea, I will describe the syntax-semantics interface using two description-based TAG variants related by a synchronization relation.

1.2 Tree descriptions as underspecified representations

In most recent theories of underspecified representations (e.g. Reyle 1993, Bos 1995, Muskens 1995, and Niehren et al. 1997a,b), scope is represented by a partial order. Underspecified analyses for scope ambiguities are then obtained by leaving this ordering relation partially unspecified.

In the case of tree descriptions, the dominance relation is a partial order. (One node u_1 *dominates* a second node u_2, if either $u_1 = u_2$ holds or u_1 has a daughter that dominates u_2.) Therefore, a representation of scope by the dominance relation suggests itself. Consider the noun phrase in (2) (taken from Richter and Sailer 1997). This NP is structurally ambiguous and has two readings.

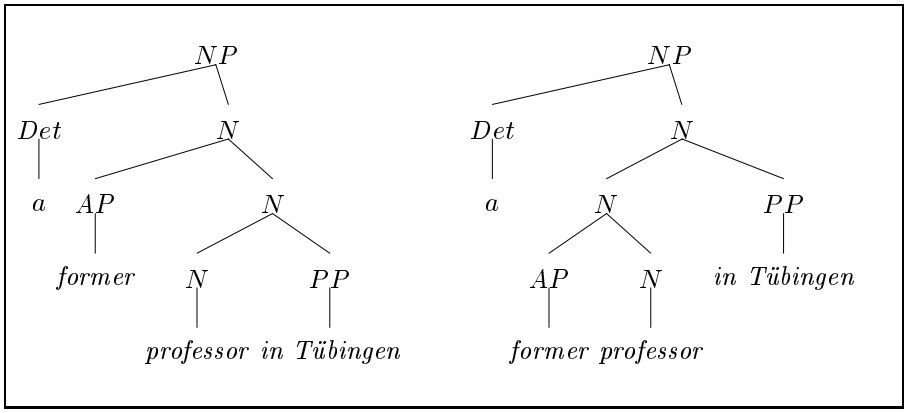


FIGURE 1.1: SYNTACTIC STRUCTURES FOR (2)

(2) a former professor in Tübingen

The two syntactic structures of (2) are shown in Fig. 1.1. Corresponding to these two structures, there are two possible scope orders:

1. either the NP describes a person that was a professor in Tübingen in some former days (this corresponds to the left structure in Fig. 1.1),
2. or the NP describes a person that is in Tübingen and that was a professor in some former days (right structure in Fig. 1.1).

An underspecified representation for the two syntactic structues of (2) should be a tree description that describes all properties that the two trees have in common. These properties can be characterized as follows:

- there is a subtree γ_1 with a root node with label NP, this root node has a left daughter with label Det and a right daughter with label N. The left daughter again has a left daughter which is labelled by the determiner *a*.
- there is a subtree γ_2 containing a root with label N, this root has a left daughter with label AP, and a right daughter with label N. Furthermore, the left daughter again has a daughter with the label *former*.
- there is a subtree γ_3 containing a root with label N, this root has a left daughter with label N and a right daughter with label PP. The right daughter has a daughter with label *in Tübingen*,
- there is a subtree γ_4 containing a root with label N and a single daughter with label *professor*.
- for the relations between these four subtrees, the following holds: the leaf in γ_1 with label N dominates the roots of γ_2 and γ_3, and the leaves in γ_2 and γ_3 with label N both dominate the root of γ_4.

If these properties are to be described in a tree description, then this tree description must correspond to the graphical representation given in Fig. 1.2. To avoid misunderstandings, I

want to emphasize that objects as the one in Fig. 1.2 are not descriptions themselves but they are only pictures of descriptions. Descriptions are formulas in a tree logic and graphical representations illustrate the information entailed by the formula in question. Dotted edges in such pictures represent dominance relations. The fact that the description of one node is depicted left of the description of another node does not always signify linear precedence. The description in Fig. 1.2 for example does not signify that the description of the root of γ_2 is left of the description of the root of γ_3.

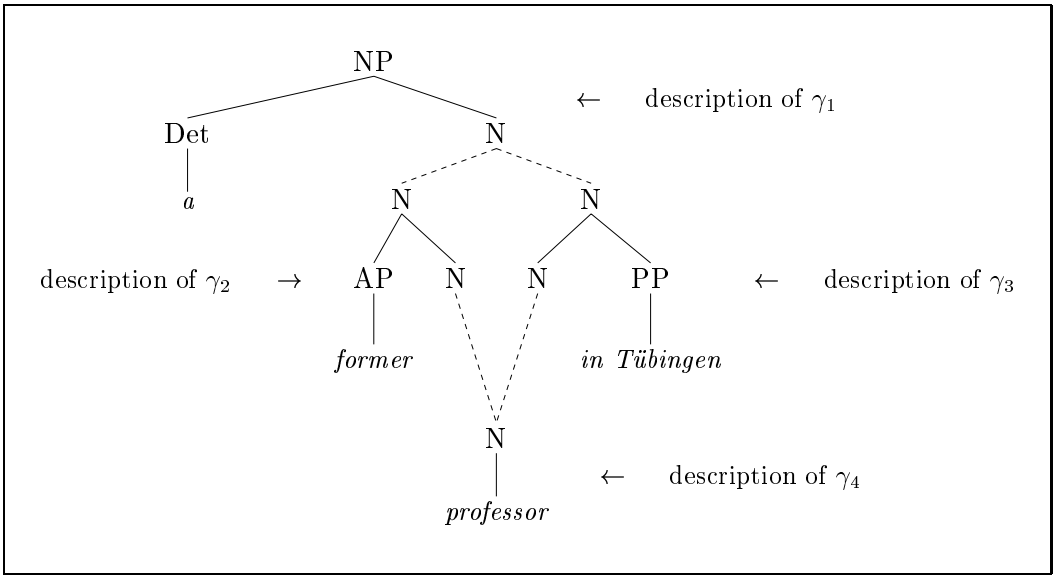


FIGURE 1.2: UNDERSPECIFIED REPRESENTATION FOR (2)

In this tree description, the dominance relation is underspecified: either the leaf with label N in γ_2 dominates the root of γ_3 or the leaf with label N in γ_3 dominates the root of γ_2. In the first case, the syntactic structure shown on the left side of Fig. 1.1 is obtained as a kind of minimal model, and in the scond case the one on the right side is obtained.

These considerations have shown that an underspecification of the dominance relation is necessary in order to obtain suitable underspecified representations for syntactic ambiguities such as (2). However, the underspecification in the above mentioned TAG-variants concerns only the parent relation. Dominance is always completely specified in these approaches, and tree descriptions as in Fig. 1.2 cannot be obtained. Therefore none of these formalisms is suitable to generate underspecified representations of the form depicted in Fig. 1.2.

1.3 Overview of the thesis

This dissertation addresses the problem of integrating underspecified representations of the form shown in Fig. 1.2 into a description-based TAG extension. I will propose a new grammar formalism called *Tree Description Grammars (TDG)*. The advantages of TDGs encompass those of previous extensions of TAGs, and TDGs allows the generation of underspecified

representations for scope ambiguities, i.e. of tree descriptions that represent scope ambiguities as lack of information about the tree structure, rather than as a disjoint list of trees.

An introduction to TAGs and the TAG variants mentioned in Section 1.1 can be found in Chapter 2 of this dissertation. In Chapter 3, a general form of TDGs is introduced, and later, in Chapter 4, a variant of TDGs, *local TDGs* are proposed. Local TDGs are the central formalism of the thesis. Chapter 5 presents a syntax-semantics interface for a fragment of French showing that with local TDGs the desired underspecified representations can be generated. A summary of the main points made in the thesis, and a brief outlook, can be found in Chapter 6.

1.3.1 Tree Description Grammars

TDGs are description-based, i.e. they consist of descriptions (formulas) denoting sets of trees. The underlying tree logic is a quantifier free first order logic similar to the one proposed in Rogers 1994 and Rogers and Vijay-Shanker 1994.

The logic allows the description of relations between node names k_1, k_2 such as parent relation (i.e. immediate dominance) $k_1 \lhd k_2$, dominance $k_1 \lhd^* k_2$, linear precedence $k_1 \prec k_2$ and equality $k_1 \approx k_2$. Furthermore, nodes are supposed to be labelled by terminals or by atomic feature structures. The labeling function is denoted by δ, and for a node name k, $\delta(k) \approx t$ signifies that k has a terminal label t, and $a(\delta(k)) \approx v$ signifies that k is labelled by a feature structure containing the attribute value pair $\langle a, v \rangle$.

For this logic, a syntactic notion of consequence is defined, based on inference rules. This syntactic consequence is decidable, sound and complete.

A presentation of the tree logic can be found in Section 3.1 of Chapter 3.

TDGs consist of tree descriptions, so-called *elementary descriptions*, and a specific *start description*. In an elementary description ψ, some of the node names are marked (those in the set K_ψ). A sample grammar is given in Fig. 1.3.

In contrast to other description-based approaches such as King's logic for HPSG, TDGs are derivation-based: they generate tree descriptions. A formalization of TDGs without a notion of derivation might also be possible but it would be a complex issue involving the definition of a more powerful tree logic than the one used for the derivation-based definition of TDGs.

A derivation in a TDG starts with the start description. In each derivation step, a derived description ϕ_1 and an elementary description ψ are used to obtain a new description ϕ_2. Roughly speaking, ϕ_2 can be viewed as a conjunction of ϕ_1, ψ and new equivalences between node names from ϕ_1 and from ψ, i.e. $\phi_2 = \phi_1 \wedge \psi \wedge k_1 \approx k'_1 \wedge \cdots \wedge k_n \approx k'_n$ where k_i are names from ϕ_1 and k'_i are names from ψ for $1 \leq i \leq n$. This derivation step must be such that

1. for a node name k_ψ in ψ, there is a new equivalence iff k_ψ is either marked (i.e. in K_ψ) or k_ψ is minimal (dominated by no other name, e.g. k_7 in ψ_1 in Fig. 1.3),

2. and the result ϕ_2 must be as "underspecified" as possible, i.e. up to some renaming of node names, ϕ_2 must not entail any other possible result of the derivation step.

E.g. in the grammar in Fig. 1.3, derivations $\phi_S \stackrel{\psi_1}{\Rightarrow} \phi_2$ with $\phi_2 = \phi_S \wedge \psi_1 \wedge k_4 \approx k_7 \wedge k_5 \approx k_{12}$

TDG with start description ϕ_S and elementary descriptions ψ_1, ψ_2:

$\phi_S = k_1 \lhd k_2 \wedge k_1 \lhd k_4 \wedge k_2 \lhd k_3 \wedge k_4 \lhd^* k_5 \wedge k_5 \lhd k_6 \wedge k_2 \prec k_3$
$\wedge cat(\delta(k_1)) \approx NP \wedge cat(\delta(k_2)) \approx Det \wedge \delta(k_3) \approx a \wedge cat(\delta(k_4)) \approx N$
$\wedge cat(\delta(k_5)) \approx N \wedge \delta(k_6) \approx professor$

$\psi_1 = k_7 \lhd^* k_8 \wedge k_8 \lhd k_9 \wedge k_9 \lhd k_{10} \wedge k_8 \lhd k_{11} \wedge k_{11} \lhd^* k_{12} \wedge k_9 \prec k_{11}$
$\wedge cat(\delta(k_7)) \approx N \wedge cat(\delta(k_8)) \approx N \wedge cat(\delta(k_9)) \approx AP \wedge \delta(k_{10}) \approx former$
$\wedge cat(\delta(k_{11})) \approx N \wedge cat(\delta(k_{12})) \approx N$
$K_{\psi_1} = \{k_{12}\}$

$\psi_2 = k_{13} \lhd^* k_{14} \wedge k_{14} \lhd k_{15} \wedge k_{15} \lhd^* k_{16} \wedge k_{14} \lhd k_{17} \wedge k_{17} \lhd k_{18} \wedge k_{15} \prec k_{17}$
$\wedge cat(\delta(k_{13})) \approx N \wedge cat(\delta(k_{14})) \approx N \wedge cat(\delta(k_{15})) \approx N \wedge cat(\delta(k_{16})) \approx N$
$\wedge cat(\delta(k_{17})) \approx PP \wedge \delta(k_{18}) \approx in\ Tübingen$
$K_{\psi_2} = \{k_{16}\}$

Graphical representations:

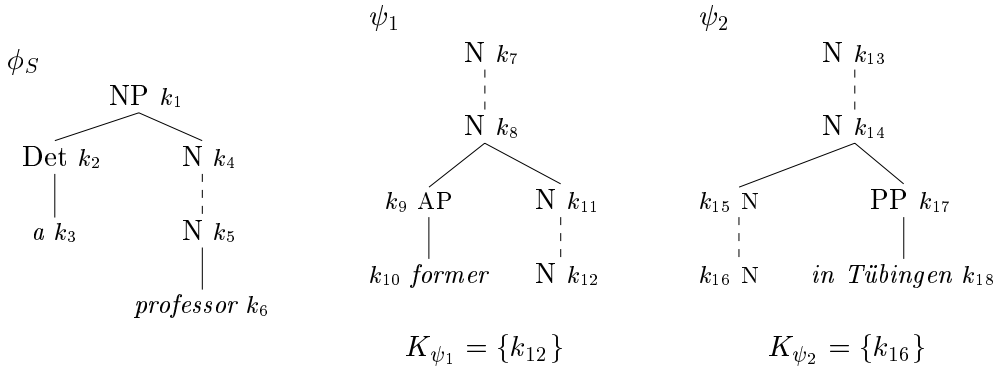


FIGURE 1.3: SAMPLE TDG

and also $\phi_2 \overset{\psi_2}{\Rightarrow} \phi_3$ with $\phi_3 = \phi_2 \wedge \psi_2 \wedge k_4 \approx k_{13} \wedge k_5 \approx k_{16}$ are possible. Graphical representations of ϕ_2 and ϕ_3 are shown in Fig. 1.4. In the second case, ϕ_3 is the only description that can be derived by putting ϕ_2 and ψ_2 together because the result of a derivation step must be maximally underspecified. This condition avoids for example the derivation of a description where $k_4 \approx k_{13}$ and $k_8 \approx k_{16}$ are added as new equivalences when combining ϕ_2 and ψ_2.

TDGs generate descriptions that are true for infinitely many trees. Therefore a notion of *minimal* trees for a description is needed. E.g. the minimal trees of ϕ_3 in Fig. 1.4 should be the two trees in Fig. 1.1. Roughly speaking, a minimal tree γ of a description is defined as a tree that satisfies the description in such a way that each parent relation in the tree is described exactly once in the description. More precisely

(a) all parent relations in the tree are described in the description, and
(b) if two node names in the description are not equivalent and for both there are daughters described, then the two node names do not denote the same node in γ.

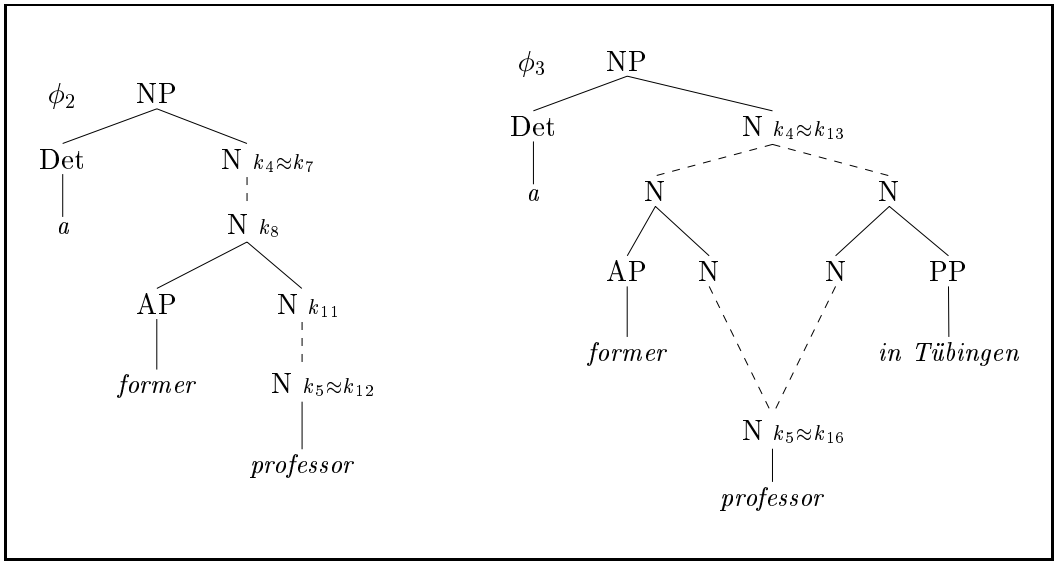


FIGURE 1.4: SAMPLE DERIVATION

The TDG formalism is presented in Section 3.2 of Chapter 3. In Section 3.3 I will give a derivation procedure that takes a derived description and a new elementary description and gives all possible results of a derivation step with these two descriptions. For the special case of lexicalized TDGs the decidability of the word problem will be shown.

TDGs give us a way of introducing underspecification to tree-generating grammars. They offer the advantages of Multi-Component TAGs and of D-Tree Grammars, without losing the main principles of TAGs, namely the extended domain of locality.

1.3.2 Local TDGs

The problem with the general definition of TDGs is that they might be too powerful, since there is no restriction with respect to the choice of equivalent node names in a derivation step.

Therefore, in Chapter 4, I will propose a restriction of TDGs called *local TDGs*. The idea is to restrict TDGs in such a way that the modifications that take place in one derivation step are local in the following sense: the part of the old description that is modified in this derivation step must come from one single elementary description that was added before. The most important restriction is that in one derivation step, all node names in the old description that are used for new node name equivalences added in this step have to be part of one single elementary description. This description is called the *derivation description* of this step. Together with some further conditions, this requirement guarantees that only local modifications are caused by a derivation step. This means that the modification of properties of node names caused by one derivation step concerns only names from one single elementary description that was added before.

As a consequence of this locality, the derivation process in a local TDG can be described by a context-free grammar. With this context-free grammar, semilinearity of the string languages

of local TDGs can be proven by showing letter equivalence to context-free languages. This result indicates that local TDGs are more adequate to handle natural languages than (general) TDGs, since the so-called *Constant Growth Property* (a consequence of semilinearity) holds for the string languages of local TDGs.

The Constant Growth Property is one of the properties Joshi (1985) lists in his characterization of mildly context-sensitive languages. I do not consider the other two properties of MCSGs in this dissertation. Cross-serial dependencies are clearly not limited in local TDGs. With respect to parsing, I do not treat parsing complexity, this is not within the scope of the work presented here.

On the other hand, local TDGs are still powerful enough: I will show that they are more powerful than TAGs, even more powerful than set-local MC-TAGs. The crucial point is that the locality restriction concerns only the derivation of descriptions and not the construction of minimal trees.

As TDGs, local TDGs are also suitable for the generation of underspecified representations. E.g. the descriptions ϕ_2 and ϕ_3 in Fig. 1.4 also can be derived with a local derivation mode. In both derivation steps, $\phi_S \Rightarrow \phi_2$ and $\phi_2 \Rightarrow \phi_3$, the node names k_4 and k_5 that are both part of the start description ϕ_S are used for the new equivalences.

The formalism of local TDGs is presented in Section 4.1, and the semilinearity proof and the comparison to MC-TAGs can be found in Section 4.2 of Chapter 4.

Local TDGs can be distinguished with respect to the form of the descriptions occurring in the grammar. Roughly speaking, the distinction concerns the maximal number of new node name equivalences that might be added in one derivation step. According to this, in Section 4.3 I will define local TDGs of different rank and I will prove a pumping lemma for the string languages generated by local TDGs of a certain rank n. Using this pumping lemma, I will show that local TDGs of rank n generate a true subset of the languages generated by local TDGs of rank $n+1$. In particular, TDGs of rank n can "count up to $2n$". This means that the languages $L_i := \{a_1^k \cdots a_i^k; k \geq 0\}$ are generated up to $i = 2n$ by local TDGs of rank n, whereas for $i > 2n$, L_i cannot be generated by a local TDG of rank n.

Adopting ideas from synchronous TAGs as defined in Shieber and Schabes 1990, I will define synchronous local TDGs in section 4.4. This formalism can be used to describe the relation between two languages that depend on each other in a compositional way but that do not have exactly the same structures. It is later used in Chapter 5 in order to describe the relation between syntax and semantics.

A synchronous pair of local TDGs consists of two local TDGs related by a synchronization relation. This is a relation between elementary descriptions in the first local TDG and elementary descriptions in the second local TDG. Furthermore, for two related descriptions, there is a relation between the node names of the two descriptions which indicates the node names that must be treated in parallel in course of a derivation.

Derivation takes place simultaneously in these two grammars and it is controlled by the synchronization relation: in each derivation step, a single derivation step takes place in each of the two local TDGs. These derivations must be such that

1. the new elementary descriptions added in this step are related by the synchronization relation,
2. the two derivation descriptions are also related to each other, and

3. if new equivalences $k_1 \approx k_2$ and $k_3 \approx k_4$ are added in the two local TDGs, and if k_1 and k_3 are related but k_2 and k_4 are not related, then neither k_2 nor k_4 is related to any node.

With respect to the generative capacity of this formalism, it can be shown that the synchronization with a second local TDG does not increase the expressivity of local TDGs. In other words, the set of languages generated by local TDGs that are part of a pair of synchronous local TDGs is equal to the set of languages generated by single local TDGs.

1.3.3 Linguistic applications

Chapter 5 of the dissertation is concerned with a linguistic application of synchronous local TDGs.

I will present a syntax-semantics interface for French that consists of a local TDG G_{syn} for the syntax, a local TDG G_{sem} for a truth conditional semantics and a synchronization relation between G_{syn} and G_{sem}.

The truth conditional logic is a typed first order λ-calculus. Since local TDGs provide an extended domain of locality, the relation between syntax and semantics is less close than in more traditional Montagovian theories where each phrase structure rule is connected to a semantic operation. Therefore the semantics depends on the syntax in a compositional way but it can be independent from word order variation. Consequently, only functional application and λ-abstraction are necessary as semantic operations, and the trees generated with respect to G_{sem} can be considered as syntactic structures of logical expressions. It is sufficient to define truth conditional interpretations for the strings yielded by these trees.

The syntax-semantics interface covers a fragment of French including proper names, common nouns, verbs, intersective and non-intersective adjectives, generalized quantifiers and restrictive and appositive relative clauses. I will show that this approach gives a way to generate suitable underspecified representations for quantifier scope ambiguities. Contrasting this with other theories proposed for underspecification, we will see that synchronous local TDGs have the advantage that island conditions for quantifier scope do not have to be explicitely stated, but are a consequence of the formalism of local TDGs and of the specific derivation mode for synchronous local TDGs.

Chapter 2

Tree Adjoining Grammars

With respect to the descriptive complexity of theories of grammar developed for natural languages, there are two aspects that must be taken into account. First, such a grammar formalism must be rich enough to provide descriptions for the full range of data observed for natural languages. Second, if possible, the formalism should even predict some restrictions that hold for natural languages, in other word it should embody a description of the nature, and hence limits, of natural language. Head-Driven Phrase Structure Grammars (HPSG) (see Pollard and Sag 1994) for example are powerful enough to deal with natural languages but they do not explain any restrictions on natural language structure. King and Vaillette (1997) show that the logic proposed for HPSG by King (1994) is capable of determining any total recursive set of strings. In contrast to this, Tree Adjoining Grammars (TAG) have been developed with the idea to come as close as possible to a characterization of the class of natural languages. TAGs are a tree generating formalism originally presented by Joshi et al. (1975). Introductions to TAGs can be found in Joshi 1987 and Abeillé 1993.

It has been shown that context-free grammars are not powerful enough to deal with natural languages. Bresnan et al. (1982) for example have argued that a structurally adequate description of cross-serial dependencies in Dutch is not possible with context-free grammars. In other words, the strong generative capacity of context-free grammars is not sufficient to handle natural languages. Shieber (1985) shows that in the case of Swiss German, where cross-serial dependencies require appropriate case-marking, even the weak generative capacity of context-free grammars is not sufficient to handle these phenomena.

The goal of the definition of TAGs is to provide a grammar formalism that is more powerful than context-free grammars and therefore more suitable for natural languages but that only slightly extends context-free grammars. To capture this idea, Joshi (1985) establishes some criteria that such a formalism should satisfy. These criteria are:

- the string languages should be of constant growth,
- the formalism should be polynomially parsable, and
- it should allow only limited cross-serial dependencies.

Formalisms satisfying these conditions are called *mildly context-sensitive grammars*. Joshi argues that grammars that are adequate for natural language structures will be found in the

class of mildly context-sensitive grammars. TAGs and even set-local Multi-Component TAGs are mildly context-sensitive.

In context-free derivation steps, trees of height 1 (i.e. trees where all daughters of the root are leaves) are substituted for leaves. The idea of the TAG formalism is to extend context-free grammars such that instead of trees of height 1, larger trees of arbitrary (finite) height are allowed as elements of the grammar. In a TAG derivation step, one node is replaced by such a larger tree. The main derivation operation of TAGs is the operation *adjunction*. Adjunction consists of replacing one internal node by a tree. It is used to factor recursion from the local domains, i.e. the trees in the grammar.

Apart from the insufficient generative power, a second problem with context-free grammars is that they cannot be lexicalized. Roughly said, a grammar is lexicalized if each element in the grammar (e.g. production, tree, tree description ...) contains a lexical item that is called its anchor. For a context-free grammar, it is of course possible to find a weakly equivalent lexicalized context-free grammar, e.g. the equivalent grammar in Greibach normal form. However, the derivation trees of these two grammars are usually not the same, and therefore the grammars are not strongly equivalent. If a grammar formalism is intended to describe natural languages, then a lexicalization should be possible in the sense that for each grammar of this formalism, a strongly equivalent lexicalized grammar exists (i.e. a grammar generating the same structures).

In order to provide lexicalized TAGs, besides adjunction, a second operation called *substitution* is used. Substitution consists of replacing a leaf by a tree. It does not extend the generative capacity. TAGs with adjunction and substitution can be lexicalized because of their larger domains of locality and because, with the additional substitution operation, unique anchors are possible for elementary structures. Furthermore, cooccurrence restrictions, e.g. filler gap dependencies or subcategorization frames can be locally described in single elementary trees, i.e. trees belonging to the grammar.

In this chapter I will describe TAGs and those extensions of TAGs that have influenced the development of Tree Description Grammars as presented in the following chapters. Before coming to the introduction of TAGs, some basic definitions are listed in the first section. These definitions are probably known to most readers (e.g. the definition of a context-free grammar). In Section 2.2 I will then present the TAG formalism with adjunction and substitution as derivation operations. Following Abeillé 1993, I will mention some linguistic principles underlying TAGs and sketch briefly how to describe natural languages with TAGs. In the next two sections, I will present some TAG-variants, synchronous TAGs (defined in Shieber and Schabes 1990) and Multi-Component TAGs, as proposed in Joshi 1987 and Weir 1988. Afterwards, I will mention some recent proposals to view TAGs as defining constraints on well-formed structures rather than generating a tree set. Besides some general considerations concerning constraints I will outline the formalism of D-Tree grammars (DTG) as presented in Rambow et al. 1995 and Unordered Vector Grammars with dominance links (UVG-DL), proposed by Rambow (1994a,b). As a conclusion, in the last section, I will summarize the criteria a grammar formalism should satisfy if it is supposed to preserve the attractive properties of TAG-related formalisms and to allow underspecified representations.

2.1 Basic definitions

In this part I will give some basic definitions from the area of formal languages that are used in the course of the dissertation. Most of the notions introduced here are generally known (see for example Hopcroft and Ullman 1979, 1994 for these basic definitions).

Definition 2.1 (Alphabet, word) *An alphabet is a nonempty finite set X. A string $x_1 dots x_n$ with $n \geq 1$ and $x_i \in X$ for $1 \leq i \leq n$ is called a nonempty word on the alphabet X. X^+ is defined as the set of all nonempty words on X.*
A new element $\epsilon \notin X^+$ is added: $X^* := X^+ \cup \{\epsilon\}$. For each $w \in X^+$ concatenation of w and ϵ is defined as follows: $w\epsilon := \epsilon w := w$. ϵ is called the empty word, and each $w \in X^*$ is called a word on X.

With respect to the concatenation operation on strings, X^+ is a semigroup, and X^* is a monoid with neutral element ϵ.

Definition 2.2 (Homomorphism) *For two alphabets X and Y, a function $\varphi : X^* \to Y^*$ is a homomorphism iff for all $v, w \in X^*$: $\varphi(vw) = \varphi(v)\varphi(w)$.*

Definition 2.3 (Prefix, suffix, substring) *Let X be an alphabet. For a word $w \in X^*$ a word $u \in X^*$ is*

- *a prefix of w iff there is a $v \in X^*$ such that $w = uv$.*
- *a suffix of w iff there is a $v \in X^*$ such that $w = vu$.*
- *a substring of w iff there are $v_1, v_2 \in X^*$ such that $w = v_1 u v_2$.*

Definition 2.4 (Length of a word, a-length) *Let X be an alphabet, $w \in X^*$.*

1. *The length of w, $|w|$, is defined as follows: if $w = \epsilon$, then $|w| = 0$. If $w = xw'$ for some $x \in X$, then $|w| = 1 + |w'|$.*
2. *For each $a \in X$, the a-length of w, $|w|_a$, is defined as follows: if $w = \epsilon$, then $|w|_a = 0$. If $w = xw'$ for some $x \in X$, then: if $x = a$, then $|w|_a = 1 + |w'|_a$, else $|w|_a = |w'|_a$.*

In other words, the a-length $|w|_a$ of a word w for a certain symbol a is the number of occurrences of a in w.

Definition 2.5 (Language) *A set L is called a language iff there is an alphabet X such that $L \subseteq X^*$.*

Definition 2.6 (Context-free grammar) *A context-free grammar (CFG) is a tuple $G = (N, T, P, S)$ such that*

1. *N and T are disjoint alphabets, the nonterminals and terminals of G.*
2. *$P \subseteq N \times (N \cup T)^*$ is the set of productions. A production (A, α) is usually written $A \to \alpha$.*
3. *$S \in N$ is the start symbol.*

Definition 2.7 (Language of a CFG) *Let $G = (N, T, P, S)$ be a CFG. Then the language $L(G)$ generated by G is the set $\{w \in T^* \mid S \stackrel{*}{\Rightarrow} w\}$ where*

1. *for* $w_1, w_2 \in (N \cup T)^*$: $w_1 \Rightarrow w_2$, *iff there is a* $A \to \alpha \in P$ *and there are* v, u *such that* $w_1 = vAu$ *and* $w_2 = v\alpha u$,
2. *and* $\overset{*}{\Rightarrow}$ *is the reflexive transitive closure of* $\Rightarrow$ *on* $(N \cup T)^*$.

A language is called context-free iff it is generated by a CFG.

2.2 TAG formalism

2.2.1 Adjunction and substitution

In order to give a precise description of the tree operations adjunction and substitution used in TAGs, a definition of trees, more precisely of finite labelled trees, is necessary. Here I will not use the notion of tree domains as Vijay-Shanker (1987) does for example. Instead I will look at trees as sets of nodes structured by binary relations *parent*, *dominance* and *linear precedence*. This concept of trees corresponds very much to the intuitive picture of a tree. In Chapter 3 these structures also will serve as models for the tree logic for Tree Description Grammars where variables denote nodes and there are constants for the binary relations on the nodes, e.g. "$\lhd$" for the parent relation.

Definition 2.8 (Finite labelled tree) *Let* N *and* T *be disjoint alphabets. A tuple* $(\mathcal{U}, u_0, \mathcal{P}, \mathcal{D}, \mathcal{L}, \mu)$ *is a finite labelled tree with nonterminals* N *and terminals* T *iff*

1. $\mathcal{U}$ *is a finite set with* $\mathcal{U} \cap (N \cup T) = \emptyset$, *the set of nodes.*
2. $u_0 \in \mathcal{U}$ *is a special node, the root.*
3. $\mathcal{P}, \mathcal{L}, \mathcal{D} \in \mathcal{U} \times \mathcal{U}$, *such that for all* $w, x, y, z \in \mathcal{U}$:
   (a) $\mathcal{P}$ *is irreflexive, and if* $x \neq u_0$, *then there is exactly one* $v \in \mathcal{U}$ *with* $\langle v, x\rangle \in \mathcal{P}$.
   (b) $\mathcal{D}$ *is the reflexive transitive closure of* $\mathcal{P}$, *and* $\mathcal{D}$ *is antisymmetric.*
   (c) $\mathcal{L}$ *is transitive.*
   (d) $\langle x, y\rangle \in \mathcal{D}$ *or* $\langle y, x\rangle \in \mathcal{D}$ *or* $\langle x, y\rangle \in \mathcal{L}$ *or* $\langle y, x\rangle \in \mathcal{L}$.
   (e) *if* $\langle x, y\rangle \in \mathcal{L}$, *then* $\langle x, y\rangle \notin \mathcal{D}$ *and* $\langle y, x\rangle \notin \mathcal{D}$.
   (f) *if* $\langle x, y\rangle \in \mathcal{L}$ *and* $\langle x, w\rangle, \langle y, z\rangle \in \mathcal{D}$, *then* $\langle w, z\rangle \in \mathcal{L}$.
4. $\mu : \mathcal{U} \to T \cup N \cup \{\epsilon\}$ *is a function, such that for all* $u \in \mathcal{U}$: $\mu(u) \in N$ *if there is a* $u' \in \mathcal{U}$ *with* $\langle u, u'\rangle \in \mathcal{P}$ *(*u *is no leaf).*

A $u \in \mathcal{U}$, *such that there is no* $u' \in \mathcal{U}$ *with* $\langle u, u'\rangle \in \mathcal{P}$, *is called a leaf.*
A $u \in \mathcal{U}$ *that is not a leaf is called an internal node.*

With *3.(a)*, each node except the root node has a single parent. Together with the irreflexivity of $\mathcal{P}$ and the antisymmetry of $\mathcal{D}$, this signifies furthermore that the root has no parent and that each node is dominated by the root node, i.e. for all $x \in \mathcal{U}$, $\langle u_0, x\rangle \in \mathcal{D}$ holds: for $\mathcal{U} = \{u_0\}$ this is obvious. Suppose that $|\mathcal{U}| \geq 2$ and $u_1 \in \mathcal{U} \setminus \{u_0\}$. Let $u_2, \ldots, u_n$, $n \geq 2$ be such that $\langle u_n, u_{n-1}\rangle, \ldots, \langle u_2, u_1\rangle \in \mathcal{P}$. Then for $1 \leq i < j \leq n$, $u_i \neq u_j$ holds because $\mathcal{P}$ is irreflexive and $\mathcal{D}$ is antisymmetric. Therefore, since $\mathcal{U}$ is finite, there must be an $u \in \mathcal{U}$ with $\langle u, u_1\rangle \in \mathcal{D}$ such that u has no parent and consequently (with *3.(a)*) $u = u_0$.

3.(a) requires the parent relation $\mathcal{P}$ to be an irreflexive relation. Furthermore, because of *3.(b)*, the relation $\mathcal{P}$ is asymmetric: suppose that $\langle x, y\rangle \in \mathcal{P}$ and $\langle y, x\rangle \in \mathcal{P}$ hold. With

3.(b) $\mathcal{P} \subset \mathcal{D}$ and therefore $\langle x, y\rangle, \langle y, x\rangle \in \mathcal{D}$. Then $x = y$ must hold since $\mathcal{D}$ is antisymmetric. This is a contradiction to the irreflexivity of $\mathcal{P}$. Intransitivity of $\mathcal{P}$ also can be shown: let $\langle x, y\rangle, \langle y, z\rangle \in \mathcal{P}$. Then x, y, z must be pairwise different and (since u_0 has no parent), $y \neq u_0$ and $z \neq u_0$. Therefore (with *3.(a)*) $\langle x, z\rangle \notin \mathcal{P}$.

Linear precedence $\mathcal{L}$ is transitive. With *3.(b)* and *3.(e)*, $\mathcal{L}$ also is an irreflexive relation: suppose that $\langle x, y\rangle \in \mathcal{L}$ and $x = y$. Then (since $\mathcal{D}$ is reflexive) $\langle x, y\rangle \in \mathcal{D}$ holds which is a contradiction to *3.(e)*. Furthermore, $\mathcal{L}$ is also asymmetric: suppose that $\langle x, y\rangle, \langle y, x\rangle \in \mathcal{L}$. Since $\mathcal{L}$ is transitive, $\langle x, x\rangle \in \mathcal{L}$ which is a contradiction to the irreflexivity of $\mathcal{L}$.

The intuitive meaning of the axioms *3.(d)* to *3.(f)* is the following: *3.(d)* requires every pair of nodes to be related either by domination or by linear precedence. This is sometimes called the *exhaustiveness* property. *3.(e)* together with asymmetry of $\mathcal{L}$ signifies that these relations are mutually exclusive. This property is sometimes referred to as *exclusiveness*. *3.(f)*, the so-called *inheritance* or *non-tangling* property requires that the entire subtrees rooted at nodes related by linear precedence are also related by linear precedence.

Henceforwad, for a tree γ, the symbols $\mathcal{U}_\gamma, u_0^\gamma, \mathcal{P}_\gamma, \mathcal{D}_\gamma, \mathcal{L}_\gamma$ and μ_γ denote the nodes, the root, the parent, dominance and linear precedence relation and the labeling function of γ respectively. In some cases, the nonterminals N and terminals T of a tree are not explicitely mentioned.

If all leaves in a finite labelled tree have terminal labels, this tree is called a *terminal* tree, and then the string consisting of these terminals read from left to right is called the *yield* of the tree.

Definition 2.9 (Terminal tree, yield) *Let γ be a finite labelled tree with nonterminals N and terminals T.*

1. *γ is called a terminal tree iff for all leaves u in γ, $\mu_\gamma(u) \in T \cup \{\epsilon\}$.*
2. *Let γ be a terminal tree with n leaves ($n \geq 1$). Let $u_1, \ldots, u_n$ be the leaves in γ such that $\langle u_i, u_{i+1}\rangle \in \mathcal{L}_\gamma$ for $1 \leq i < n$.*
   Then $yield(\gamma) := \mu(u_1) \ldots \mu(u_n)$ is the yield of γ.

With respect to TAGs, two kinds of trees are distinguished: on the one hand, finite labelled trees as defined above can occur in a TAG. In this case such a tree is called an *initial* tree of the grammar.[1] On the other hand, a TAG contains trees whith a unique leaf in the tree that is marked as a so-called *foot node* and that has the same label as the root of the tree. Such trees will be called *foot-marked*, and a foot-marked tree occurring in a TAG is a so-called *auxiliary tree*.[2]

Definition 2.10 (Foot-marked trees, foot node) *A pair $\langle \gamma, u_f^\gamma\rangle$ is called a foot-marked tree iff*

1. *γ is a finite labelled tree with nonterminals N and terminals T, and*
2. *u_f^γ is a leaf in γ with $\mu_\gamma(u_f^\gamma) \in N$ and $\mu_\gamma(u_0^\gamma) = \mu_\gamma(u_f^\gamma)$.*

[1] Originally, in the basic definition of TAGs, all leaves in initial trees are labelled by terminals, i.e. initial trees are terminal trees. But I will use an extended notion of initial trees allowing leaves with nonterminal labels. In the course of a derivation, other initial trees must be substituted for these nodes (substitution operation). Therefore such nodes are called *substitution nodes*.

[2] In other definitions of TAGs, there is no distinction between foot-marked trees in general and auxiliary trees as particular trees of a grammar, and therefore the term *foot-marked* is not used. However I think this notion necessary to give an exact definition of TAGs.

u_f^γ is called the foot node of γ.

In the following, the foot node will usually not be explicitely mentioned, i.e. γ will be written instead of $\langle \gamma, u_f^\gamma \rangle$ if it is clear that γ is a foot-marked tree. Furthermore, if γ is a foot-marked tree, then the symbol u_f^γ denotes the foot node.

The set of nodes between the root and the foot node of a foot-marked tree, i.e. the path from the root to the foot, is called the *spine*. This path contains all nodes that dominate the foot node.

Definition 2.11 (Spine) *Let β be a foot-marked tree with foot node u_f^β. The spine of β is the set $U_s(\beta) = \{u_s \in \mathcal{U}_\beta \mid \langle u_s, u_f^\beta \rangle \in \mathcal{D}_\beta\}$.*

Next the *adjunction* operation will be defined, an operation on trees that consists of replacing a node in a finite labelled tree by a foot-marked tree (see Fig. 2.1). Roughly said, if a foot-marked tree β is adjoined to a tree γ at an internal node u, then the subtree $\gamma(2)$ with root u is replaced by β and afterwards $\gamma(2)$ is inserted under the foot node of β. To render adjunction possible, the node where adjunction takes place and the root and the foot of the foot-marked tree must be labelled by the same nonterminal.

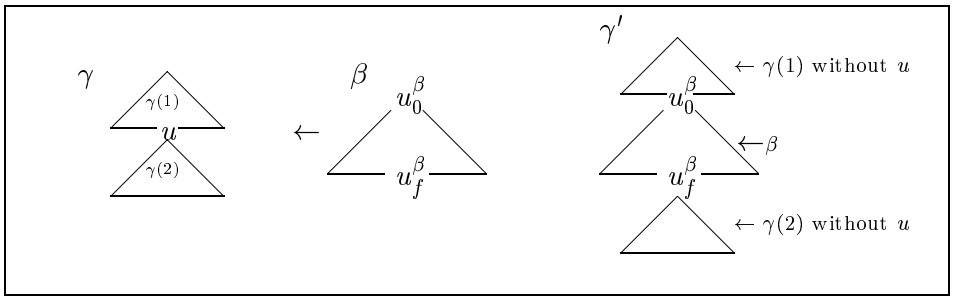


FIGURE 2.1: ADJUNCTION

Definition 2.12 (Adjunction) *Let γ be a finite labelled tree and β a foot-marked tree such that $\mathcal{U}_\gamma \cap \mathcal{U}_\beta = \emptyset$ holds. β can be adjoined to γ at the node $u \in \mathcal{U}_\gamma$, iff u is an internal node with $\mu_\gamma(u) = \mu_\beta(u_0^\beta)$. The result of this adjunction is the finite labelled tree $\gamma' = (\mathcal{U}', u_0', \mathcal{P}', \mathcal{D}', \mathcal{L}', \mu')$ with*

- $\mathcal{U}' = (\mathcal{U}_\gamma \setminus \{u\}) \cup \mathcal{U}_\beta$
- *For all $u_1, u_2 \in \mathcal{U}'$: $\langle u_1, u_2 \rangle \in \mathcal{P}'$*
  iff either $\langle u_1, u_2 \rangle \in \mathcal{P}_\gamma \cup \mathcal{P}_\beta$,
  or $\langle u_1, u \rangle \in \mathcal{P}_\gamma$ and $u_2 = u_0^\beta$,
  or $\langle u, u_2 \rangle \in \mathcal{P}_\gamma$ and $u_1 = u_f^\beta$.
- *For all $u_1, u_2 \in \mathcal{U}'$:*
  if $\langle u_1, u_2 \rangle \in \mathcal{L}_\gamma \cup \mathcal{L}_\beta$ then $\langle u_1, u_2 \rangle \in \mathcal{L}'$,
  if $\langle u_1, u \rangle \in \mathcal{L}_\gamma$ then $\langle u_1, u_0^\beta \rangle \in \mathcal{L}'$,

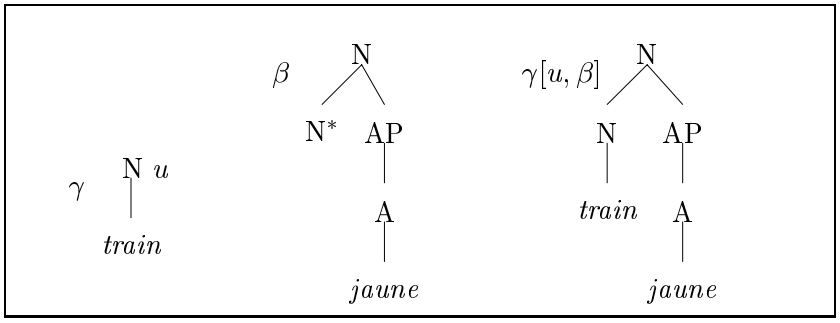


FIGURE 2.2: ANALYSIS OF *train jaune* (*'yellow train'*)

and if $\langle u, u_2\rangle \in \mathcal{L}_\gamma$ *then* $\langle u_0^\beta, u_2\rangle \in \mathcal{L}'$.

- *For all* $u_\gamma \in \mathcal{U}_\gamma \setminus \{u\}$: $\mu'(u_\gamma) = \mu_\gamma(u_\gamma)$.
  For all $u_\beta \in \mathcal{U}_\beta$: $\mu'(u_\beta) = \mu_\beta(u_\beta)$.

Notation: $\gamma' = \gamma[u, \beta]$.

Adjunction is the main derivation operation used for Tree Adjoining Grammars. It gives us a way of factoring recursion, e.g. to add modifiers. The modification of a noun phrase by a nominal modifier for example results in a new noun phrase that can again be modified by a nominal modifier (see Fig. 2.2). In the figures, foot nodes are marked by an asterisk in order to distinguish them from substitution nodes (other leaves with nonterminal labels).

Besides adjunction, a second operation is used in TAGs, namely *substitution*. As adjunction, substitution also consists of replacing one node in a finite labelled tree by a tree. But in contrast to adjunction, this node must be a substitution node, i.e. a leaf, and the tree must be a finite labelled tree.

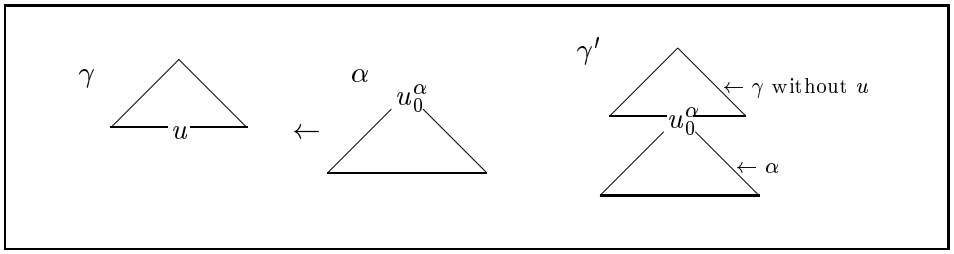


FIGURE 2.3: SUBSTITUTION

Definition 2.13 (Substitution) *Let* γ *and* α *be finite labelled trees such that* $\mathcal{U}_\gamma \cap \mathcal{U}_\alpha = \emptyset$. α *can be substituted for a node* $u \in \mathcal{U}_\gamma$, *iff* u *is a leaf with* $\mu_\gamma(u) = \mu_\alpha(u_0^\alpha)$. *The result of this operation is the finite labelled tree* $\gamma' = (\mathcal{U}', u_0', \mathcal{P}', \mathcal{D}', \mathcal{L}', \mu')$ *with*

- $\mathcal{U}' = (\mathcal{U}_\gamma \setminus \{u\}) \cup \mathcal{U}_\alpha$
- *For all* $u_1, u_2 \in \mathcal{U}'$: $\langle u_1, u_2\rangle \in \mathcal{P}'$
  iff either $\langle u_1, u_2\rangle \in \mathcal{P}_\gamma \cup \mathcal{P}_\alpha$,

or $\langle u_1, u\rangle \in \mathcal{P}_\gamma$ *and* $u_2 = u_0^\alpha$.

- *For all* $u_1, u_2 \in \mathcal{U}'$:

  if $\langle u_1, u_2\rangle \in \mathcal{L}_\gamma \cup \mathcal{L}_\beta$ *then* $\langle u_1, u_2\rangle \in \mathcal{L}'$,
  if $\langle u_1, u\rangle \in \mathcal{L}_\gamma$ *then* $\langle u_1, u_0^\beta\rangle \in \mathcal{L}'$,
  and if $\langle u, u_2\rangle \in \mathcal{L}_\gamma$ *then* $\langle u_0^\beta, u_2\rangle \in \mathcal{L}'$.

- *For all* $u_\gamma \in \mathcal{U}_\gamma \setminus \{u\}$: $\mu'(u_\gamma) = \mu_\gamma(u_\gamma)$.
  For all $u_\alpha \in \mathcal{U}_\alpha$: $\mu'(u_\alpha) = \mu_\alpha(u_\alpha)$.

Notation: $\gamma' = \gamma[u, \alpha]$.

The idea of substitution is that there is a slot – the substitution node – that must be filled in order to obtain a correct structure. Argument slots in subcategorization frames are therefore usually substitution nodes (see Fig. 2.4). This reflects the obligatory character of arguments and the fact that they must be added only once.

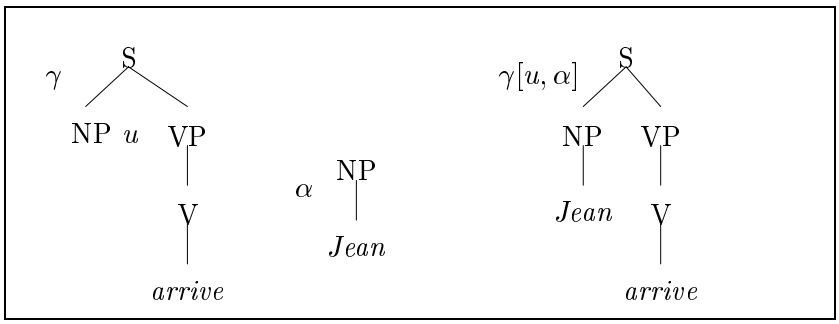


FIGURE 2.4: ANALYSIS OF *Jean arrive* (*'Jean arrives'*)

As we will see later, there are some exceptions from this principle of realizing argument slots by using substitution nodes, e.g. complement clauses.

The notations for adjunction and substitution are the same. This is no problem since the nature of the operation is uniquely determined by the second tree: if it is a foot-marked tree, the operation is an adjunction, and if it is a finite labelled tree, the operation is a substitution.

2.2.2 Grammar formalism

Using the definitions above, Tree Adjoining Grammars can be defined. Mainly, the idea of TAGs is that there are certain trees that already represent complete argument structures, the set of initial trees of a TAG (e.g. the tree γ in Fig. 2.4). Additionally from these initial trees other trees can be derived by adjunction or substitution.

TAGs consist of a set I of finite labelled trees, called the *initial* trees, a set A of foot-marked trees, called the *auxiliary* trees and alphabets T and N of terminals and nonterminals that are common for all trees in the grammar. A tree that is part of a TAG, i.e. either initial or auxiliary, is called an *elementary* tree. Some definitions of TAGs also include a start symbol (see Vijay-Shanker 1987). But I will use a simpler definition of TAGs without a start symbol.

Derivation in TAGs starts with an initial tree. In each derivation step, a new elementary tree is added by adjunction or substitution to the tree that has been already derived. In this way, a set of trees can be generated.

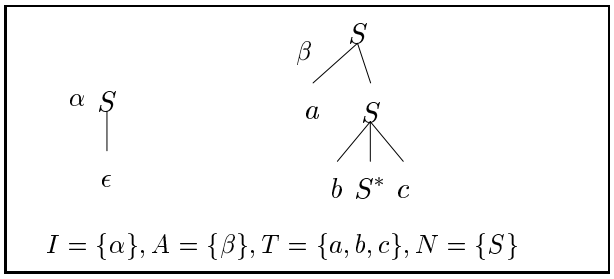


FIGURE 2.5: SAMPLE TAG WITHOUT ADJUNCTION CONSTRAINTS

Simple TAGs as sketched above are more powerful than context-free grammars, but this extra power is quite limited. A TAG generating the language $L = \{wc^n \mid n \geq 0, |w|_a = |w|_b = n$ and for each suffix u of $w : |u|_a \leq |u|_b\}$ is given in Fig. 2.5. This language is not context-free for the following reason: $L_3 = \{a^n b^n c^n \mid n \geq 0\}$ is the intersection of L with the regular language $\{a^l b^m c^n \mid l \geq 0, m \geq 0, n \geq 0\}$. Since L_3 is not context-free (this can be shown with the pumping lemma for context-free languages), and since context-free languages are closed with respect to intersection with regular languages, L cannot be context-free.

However, the language L_3 cannot be generated by a simple TAG without adjunction constraints: clearly, for a TAG generating L, each elementary tree must contain equal numbers of a's, b's and c's. Furthermore, in each case, the a's must precede the b's and the b's must precede the c's. Then it is easy to see that a mixing of either the a's and b's or the b's and c's cannot be avoided. The copy-language $\{ww \mid w \in \{a, b\}^*\}$ also cannot be generated by such a TAG. Therefore it is desirable to have the possibility of further restricting the adjunction operation. To allow this, local constraints on adjunction have been introduced. There are several forms of adjunction constraints that have been proposed (see for example Joshi 1985, 1987). The form of local constraints presented here was first proposed in Joshi 1987, and the formalization chosen here is very similar to the one in Vijay-Shanker 1987.

There are two aspects of possible adjunctions that are specified by adjunction constraints: for each node u

1. the set of auxiliary trees that are allowed to be adjoined at u is restricted, and
2. the constraints specify whether adjunction at u is obligatory or not.

The first of these two aspects is expressed by some function C_γ for each tree γ. For each node u in γ, C_γ maps u to the set of auxiliary trees that are allowed to be adjoined at u. The second constraint is given by a function O_γ mapping from the nodes of γ to $\{0, 1\}$. For each node u in γ, $O_\gamma(u) = 0$ signifies that adjunction is not obligatory for u whereas $O_\gamma(u) = 1$ means that adjunction is obligatory for u.

With C and O the three kinds of adjunction constraints for nodes that are mentioned in Joshi 1987 can be distinguished: nodes where no adjunction is possible (*null adjunction*

or NA-constraint), nodes where adjunction is possible but not obligatory for a certain set of auxiliary trees (*selective adjunction* or SA-constraint), and nodes where adjunction is obligatory (*obligatory adjunction* or OA-constraint). C is a restriction of the adjunction operation whereas O is a restriction on the set of derived trees, the tree language. Trees in the tree language must not contain any OA-constraints, i.e. for all nodes u in a tree γ in the tree language, $O_\gamma(u) = 0$ must hold.

Definition 2.14 (Tree Adjoining Grammar) *A Tree Adjoining Grammar (TAG) is a tuple $\langle N, T, I, A\rangle$ such that*

1. *N and T are disjoint alphabets, the nonterminals and the terminals.*
2. *I is a finite set of triples $\langle \alpha, C_\alpha, O_\alpha\rangle$ where α is a finite labelled tree with nonterminals N and terminals T. $C_\alpha : \mathcal{U}_\alpha \to Pow(A)$, and $O_\alpha : \mathcal{U}_\alpha \to \{0,1\}$ are functions.*
   ($Pow(A)$ is a notation for the set of all subsets of A.)
3. *A is a finite set of triples $\langle \beta, C_\beta, O_\beta\rangle$ where β is a foot-marked tree with nonterminals N and terminals T. $C_\beta : \mathcal{U}_\beta \to Pow(A)$, and $O_\beta : \mathcal{U}_\beta \to \{0,1\}$ are functions.*

Each $\langle \alpha, C_\alpha, O_\alpha\rangle \in I$ is called an initial tree of G, each $\langle \beta, C_\beta, O_\beta\rangle \in A$ is called an auxiliary tree in G, and each $\langle \gamma, C_\gamma, O_\gamma\rangle \in I \cup A$ is called an elementary tree of G.

For initial trees, the symbols $\alpha, \alpha_1, \alpha_2, \ldots$ are usually used whereas $\beta, \beta_1, \beta_2, \ldots$ in general stand for auxiliary trees.

As already mentioned, the derivation operations in TAGs are adjunction and substitution, where adjunction to a tree γ is restricted by C_γ. To avoid some technical problems when defining the derivation in TAGs, the notion of isomorphic trees is needed. A tree isomorphism is simply a node renaming.

Definition 2.15 (Isomorphic trees) *Let $G = \langle N, T, I, A\rangle$ be a TAG, and let $\langle \gamma_1, C_{\gamma_1}, O_{\gamma_1}\rangle$ and $\langle \gamma_2, C_{\gamma_2}, O_{\gamma_2}\rangle$ be two triples where γ_1 and γ_2 are either both finite labelled trees or both foot-marked trees and $O_{\gamma_1} : \mathcal{U}_{\gamma_1} \to \{0,1\}$, $O_{\gamma_2} : \mathcal{U}_{\gamma_2} \to \{0,1\}$, $C_{\gamma_1} : \mathcal{U}_{\gamma_1} \to Pow(A)$ and $C_{\gamma_2} : \mathcal{U}_{\gamma_2} \to Pow(A)$ are functions.*
$\langle \gamma_1, C_{\gamma_1}, O_{\gamma_1}\rangle$ and $\langle \gamma_2, C_{\gamma_2}, O_{\gamma_2}\rangle$ are isomorphic (written $\langle \gamma_1, C_{\gamma_1}, O_{\gamma_1}\rangle \approx \langle \gamma_2, C_{\gamma_2}, O_{\gamma_2}\rangle$) iff there is a bijection $f : \mathcal{U}_{\gamma_1} \to \mathcal{U}_{\gamma_2}$ such that

1. *For all $u_1, u_2 \in \mathcal{U}_{\gamma_1}$ and for all $\mathcal{R} \in \{\mathcal{P}, \mathcal{D}, \mathcal{L}\}$:*
   $\langle u_1, u_2\rangle \in \mathcal{R}_{\gamma_1}$ iff $\langle f(u_1), f(u_2)\rangle \in \mathcal{R}_{\gamma_2}$.
2. *For all $u \in \mathcal{U}_{\gamma_1}$: $\mu_{\gamma_2}(f(u)) = \mu_{\gamma_1}(u)$, $C_{\gamma_2}(f(u)) = C_{\gamma_1}(u)$ and $O_{\gamma_2}(f(u)) = O_{\gamma_1}(u)$.*
3. *If γ_1 and γ_2 are foot-marked, then $f(u_f^{\gamma_1}) = u_f^{\gamma_2}$.*

f is then called a tree isomorphism.

Henceforward, I will often omit the two functions C_γ and O_γ and write γ instead of $\langle \gamma, C_\gamma, O_\gamma\rangle$.

Definition 2.16 (Derivation in a TAG) *Let $G = \langle N, T, I, A\rangle$ be a Tree Adjoining Grammar, and let γ_1 be a finite labelled tree (with C_{γ_1} and O_{γ_1}).*

1. *For each finite labelled tree γ_2 (with C_{γ_2} and O_{γ_2}) and each γ_0 such that there is a $\gamma_e \in I \cup A$ with $\gamma_0 \approx \gamma_e$:*
   γ_2 can be derived from γ_1 in one step using γ_0 (written $\gamma_1 \overset{\gamma_0}{\Rightarrow} \gamma_2$) iff there is a $u \in \mathcal{U}_{\gamma_1}$ such that

(a) $\gamma_2 = \gamma_1[u, \gamma_0]$,
(b) $C_{\gamma_2}(u') = C_{\gamma_1}(u')$ *and* $O_{\gamma_2}(u') = O_{\gamma_1}(u')$ *for all* $u' \in \mathcal{U}_{\gamma_1} \setminus \{u\}$,
(c) $C_{\gamma_2}(u') = C_{\gamma_0}(u')$ *and* $O_{\gamma_2}(u') = O_{\gamma_0}(u')$ *for all* $u' \in \mathcal{U}_{\gamma_0}$,
(d) *and if* $\gamma_e \in A$, *then* $\gamma_e \in C_{\gamma_1}(u)$.

2. *A finite labelled tree* γ_2 *can be derived from* γ_1 *in one step (written* $\gamma_1 \Rightarrow \gamma_2$*) iff there is a* γ_0 *such that* $\gamma_1 \overset{\gamma_0}{\Rightarrow} \gamma_2$.
3. *A finite labelled tree* γ_n *can be derived from* γ_1 *in* n *steps (written* $\gamma_1 \overset{n}{\Rightarrow} \gamma_2$*) for* $n \geq 1$ *iff there is a tree* γ_{n-1} *that can be derived in* $n-1$ *step from* γ_1 *such that* $\gamma_{n-1} \Rightarrow \gamma_n$.
4. *A finite labelled tree* γ_2 *can be derived from* γ_1 *(written* $\gamma_1 \overset{*}{\Rightarrow} \gamma_2$*) iff either* $\gamma_1 = \gamma_2$ *or there is a* n *with* $\gamma_1 \overset{n}{\Rightarrow} \gamma_2$. *(In other words,* $\overset{*}{\Rightarrow}$ *is the reflexive and transitive closure of* $\Rightarrow$.*)*

With respect to the language of a TAG, the set of derived trees (*tree language*) and the set of strings yielded by these trees (*string language*) must be distinguished. The trees in the tree language must be such that their leaves are all labelled by terminals or ϵ and that for none of the nodes adjunction is obligatory.

Definition 2.17 (Tree language, string language) *Let* $G = \langle N, T, I, A \rangle$ *be a Tree Adjoining Grammar.*

(i) *The tree language of* G *is the set*
$L_T(G) := \{\gamma \mid \gamma$ *is a terminal tree, there is an* $\alpha \in I$ *with* $\alpha \overset{*}{\Rightarrow} \gamma$, *and* $O_\gamma(u) = 0$ *holds for all* $u \in \mathcal{U}_\gamma\}$.
(ii) *The string language of* G, $L_S(G)$, *is the set*
$L_S(G) := \{w \mid$ *there is a* $\gamma \in L_T(G)$ *such that* $w = yield(\gamma)\}$.
(iii) *A language* $L \subseteq T^*$ *is a Tree Adjoining Language (TAL), iff there is a TAG* G_L *with* $L = L_S(G_L)$.

The TAG shown in Fig. 2.5 generates a string language that is not context-free. Since it is easy to show that each context-free language can be generated by a TAG without constraints (i.e. with $C_\gamma(u) = A$ and $O_\gamma(u) = 0$ for all $\gamma \in I \cup A$ and all $u \in \mathcal{U}_\gamma$), even the languages generated by TAGs without constraints are a proper superset of the context-free languages. Examples for TAGs where adjunction constraints are of consequence, are those for the copy language $\{ww \mid w \in \{a, b\}^*\}$ (see Fig. 2.6) and for $L_4 = \{a^n b^n c^n d^n \mid 0 \leq n\}$ (see Fig. 2.7).

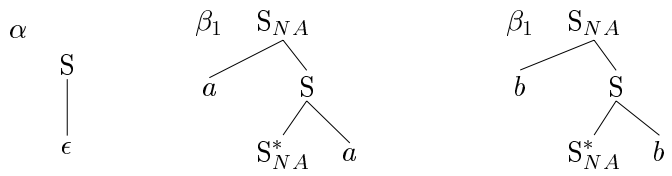
Henceforward, when talking about TAGs, these are always TAGs with adjunction constraints, i.e. TAGs as introduced in Def. 2.14.

2.2.3 Some formal properties of TAGs

In this subsection I restrict myself to those formal properties of TAGs that are important for this dissertation. Further discussions of formal properties of TAGs can be found in Vijay-Shanker and Joshi 1985 and Vijay-Shanker 1987.

Vijay-Shanker (1987) has proven the following pumping lemma for Tree Adjoining Languages:

Initial and auxiliary trees:



Adjunction constraints:
For all $\gamma \in \{\alpha, \beta_1, \beta_2\}$ and all $u \in \mathcal{U}_\gamma$:
1. $O_\gamma(u) = 0$, and
2. if u has a subscript NA, then $C_\gamma(u) = \emptyset$, else $C_\gamma(u) = \{\beta_1, \beta_2\}$.

Derivation for *abbabb*:

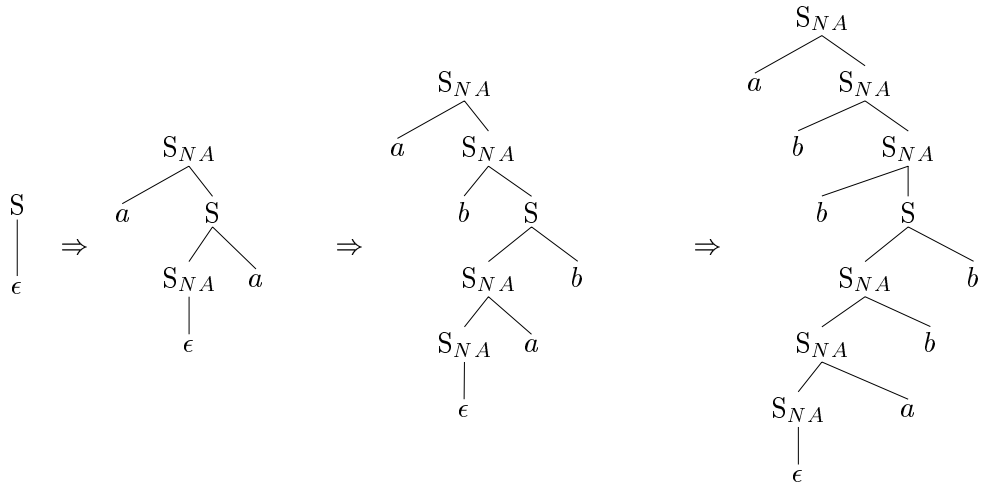


FIGURE 2.6: TAG FOR $\{ww \mid w \in \{a, b\}^*\}$

Proposition 2.1 (Pumping lemma for TALs) *If L is a TAL, then there is a constant n such that for all $z \in L$ with $|z| \geq n$, the following holds: z may be written as $z = u_1v_1w_1v_2u_2v_3w_2v_4u_3$ with $|v_1w_1v_2v_3w_2v_4| \leq n$, $|v_1v_2v_3v_4| \geq 1$ such that for all $i \geq 0$, $u_1v_1^iw_1v_2^iu_2v_3^iw_2v_4^iu_3 \in L$.*

This pumping lemma is important for the work presented in the next chapters because I will use the proof idea of this lemma to show a pumping lemma for the languages generated by the local Tree Description Grammars defined in Chapter 4. The pumping lemma for TALs is proven in the following way: Vijay-Shanker first defines a derivation grammar describing the derivation process of a TAG. This grammar is context-free, and therefore the CFG pumping lemma holds for the derivation grammar of a TAG. For a derivation in a TAG, a pumping in the corresponding local derivation tree implies a more complex pumping of the TAG tree. By looking at this more closely, Vijay-Shanker has shown the TAG pumping lemma.

Such a pumping lemma is very useful because it gives us a way of showing for certain

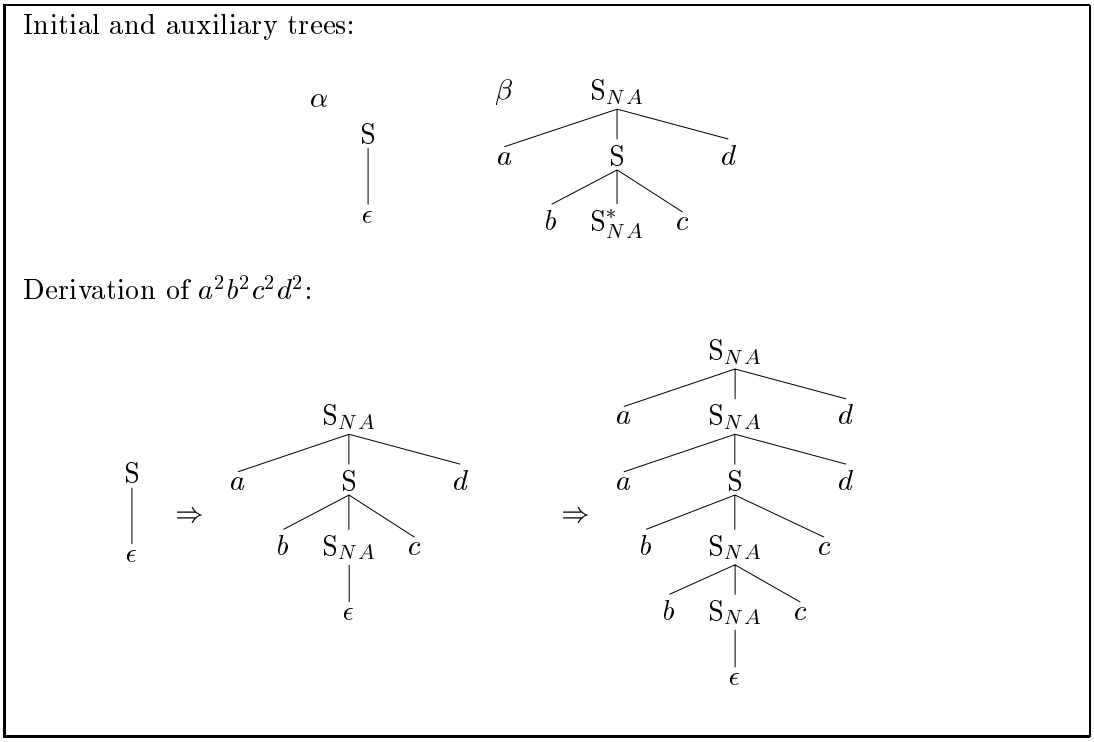


FIGURE 2.7: TAG FOR $\{a^n b^n c^n d^n \mid 0 \leq n\}$

languages that they cannot be generated by a grammar of the formalism in question. This can be done by showing that they do not satisfy the pumping lemma. In this respect, pumping lemmas are helpful in order to show descriptive complexity results, i.e. in order to classify a class of language with respect to other classes of languages.

It is easy to see that context-free languages are a proper subset of Tree Adjoining Languages. The construction of a TAG for a given context-free grammar is quite straightforward, and for example $\{a^n b^n c^n d^n \mid 0 \leq n\}$ is a TAL (see Fig. 2.7) but no context-free language. With the pumping lemma for TALs it can be shown that for $n > 4$ the languages $\{a_1^k \ldots a_n^k \mid 0 \leq k\}$ are no TALs. However these languages are indexed languages. As Vijay-Shanker (1987) has proven, TALs are a subset of indexed languages. Altogether the following proposition for TALs holds:

Proposition 2.2 ($CFL \subset TAL \subset IND$) *The class of Tree Adjoining Languages is a proper superset of the set of context-free languages and a proper subset of the set of indexed languages.*

Next the notion of projection will be introduced. Roughly, two trees are projections of each other if they have the same structures and if they yield the same strings (they only differ with respect to their nonterminal labels).

Definition 2.18 (Projection)

1. *Let γ_1 and γ_2 be finite labelled trees. γ_1 is a projection of γ_2 (and vice versa) iff there is a bijection $f : \mathcal{U}_{\gamma_1} \to \mathcal{U}_{\gamma_2}$ such that*
   - *for all $u_1, u_2 \in \mathcal{U}_{\gamma_1}$ and all $\mathcal{R} \in \{\mathcal{P}, \mathcal{D}, \mathcal{L}\}$: $\langle u_1, u_2 \rangle \in \mathcal{R}_{\gamma_1}$ iff $\langle f(u_1), f(u_2) \rangle \in \mathcal{R}_{\gamma_2}$,*
   - *and for all $u_1 \in \mathcal{U}_{\gamma_1}$ and $t \in T \cup \{\epsilon\}$: $\mu_{\gamma_1}(u_1) = t$ iff $\mu_{\gamma_2}(f(u_1)) = t$.*
2. *Let T_1 and T_2 be two sets of finite labelled trees. T_1 is a projection of T_2 iff for all $\gamma_2 \in T_2$ there is a projection γ_1 of γ_2 in T_1 and vice versa.*

Note that for two sets of trees, T_1, T_2, if T_1 is a projection of T_2, then of course T_2 is also a projection of T_1. But this does not necessarily mean that $|T_1| = |T_2|$.

Definition 2.19 (Weakly, projection and strongly equivalent) *Let G_1 and G_2 be two TAGs.*

1. *G_1 and G_2 are weakly equivalent, iff $L_S(G_1) = L_S(G_2)$.*
2. *G_1 and G_2 are projection equivalent, iff $L_T(G_1)$ is a projection of $L_T(G_2)$.*
3. *G_1 and G_2 are strongly equivalent, iff $L_T(G_1) = L_T(G_2)$.*

The following lemmata are useful because they facilitate proofs concerned with TAGs. They will be used in the course of the following chapters.

The first lemma says that the substitution operation does not increase the generative capacity of TAGs. Originally TAGs were introduced without the substitution operation. Substitution is desirable because of linguistic motivation but from a purely formal point of view it is not necessary. In the TAG literature, this fact is taken to be well-known, but I did not see any proof of it yet. The proof given below is my own.

Lemma 2.1 *For each TAG G there is a strongly equivalent TAG G' without substitution nodes.*

This lemma is proven by constructing the TAG G' for a given TAG G. In principle all substitution nodes must be replaced by appropriate initial trees. But, besides trees without substitution nodes, there might be cycles, i.e. trees with a substitution node labelled by the same label as the root of the initial tree they are derived from. These trees will be auxiliary trees in G'. Accordingly, the construction is as follows: first, starting from the elementary trees of G, a tree set T_G is obtained by performing any number of substitutions as long as there are no cycles, i.e. as long as there is no α substituted for a node in a subtree derived from some other initial α' with the same root symbol as α. Then all terminal trees in this set T_G are initial trees in G', all trees without substitution nodes but with a foot node are auxiliary trees, and, furthermore, all trees derived from an initial tree that have exactly one substitution node labelled by the same symbol as the root are also auxiliary trees in G'.

Proof Let $G = \langle N, T, I, A \rangle$ be a TAG.
First, a set T_G is constructed:

(a) If $\gamma \in I \cup A$, then $\gamma \in T_G$.

(b) If there are $\gamma_0 \in I \cup A$ and $\alpha_1, \ldots, \alpha_n \in I$ and $\alpha'_1, \ldots, \alpha'_n$ isomorphic to $\alpha_1, \ldots, \alpha_n$ respectively, such that

- $\gamma_0 \stackrel{\alpha_1}{\Rightarrow} \gamma_1 \ldots \gamma_{n-1} \stackrel{\alpha_n}{\Rightarrow} \gamma_n = \gamma$ with $\gamma_i = \gamma_{i-1}[u_i, \alpha_i']$ for $1 \leq i \leq n$, and
- for all $\gamma_i, 1 \leq i \leq n$: if there is a $j < i$ such that $\mu(u_0^{\alpha_j'}) = \mu(u_0^{\alpha_i'})$ then $\langle u_0^{\alpha_j'}, u_0^{\alpha_i'} \rangle \notin \mathcal{D}_{\gamma_i}$.

then $\gamma \in T_G$.

In γ, for all i $(1 \leq i \leq n)$, $u_0^{\alpha_i'}$ is marked as root of the initial α_i.

(c) These are all trees in T_G.

Construction of a TAG $G' = \langle N, T, I', A' \rangle$:

1. Initial trees: $\gamma \in I'$ iff $\gamma \in T_G$ and γ is a terminal tree.
2. Auxiliary trees: $\gamma \in A'$ iff
   (i) either $\gamma \in T_G$, γ has no substitution nodes and γ has a foot node,
   (ii) or there is a $\gamma' \in T_G$ without foot node that has exactly one substitution node u_s with $\mu_{\gamma'}(u_s) = \mu_{\gamma'}(u_0^{\gamma'})$, and there is a $\alpha \in I$ with $\mu_\alpha(u_0^\alpha) = \mu_{\gamma'}(u_s)$, such that γ is (isomorphic to) the following tree γ'':
   - $\mathcal{U}_{\gamma''} = \mathcal{U}_{\gamma'}, \mathcal{P}_{\gamma''} = \mathcal{P}_{\gamma'}, \mathcal{L}_{\gamma''} = \mathcal{L}_{\gamma'}, u_0^{\gamma''} = u_0^{\gamma'}$, and $\mu_{\gamma''} = \mu_{\gamma'}$.
   - $C_{\gamma''}(u) = C_{\gamma'}(u)$ and $O_{\gamma''}(u) = O_{\gamma'}(u)$ for all $u \in \mathcal{U}_{\gamma''} \setminus \{u_s\}$.
   - u_s is foot node of γ'' with $C_{\gamma''}(u_s) := C_\alpha(u_0^\alpha)$ and $O_{\gamma''}(u_s) := O_\alpha(u_0^\alpha)$.

   γ is then marked as α-*auxiliary* tree.
3. Adjunction constraints:
   First, for all $\beta' \in A' \cap T_G$ that were obtained (when constructing T_G) from some $\beta \in A$: replace β in all sets $C_\gamma(u)$ for any $\gamma \in I' \cup A'$ and $u \in \mathcal{U}_\gamma$ by all these β'.
   Then, for all $\gamma \in I' \cup A'$ and $u \in \mathcal{U}_\gamma$: if u is marked as root of the initial tree $\alpha \in I$, then add all $\gamma_\alpha \in A'$ to $C_\gamma(u)$ that are marked as α-auxiliary.

Clearly, (if there are no isomorphic trees γ, γ' that are both in $I' \cup A'$) the construction terminates after finitely many steps, and the result G' is a TAG.

To show: $L_T(G) = L_T(G')$.

First, the notion of s-nodes of a tree $\gamma \in L_T(G)$ wrt its derivation is defined:

Let $\alpha \in I$, $\alpha \stackrel{*}{\Rightarrow} \gamma$ in G. Let this derivation be ordered in the following way (since the derivation process is local, this is possible without loss of generality):

$$\alpha = \gamma_0' \stackrel{*}{\Rightarrow} \gamma_1 \stackrel{\beta_1}{\Rightarrow} \gamma_1' \stackrel{*}{\Rightarrow} \ldots \gamma_{k-1} \stackrel{\beta_{k-1}}{\Rightarrow} \gamma_{k-1}' \stackrel{*}{\Rightarrow} \gamma_k = \gamma \text{ where}$$

(a) $\beta_i \in A$ $(1 \leq i \leq k-1)$,
(b) and the derivation steps $\gamma_i' \stackrel{*}{\Rightarrow} \gamma_{i+1}$ $(0 \leq i < k)$ only consist of substitutions where for $i > 0$ all substitution nodes replaced in these steps do not occur in γ_i.

Then for all i $(1 \leq i < k)$, and $u_1, u_2 \in \mathcal{U}_{\gamma_{i+1}} \setminus \mathcal{U}_{\gamma_i'}$: $\langle u_1, u_2 \rangle$ is called an *s-pair* in γ_{i+1} iff

- $\mu_{\gamma_{i+1}}(u_1) = \mu_{\gamma_{i+1}}(u_2)$,
- $u_1 \neq u_2$,
- $\langle u_1, u_2 \rangle \in \mathcal{D}_{\gamma_{i+1}}$, and
- both, u_1 and u_2 are roots of some initial tree that was added in the course of the derivation $\gamma_i' \stackrel{*}{\Rightarrow} \gamma_{i+1}$.

Furthermore, for an s-pair $\langle u_1, u_2 \rangle$ in γ_{i+1}, u_1 is called *s-node* in γ_{i+1}, iff

for all s-pairs $\langle u, u' \rangle$ in γ_{i+1}:
if $\langle u_1, u \rangle \in \mathcal{D}_{\gamma_{i+1}}$, $\langle u, u_2 \rangle \in \mathcal{D}_{\gamma_{i+1}}$ and $u \neq u_2$, then $\langle u', u_2 \rangle \in \mathcal{D}_{\gamma_{i+1}}$.

For this order of the derivation of γ, I define $n_s(\gamma) := \sum_{i=1}^{n} |\{u \mid u \text{ s-node in } \gamma_i\}|$.

Now by induction on n, the following can be shown:

$\gamma \in L_T(G)$ with a derivation $\alpha \stackrel{*}{\Rightarrow} \gamma$ for some $\alpha \in I$ such that n_1 is the number of auxiliary trees adjoined in the course of this derivation, $n_2 := n_s(\gamma)$ wrt to this derivation, and $n = n_1 + n_2$
iff
$\gamma \in L_T(G')$ with a derivation $\alpha' \stackrel{n}{\Rightarrow} \gamma$ for some $\alpha' \in I'$.

Induction on n:

1. Induction start $n = 0$: in G, $\gamma \in L_T(G)$ is derived without adjunction and it does not contain any s-node $\Leftrightarrow$ γ is one of the terminal trees in $T_G \Leftrightarrow \gamma \in I'$.
2. Induction step $n \rightarrow n + 1$:
   Suppose that there is an $\alpha \in I$ such that $\alpha \stackrel{*}{\Rightarrow} \gamma$ in G with n_1 adjunctions, $n_2 = n_s(\gamma)$ wrt this derivation, and $n + 1 = n_1 + n_2$.
   There are two cases that must be distinguished:
   (a) eiter there is a β' derived by substitutions from a tree $\beta \in A$ that has neither substitution nodes nor s-nodes such that β' is a subtree of γ, and there is an order $\alpha \stackrel{*}{\Rightarrow} \gamma' \stackrel{\beta}{\Rightarrow} \gamma'' \stackrel{*}{\Rightarrow} \gamma$ of the derivation with $\gamma'' = \gamma'[u, \beta]$. There are no adjunctions in $\gamma'' \stackrel{*}{\Rightarrow} \gamma$, and all substitutions in this part are done with substitution nodes that are not in $\mathcal{U}_{\gamma'}$.
   In this case γ' also has n_2 s-nodes, and it can be derived by performing only $n_1 - 1$ adjunctions.
   iff
   (induction claim) there is an $\alpha' \in I'$ with $\alpha' \stackrel{n}{\Rightarrow} \gamma'$, and (because of the construction of T_G), $\beta' \in A'$, and $\gamma' \stackrel{\beta'}{\Rightarrow} \gamma$ with $\gamma = \gamma'[u, \beta']$ ($\beta' \in C_{\gamma'}(u)$ because of the construction of G').
   (b) or there is an s-node u_1 in γ, and u_2 is the node such that $\langle u_1, u_2 \rangle$ form an s-pair, and furthermore, there is no s-node u_s in γ with $u_s \neq u_1$, $\langle u_1, u_s \rangle \in \mathcal{D}_\gamma$ and $\langle u_2, u_s \rangle \notin \mathcal{D}_\gamma$.
   iff
   There are $\alpha_1, \alpha_2 \in I$ such that the derivaition of γ in G can be ordered as follows: $\alpha \stackrel{*}{\Rightarrow} \gamma' \stackrel{\alpha_1}{\Rightarrow} \gamma_1 \stackrel{*}{\Rightarrow} \gamma_1' \stackrel{\alpha_2}{\Rightarrow} \gamma_2 \stackrel{*}{\Rightarrow} \gamma$ with $\gamma_1 = \gamma'[u_1', \alpha_1']$ and $\gamma_2 = \gamma'[u_2', \alpha_2']$such that there is no adjunction in $\gamma' \stackrel{*}{\Rightarrow} \gamma$, and all substitution nodes in γ_2 are in $\mathcal{U}_{\alpha_2'}$.
   Because of $\mu_\gamma(u_0^{\alpha_1'}) = \mu_\gamma(u_0^{\alpha_2'})$, there is another derivation possible in G: $\alpha \stackrel{*}{\Rightarrow} \gamma' \stackrel{\alpha_2}{\Rightarrow} \gamma'' \stackrel{*}{\Rightarrow} \gamma_n$ with $\gamma'' = \gamma'[u_1', \alpha_2']$. and the derivation steps in $\gamma'' \stackrel{*}{\Rightarrow} \gamma_n$ are the same substitutions (i.e. with the same initial trees and the same substitution nodes) as in $\gamma' \stackrel{*}{\Rightarrow} \gamma$. γ_n has $n_2 - 1$ s-nodes.
   iff
   There is an $\alpha' \in I'$ with $\alpha' \stackrel{n}{\Rightarrow} \gamma_n$ in G' (induction claim), and there is a $\beta' \in A'$ that was obtained from α_1 in T_G such that $\gamma_n \stackrel{\beta'}{\Rightarrow} \gamma$ with $\gamma = \gamma_n[u_0^{\alpha_2'}, \beta'']$ (because of adjunction constraints in G', this adjunction is allowed, and the constraints for the

foot of β'' are those that hold for the root of α_2 in G).

□

The next lemma is easier to show. It says that for each TAG, another TAG generating the same string language can be found with NA constraints at all foot nodes of auxiliary trees.

Lemma 2.2 *For each TAG G, there is a weakly equivalent TAG G' such that for all auxiliary trees β in G': if u_f^β is the foot node of β, then $C_\beta(u_f^\beta) = \emptyset$.*

Proof Let $G = \langle N, T, I, A\rangle$ be a TAG. Construct an equivalent TAG $G'' = \langle N, T, I, A'\rangle$ as follows:
for all $\beta \in A$ with $C_\beta(u_f^\beta) \neq \emptyset$: replace β in A and in all adjunction constraints by the following tree β':

1. $\mathcal{U}_{\beta'} = \mathcal{U}_\beta \cup \{u_{new}\}$ where $u_{new} \notin \mathcal{U}_\beta$.
2. $\mathcal{L}_\beta \subseteq \mathcal{L}_{\beta'}$, and $\mathcal{P}_{\beta'} = \mathcal{P}_\beta \cup \{\langle u_f^\beta, u_{new}\rangle\}$.
3. $u_f^{\beta'} = u_{new}$ is the foot node of β'.
4. $\mu_{\beta'}(u) = \mu_\beta(u)$ for all $u \in \mathcal{U}_\beta$, and $\mu_{\beta'}(u_{new}) = \mu_\beta(u_f^\beta)$.
5. $C_{\beta'}(u) = C_\beta(u)$ and $O_{\beta'}(u) = O_\beta(u)$ for all $u \in \mathcal{U}_\beta$, and $O_{\beta'}(u_{new}) = 0$ and $C_{\beta'}(u_{new}) = \emptyset$.

The result is a TAG G'.

The following is easy to see for all $n \in \mathbb{N}$ and for each $\alpha \in I$:

there is a derivation $\alpha \stackrel{n}{\Rightarrow} \gamma$ in G
iff
there is a derivation $\alpha \stackrel{n}{\Rightarrow} \gamma'$ in G' such that there is a partial function $f : \mathcal{U}_{\gamma'} \to \mathcal{U}_\gamma$ that is defined for all $u \in \mathcal{U}_{\gamma'}$ that are not one of the newly introduced foot nodes of an auxiliary tree that was added in the course of the derivation of γ'. For this partial function and for all $u, u' \in \mathcal{U}_{\gamma'}$, the following holds:

- if $f(u)$ is defined, then $O_{\gamma'}(u) = O_\gamma(f(u))$ and $C_{\gamma'}(u) = C_\gamma(f(u))$, and
- if $u' \neq u$ and both $f(u)$ and $f(u')$ are defined, then
  if $\langle u', u\rangle \in \mathcal{P}_{\gamma'}$ then $\langle f(u'), f(u)\rangle \in \mathcal{P}_\gamma$,
  $\langle u', u\rangle \in \mathcal{D}_{\gamma'}$ iff $\langle f(u'), f(u)\rangle \in \mathcal{D}_\gamma$, and
  $\langle u', u\rangle \in \mathcal{L}_{\gamma'}$ iff $\langle f(u'), f(u)\rangle \in \mathcal{L}_\gamma$.

By induction on the length n of the derivation, this can be easily shown.

Since none of the leaves in γ' was added in the course of the construction of G', $L_S(G) = L_S(G')$ holds.

□

In an analogous way, the following lemma can be shown. This lemma signifies that for each TAG, another TAG generating the same string language exists with NA constraints at the roots of auxiliary trees.

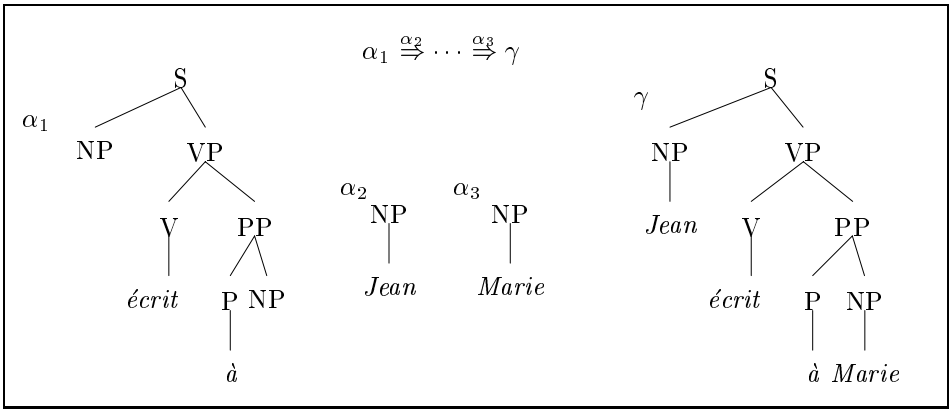


FIGURE 2.8: ELEMENTARY TREES AND DERIVATION FOR *Jean écrit à Marie ('Jean writes to Mary')*

Lemma 2.3 *For each TAG G, there is a weakly equivalent TAG G' such that for all auxiliary trees β in G': if u_0^β is the root of β, then $C_\beta(u_0^\beta) = \emptyset$.*

These are all formal properties of TAGs that are referred to in the course of the dissertation.

2.3 TAGs for natural languages

For purposes of natural language processing, lexicalized TAGs (LTAG) are used. These are TAGs where each elementary tree contains at least one leaf labelled by a terminal symbol. The underlying idea is that each elementary tree has a unique anchor. This is a leaf with a terminal label. Besides this anchor, other leaves may also be labelled by terminals.

Definition 2.20 (Lexicalized TAG) *A TAG $G = \langle N, T, I, A \rangle$ is lexicalized iff for all $\gamma \in I \cup A$ there is a leaf $u \in \mathcal{U}_\gamma$ with $\mu_\gamma(u) \in T$.*

Using LTAGs for natural languages, several linguistic constraints are imposed on elementary structures (see Abeillé 1993, 1994). These constraints on elementary trees are the following:

- *Lexicalization*: The TAG must be lexicalized, and for each elementary tree there must be exactly one leaf that is the unique anchor (or head) of the tree. This leaf has a terminal label.
- *Predicate-argument cooccurrence*: For each predicate and each elementary tree with that predicate as label of the anchor, there is a subcategorization frame for this predicate, such that there is a node for each of the argument slots.
- *Semantic consistency*: Each elementary tree has a meaning, i.e. a semantic interpretation.

- *Semantic minimality*: An elementary tree corresponds to a single semantic unit.

The definition of a lexicalized TAG given above captures only one aspect of the principle of lexicalization, since it does not say anything about the uniqueness of the anchor. The other principles are not formalized and partly they are quite vague. They are not part of the grammar, in contrast to principles in HPSG for example, but they are taken into account when constructing a TAG for natural language analysis.

The last two principles in particular are quite unclear without knowing what is meant by the semantic interpretation of a syntactic structure. However, the idea of these two principles is clear: corresponding to the TAG describing the syntax of a language there is a compositional semantic theory. The elementary trees should be such that for each elementary tree there is a corresponding semantics, which must be minimal in the sense that it cannot be a composition of other semantic expressions. This means that for each lexical item that is not semantically void, there must be an elementary tree in the grammar with this lexical item as unique anchor. Besides this anchor, the elementary tree must not contain any other lexical item with a semantic interpretation. The initial tree α_1 in Fig. 2.8 contains more than one lexical item, *écrit* and *à*. In this case, the leaf with label *écrit* is the anchor of the elementary tree. *écrit* selects of a PP with preposition *à*, but this preposition itself is semantically void in this context.

Originally adjunction was the only operation in TAGs. As we have seen, from a technical point of view, substitution is not necessary. According to Lemma 2.1, for each TAG a strongly equivalent TAG without substitution nodes can be found. But substitution is needed for natural languages because of the uniqueness of the anchor. With adjunction as the only derivation operation, the semantic principles underlying TAGs and the requirement of the uniqueness of the anchor for natural languages cannot be satisfied.

As already mentioned, modifiers are usually added by adjunction, i.e. they do not fill an open slot, which reflects the optional character of modifiers. Examples are adjectives modifying nouns as in Fig. 2.2. In contrast to this, according to the predicate-argument cooccurrence principle, in each elementary tree of a predicate there must be substitution or foot nodes for each of the arguments of the predicate. For most complements, there are substitution nodes, e.g. for NP and PP complements (see Fig. 2.8). This corresponds to the obligatory character of the complements as substitution nodes always must be replaced by some initial tree to derive a tree belonging to the tree language.

The only exception are complement clauses. The slot for a complement clause does not consist of a substitution node but of a foot node in an auxiliary tree describing the whole argument structure. This is one of the problems with TAGs. One would like to have a uniform distinction between complements and modifiers by analyzing complements by substitution and modifiers by adjunction. But for the following reasons this is not possible (see Kroch 1987 and Abeillé 1988a,b for the analysis of long-distance dependencies and complement clauses in TAGs).

With complement clauses the problem is that there might be a long-distance dependency, e.g. a wh-movement out of a complement clause as in (3).

(3) qui penses-tu que Marie aime? ('whom do you think Mary loves?')

Even in this case the predicate-argument cooccurrence principle should be respected. Therefore the substitution or foot node for the moved element and the trace must be in

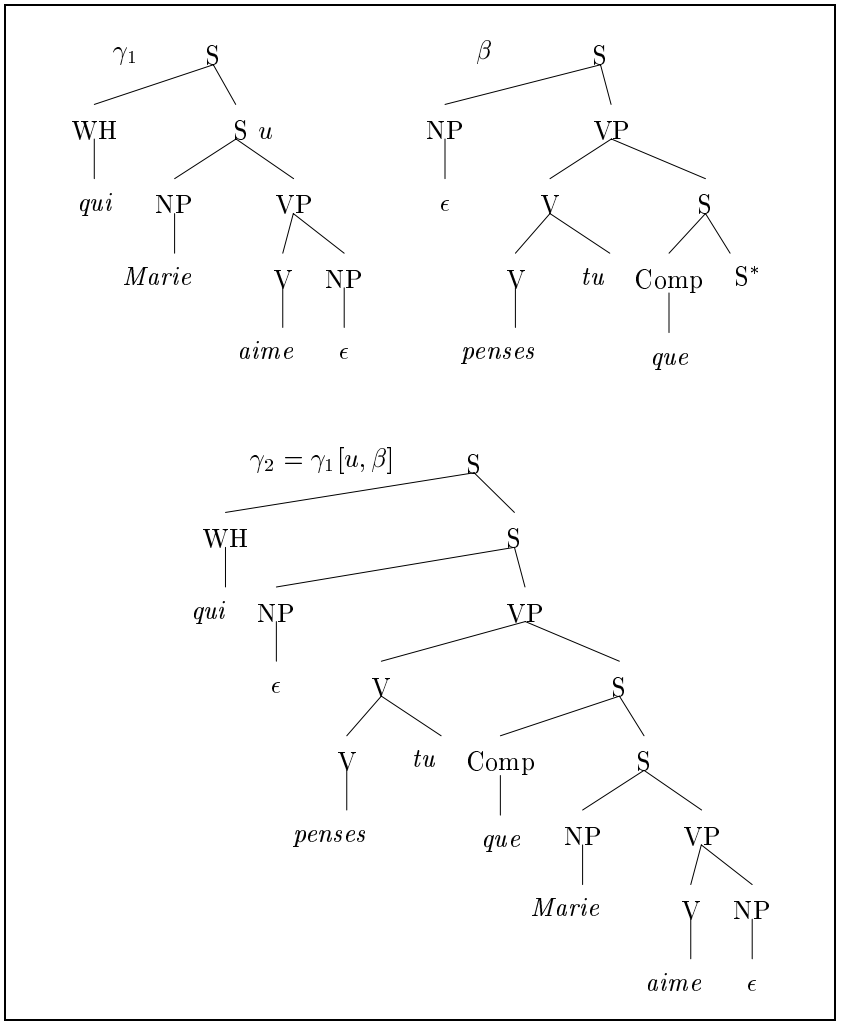


FIGURE 2.9: DERIVATION STEP FOR (3) *qui penses-tu que Marie aime?*

the same elementary tree. But this is only possible if the predicate-argument structure (the elementary tree of the predicate) is adjoined to the complement clause, as is shown in Fig. 2.9.

This problem that complement clauses and other arguments cannot be handled in a uniform way, is one of the motivations for replacing the substitution operation by the *subsertion* operation defined in D-Tree Grammars (see Rambow et al. 1995). I will present this formalism in 2.5.2. When defining Tree Description Grammars in Chapter 3 we will see that these grammars also allow derivation steps similar to subsertion and therefore allow a uniform treatment of arguments.

2.4 TAG variants

Several variants of standard TAGs have been proposed, e.g. feature-structure-based TAGs (FTAG, see Vijay-Shanker and Joshi 1988), probabilistic TAGs (see Resnik 1992), synchronous TAGs (see Shieber and Schabes 1990) and multicomponent TAGs (MC-TAG, see Weir 1988).

In this section, I will present two of these variants, namely *Multi-Component TAGs* and *synchronous TAGs*. Both have influenced the work presented in this dissertation.

The idea of Multi-Component TAGs, first mentioned in Joshi et al. 1975, is to add not only one elementary tree but a set of elementary trees in one derivation step. This extension of TAG is necessary to handle non-local dependencies such as extraposition.

Multi-Component TAGs are introduced in this chapter because they constitute a first step towards the idea of adding more than one subtree in one dervivation step. Later we will see that this idea also can be found with Tree Description Grammars, the grammars introduced in Chapter 3. In Chapter 4, the local variant of Tree Description Grammars will be compared to set-local Multi-Component TAGs.

Synchronous TAGs were proposed by Shieber and Schabes (1990) to describe a relation between two grammars, in particular to describe the relation between syntax and semantics for natural languages. Following this idea, I will define synchronous local TDGs in Chapter 4, and in Chapter 5 a syntax-semantics interface for a fragment of French is developed using this formalism.

2.4.1 Multi-Component TAGs

Multi-Component TAGs were first defined in Joshi et al. 1975 where they are not yet called Multi-Component TAGs but simultaneous TAGs. The idea of Multi-Component TAGs is to adjoin several auxiliary trees that form a so-called auxiliary set, at the same time. In Joshi 1987 this kind of derivation is first called multicomponent adjoining, and Weir (1988) gives a definition of this TAG extension calling it Multi-Component TAG (MC-TAGs). Depending on the way the derivation is restricted, *tree-local*, *set-local* and *non-local* MC-TAGs are distinguished.

In this subsection, I will first present the three versions of MC-TAGs and then I will mention some observations made in the literature concerning the relevance of MC-TAGs for natural languages.

MC-TAG formalism

A Multi-Component TAG is a simple TAG with an additional set of sets of auxiliary trees, the so-called *auxiliary sets*.

Definition 2.21 (Multi-Component TAG) *A Multi-Component TAG (MC-TAG) is a tuple* $G = \langle N, T, I, A, \mathcal{A} \rangle$ *such that*

1. $\langle N, T, I, A \rangle$ *is a TAG.*

2. $\mathcal{A} \subseteq Pow(A) \setminus \emptyset$.

Each set in $\mathcal{A}$ *is called an auxiliary set.*

The set $\mathcal{A}$ contains sets of auxiliary trees. The derivation process is such that in each derivation step, one of these auxiliary sets is used. All auxiliary trees in this set must be adjoined simultaneously, i.e. in this single derivation step. Three different types of MC-TAGs are distinguished, depending on different restrictions with respect to the nodes where simultaneous adjunction is possible. An MC-TAG is called tree-local if all nodes where trees from one auxiliary set are adjoined, are in one single elementary tree. It is called set-local, if these nodes belong either all to one initial tree or all to elementary trees from one single auxiliary set. If there is no restriction on the choice of the nodes for the simultaneous adjunctions in one derivation step, then the MC-TAG is called non-local. This distinction was first made in Joshi 1987, and the names "tree-local" and "set-local" stem from Rambow (1994a).

Definition 2.22 (Multicomponent derivation) *Let $G = \langle N, T, I, A, \mathcal{A} \rangle$ be an MC-TAG. Derivation in G is inductively defined:*

1. *Each $\alpha \in I$ can be derived in 0 derivation steps from α.*
2. *Let γ be a tree (with C_γ and O_γ) derived in n steps from an initial $\alpha \in I$.*
   γ' can be derived in 1 step from γ (i.e. in $n+1$ steps from α) iff there is a $\{\beta_1, \ldots, \beta_k\} \in \mathcal{A}$ and a set $\{\beta'_1, \ldots, \beta'_k\}$ such that β_i is isomorphic to β'_i for $1 \leq i \leq k$, and if there are nodes $u_1, \ldots, u_k \in \mathcal{U}_\gamma$ such that:
   (a) *$\gamma' = \gamma[u_1, \beta'_1] \cdots [u_k, \beta'_k]$,*
   (b) *$C_{\gamma'}(u') = C_\gamma(u')$ and $O_{\gamma'}(u') = O_\gamma(u')$ for all $u' \in \mathcal{U}_\gamma$, and*
   for all i, $1 \leq i \leq k$: $C_{\gamma'}(u') = C_{\beta'_i}(u')$ and $O_{\gamma'}(u') = O_{\beta'_i}(u')$ for all $u' \in \mathcal{U}_{\beta'_i}$,
   (c) *for all i, $1 \leq i \leq k$: $\beta_i \in C_\gamma(u_i)$,*
   (d) *and a locality condition (L) specified below holds.*

   Notation: $\gamma \overset{\{\beta_1,\ldots,\beta_k\}}{\Rightarrow} \gamma'$ or $\gamma \Rightarrow \gamma'$.

The locality condition (L) is

- *either the following: there is an elementary tree γ_e added in the course of the derivation of γ such that $\{u_1, \ldots, u_k\} \subseteq \mathcal{U}_{\gamma_e}$. Then the MC-TAG is called tree-local.*
- *or the following: either $u_1, \ldots, u_k \in \mathcal{U}_\alpha$, or there is an auxiliary set S_a isomorphic to some element in $\mathcal{A}$ that has been added in one of the previous derivation steps such that $\{u_1, \ldots, u_k\} \subseteq \bigcup_{\beta \in S_a} \mathcal{U}_\beta$. Then the MC-TAG is called set-local.*
- *or no restriction at all, i.e. $\{u_1, \ldots, u_k\} \subseteq \mathcal{U}_\gamma$. Then the MC-TAG is called non-local.*

Roughly said, the notion "local" signifies that the modifications of the derived structure that are caused by one derivation step are local in the following sense: these modifications do not concern arbitrary nodes in the derived tree. Instead they concern only a part of the tree, and this part is an elementary tree (in the case of tree-local MC-TAGs) or an auxiliary set (in the case of set-local MC-TAGs), i.e. this part is an element of the grammar and it was added in one previous derivation step.

Tree-local MC-TAGs are even strongly equivalent to TAGs. For set-local MC-TAGs, Weir (1988) has shown that they are still in the class of mildly context-sensitive grammar formalisms, i.e. that the derivation process is context-free and therefore the languages generated by these grammars are semilinear, the grammars are polynomially parsable, and they

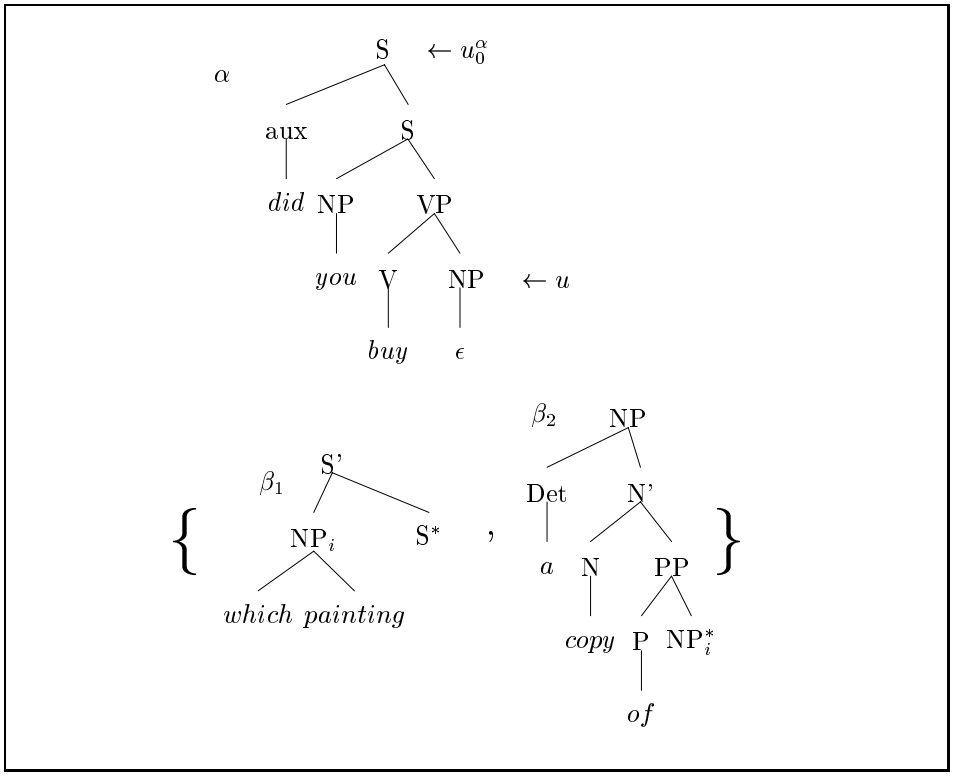


FIGURE 2.10: A SAMPLE MC-TAG FOR EXTRACTION FROM NP

allow only limited cross-serial dependencies. Non-local MC-TAGs are more powerful, since they generate non-semilinear and NP-complete languages (see Rambow 1994a).

MC-TAGs and natural languages

As we have already seen, long distance dependencies such as wh-movement are handled in TAGs without any movement or transformation mechanism because the intermediate material is added by adjunction. However, there are other cases of nonlocal dependencies, e.g. extraction out of noun phrases that cannot be handled in standard TAG while preserving the predicate-argument cooccurrence restriction. Examples are (see Kroch 1987, Kroch and Joshi 1987):

(4) a. A man arrived who knew Mary.

b. John gave everyone trouble who knew Mary.

c. Which painting$_i$ did you buy a copy of ϵ_i?

The problem here is that there is an NP that cannot be part of one single elementary tree. In (4)a. this is *a man who knew Mary*, in (4)b. the NP is *everyone who knew Mary*, and in (4)c. *which painting$_i$ a copy of* ϵ_i. Kroch and Joshi (1987) propose the use of tree-local

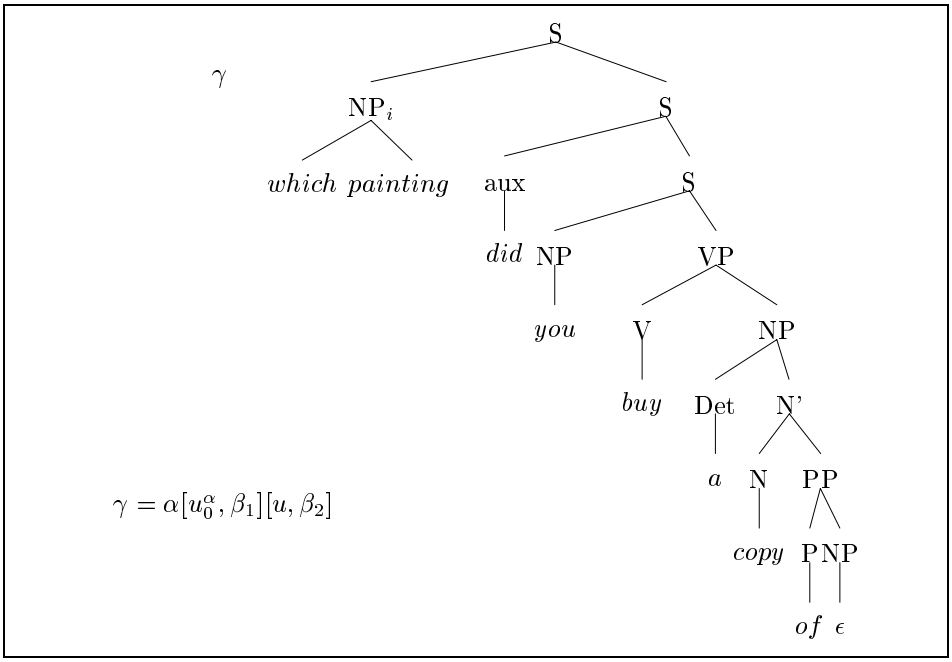


FIGURE 2.11: A DERIVATION IN AN MC-TAG

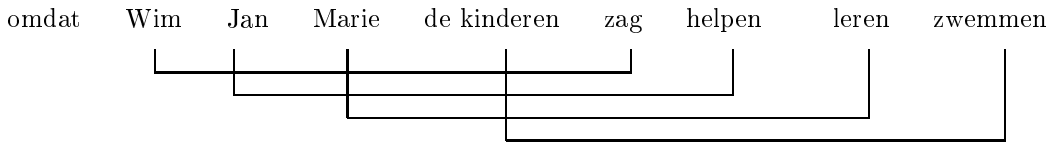
MC-TAGs to handle these examples. For (4)c., the MC-TAG shown in Fig. 2.10 generates the tree of Fig. 2.11 by simultaneous adjoining of the two trees in the auxiliary set $\{\beta_1, \beta_2\}$ (see Kroch 1987). An application of tree-local MC-TAGs to the representation of extraposed PPs in French can be found in Abeillé 1994.

However, tree-local and even set-local MC-TAGs are probably not powerful enough to represent the syntax of natural languages. In Becker et al. 1992 and Rambow 1994a, it is shown that scrambling phenomena in free word order languages such as German cannot be handled by set-local MC-TAGs.

Besides weak and strong generative capacity of a grammar, Becker et al. (1992) introduce the notion of *derivational generative capacity*. This is motivated by the following observation: while the specifics of a syntactic analysis of a particular construction may be subject to controversy, the (semantic) predicate-argument structure of sentences is fairly uncontroversal. Furthermore, what characterizes many syntactic phenomena in natural languages is the way the predicate-argument structures map onto the surface string. Therefore, in order to decide whether a grammar formalism is adequate for a natural language L, it is not only important whether L can be generated as a string language by this formalism, but (if the predicate-argument cooccurrence condition is accepted as a reasonable condition) it must be generated in such a way that each predicate is added in the same derivation step as the slots for its arguments.

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

For cross-serial dependencies in Dutch such as (5) for example, different syntactic structures have been proposed (see for example Bresnan et al. (1982)) but the dependencies between predicates and arguments are always analyzed as in



In other words, cross-serial dependencies are such that a string of nouns is followed by a string of verbs, where the first noun is the argument of the first verb, and so on. Therefore, the i-th noun and the i-th verb depend on each other and must be added in the same derivation step.

In order to formalize this, Becker et al. (1992) define the derivational generative capacity of a grammar in terms of its ability to generate sets of derivation structures. The idea is to decorate the symbols of a string derived by a grammar by indices where two symbols have the same index if they were added in the same derivation step. These strings are called indexed strings.

Definition 2.23 (Indexed string language, derivationally equivalent)

1. *A pair* $\langle w, f\rangle$ *is called an indexed string, if* $w \in T^+$ *for some alphabet* T *and* $f : \{1, \ldots, |w|\} \to \mathbb{N}$ *is a mapping.*
   Notation: If $w = a_1 \ldots a_n$ *with* $a_i \in T$ $(1 \leq i \leq n)$*, then* $\langle w, f\rangle$ *is written as* $a_1^{[f(1)]} \ldots a_n^{[f(n)]}$.
2. *Let* G *be a grammar,* $w \in L(G)$ *be a word derived in* m *steps in* G *(or yielded by an object, e.g. a tree that was derived in* m *steps in* G*) and* f *a mapping such that*
   - $w = a_1 \ldots a_n$ *with* $a_i \in T$ $(1 \leq i \leq n)$*, and*
   - a_i *was added in the* $f(i)^{th}$ *derivation step.*

   Then $a_1^{[f(1)]} \ldots a_n^{[f(n)]} \in L_{ind}(G)$.
   These are all indexed strings in $L_{ind}(G)$.
   $L_{ind}(G)$ *is called the indexed string language of* G.
3. *Two grammars* G_1, G_2 *are called derivationally equivalent, iff* $L_{ind}(G_1) = L_{ind}(G_2)$ *holds.*

For the case of cross-serial dependencies, this formalization signifies that a grammar that can handle cross-serial dependencies in an adequate way must be able to derive the indexed string language $L_{cross} := \{n^{[0]} \ldots n^{[k]} v^{[0]} \ldots v^{[k]} \mid k \geq 0\}$. Obviously, if a grammar can derive this indexed language, then it can also generate the copy language $\{ww \mid w \in \{a, b\}^*\}$. Consequently, if the copy language cannot be generated by a grammar, then this grammar is not powerful enough to handle cross-serial dependencies.

Becker et al. (1992) argue that a grammar formalism that is suitable for an analysis of scrambling in German, must be able to generate the following indexed string language:

$$SCR^{ind} := \{n^{[\sigma(1)]} \ldots n^{[\sigma(m)]} v^{[1]} \ldots v^{[m]} \mid m \geq 0 \text{ and } \sigma \in S_m\}$$

where S_m is the group of the permutations of $\{1,\ldots,m\}$, i.e. each $\sigma \in S_m$ is a bijection $\sigma : \{1,\ldots,m\} \to \{1,\ldots,m\}$.

The argument is as follows: there are matrix verbs in German that allow scrambling out of embedded infinitival clauses. This phenomenon is called *long-distance scrambling*. Long-distance scrambling is quite free. As shown in (6), there is no bound to the number of clause boundaries over which an element can scramble. Furthermore, examples such as (7) show that there is also no bound on the number of elements that can scramble.

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

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

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

Under the assumption that each verb has exactly one overt nominal argument, the language SCR^{ind} gives a formal presentation for the subset of scrambled German sentences with such verbs. Consequently, a grammar that can deal with German scrambling phenomena must be able to derive SCR^{ind}.

However, for SCR^{ind}, Becker et al. show the following result:

Proposition 2.3 *There is no set-local MC-TAG G with $SCR^{ind} = L_{ind}(G)$.*

Therefore set-local MC-TAGs are not powerful enough to handle scrambling phenomena in free word order languages. In Chapter 4, this proposition is used to show that local TDGs are more powerful than MC-TAGs.

An analysis of scrambling using non-local MC-TAGs is proposed in Becker et al. 1991.

Since non-local MC-TAGs are very powerful, it is desirable to find alternatives to MC-TAGs, i.e. formlisms that also allow a kind of multicomponent derivation and that generate only semilinear languages but that allow the derivation of the indexed string language SCR^{ind}. We will see in Chapter 4 that local TDGs satisfy these conditions.

2.4.2 Synchronous TAGs

Synchronous TAGs were originally introduced in Shieber and Schabes 1990. They are useful to describe the relation between two languages that depend on each other in a compositional

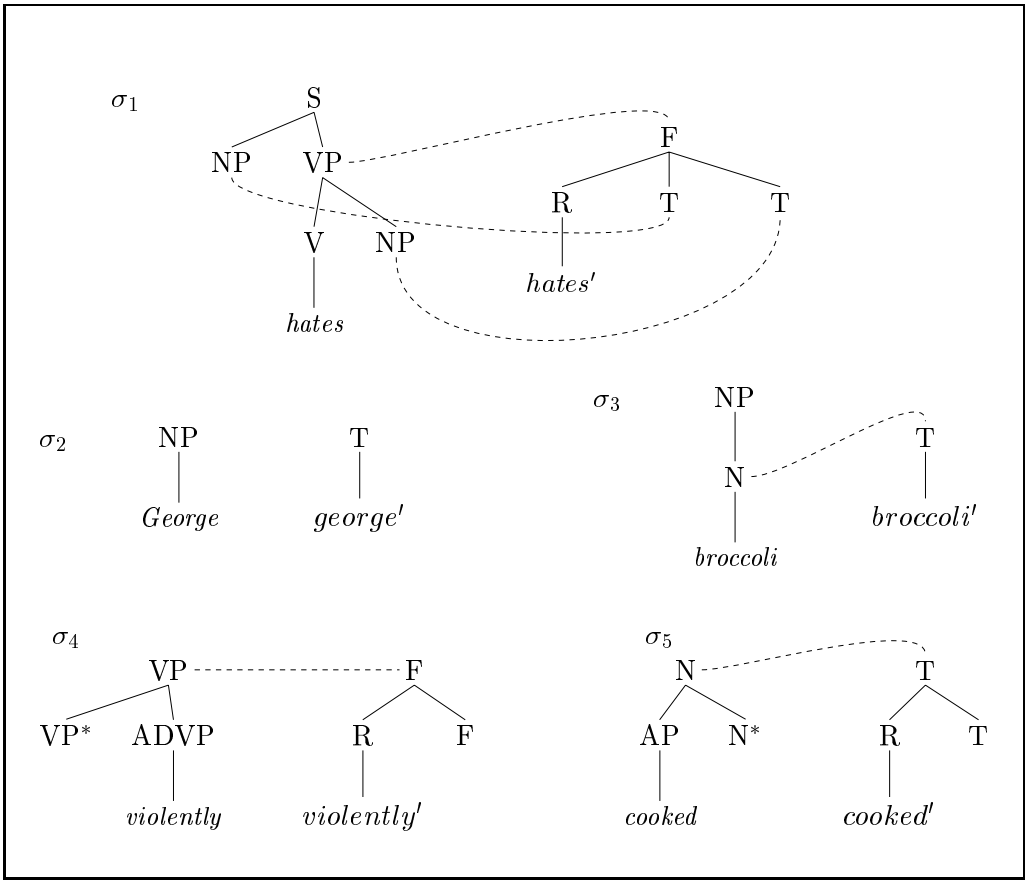


FIGURE 2.12: A SAMPLE SYNCHRONOUS TAG

way but that do not have the same syntactic structure. Synchronous TAGs are used to handle the relation between syntax and semantics as proposed in Shieber and Schabes 1990 (see also Abeillé 1993, p. 240) and for Machine Translation (see Abeillé et al. 1990, Prigent 1994).

Synchronous TAGs are important for this dissertation because in Section 4.4, the idea of synchronization underlying synchronous TAGs will be the starting point for the definition of synchronous local TDGs. Similar to Shieber and Schabes 1990, synchronous local TDGs are used to describe a syntax-semantics interface for a fragment of French in Chapter 5.

In the following I will first present in an informal way the formalism of synchronous TAGs together with some examples concerning the relation between syntax and semantics. Then I will state that, as observed in Shieber 1994, the synchronization mechanism increases the generative capacity of the grammar.

Synchronization

A synchronous TAG establishes a relation between two TALs via a relation between the elementary trees of the two corresponding grammars. Since I don't need a formal definition of synchronous TAGs in the course of this work, I will give, as Shieber and Schabes (1990)

also do, only an informal description of synchronous TAGs together with some examples.

A synchronous TAG consists of two TAGs together with a synchronization relation Σ between the two sets of elementary trees. Trees related by Σ are either both initial or both auxiliary trees. If two trees γ_1 and γ_2 are related by Σ, then there is an additional relation between the nodes $\mathcal{U}_{\gamma_1}$ of γ_1 and the nodes $\mathcal{U}_{\gamma_2}$ of γ_2. This relation, or as Shieber and Schabes (1990) call it, these *links* between nodes in γ_1 and in γ_2 specify which nodes must be treated in parallel in the course of the derivation.

The derivation is such that, starting with a pair of initial trees in Σ, in each step, the following takes place for the already derived pair $\langle \gamma_1, \gamma_2 \rangle$:

1. some node u_1 in γ_1 and some node u_2 in γ_2 is chosen such that u_1 and u_2 are linked
2. a pair of trees $\langle \gamma_1^e, \gamma_2^e \rangle$ is chosen from Σ
3. the new pair of trees then is $\langle \gamma_1[u_1, \gamma_1^e], \gamma_2[u_2, \gamma_2^e] \rangle$ if these two trees exist. The new links are the union of the links between $\mathcal{U}_{\gamma_1}$ and $\mathcal{U}_{\gamma_2}$ and the links between $\mathcal{U}_{\gamma_1^e}$ and $\mathcal{U}_{\gamma_2^e}$.

Figure 2.12 is taken from Shieber and Schabes 1990 and shows a very simple synchronous TAG. It illustrates the idea of relating English to a representation of its logical form. Links are represented by dotted lines. With this grammar, for the sentence *George hates cooked broccoli violently*, a pair of trees as shown in Fig. 2.13 can be derived.

Shieber and Schabes do not explain their links and their choice of nonterminals in the TAG for the logical form. The problem of relating syntactic and semantic representations using a synchronous pair of grammars will be investigated in Chapter 5, and the synchronous TAG in Fig. 2.12 is only an illustration of the formalism of synchronous TAGs.

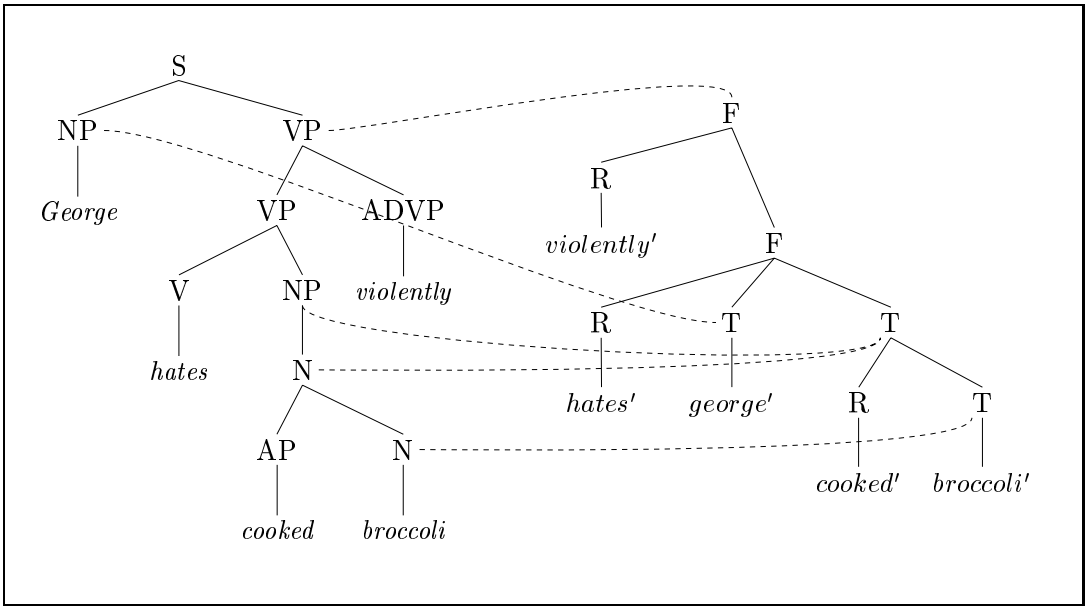


FIGURE 2.13: TREE PAIR DERIVED FOR *George hates cooked broccoli violently*

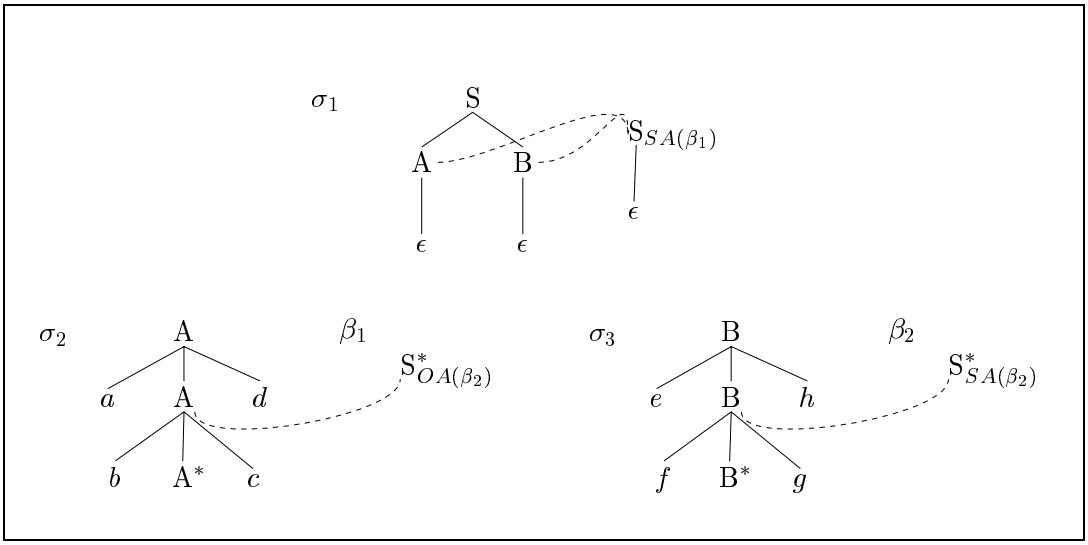


FIGURE 2.14: A SYNCHRONOUS TAG FOR L_8

Expressivity of synchronous TAGs

As Shieber (1994) points out, the definition of derivation for synchronous TAGs is problematic because the weak generative expressivity of TAGs is increased by the synchronization. In other words, the control of the derivation by the linking to a second TAG creates a more powerful formalism. An example is the language $L_8 := \{a^n b^n c^n d^n e^n f^n g^n h^n \mid n \geq 0\}$ that is no TAL. Shieber (1994) gives the synchronous TAG shown in Fig. 2.14 that generates L_8 by synchronous derivation as string language of the left TAG.

In general, it is not necessarily a problem if the expressivity of a grammar formalism is increased by adding a control mechanism using a second grammar. In the case of synchronus TAGs, however, the intention of proposing synchronous TAG was not to get a formalism with increased generative expressivity but to define a formalism that allows the description of a relation between two TALs. From this point of view, the fact that synchronous TAGs are more powerful than simple TAGs, is a problem.

2.5 Tree descriptions

In recent work on TAG-related formalisms, there have been several proposals to view TAGs as defining constraints on well-formed structures instead of generating a tree set from elementary trees. This idea stems from Vijay-Shanker (1992). He proposes to use tree descriptions, so-called *quasi-trees* to define TAGs. Other TAG variants using tree descriptions are the *Description Tree Grammars* (D-Tree Grammars) in Rambow et al. 1995, and the *Unordered Vector Grammars with Dominance Links* (UVG-DL) presented in Rambow 1994a,b.

A common property of all these formalisms is that trees are replaced by tree descriptions where the parent relation is only partially specified. In other words, a kind of underspecification is introduced by relaxing one of the binary relations holding on the nodes of a tree.

Although these approaches do not aim at the generation of underspecified representations for ambiguous expressions, such an idea suggests itself. In the following chapter we will see that suitable underspecified representations can be obtained by leaving even the dominance relation underspecified.

In this section, I will present the ideas of quasi-trees, DTGs and UVG-DLs.

2.5.1 Quasi-trees

Borrowing ideas from Marcus et al. 1983, Vijay-Shanker (1992) proposes a reinterpretation of TAGs as defining constraints on trees. His main motivation for this approach is the non-monotonicity of the TAG derivation process. In a TAG derivation, the adjoining operation on trees does not preserve the structural relationships holding in the trees that are combined. E.g., in the derivation step shown in Fig., 2.15, in α, there is only one single node between u_1 and u_3, i.e. there is a node u_2 with $\langle u_1, u_2\rangle \in \mathcal{P}_\alpha$ and $\langle u_2, u_3\rangle \in \mathcal{P}_\alpha$. In the resulting tree γ this is no longer the case because the node u_2 is no longer part of the tree, and instead there are two different nodes between u_1 and u_3.

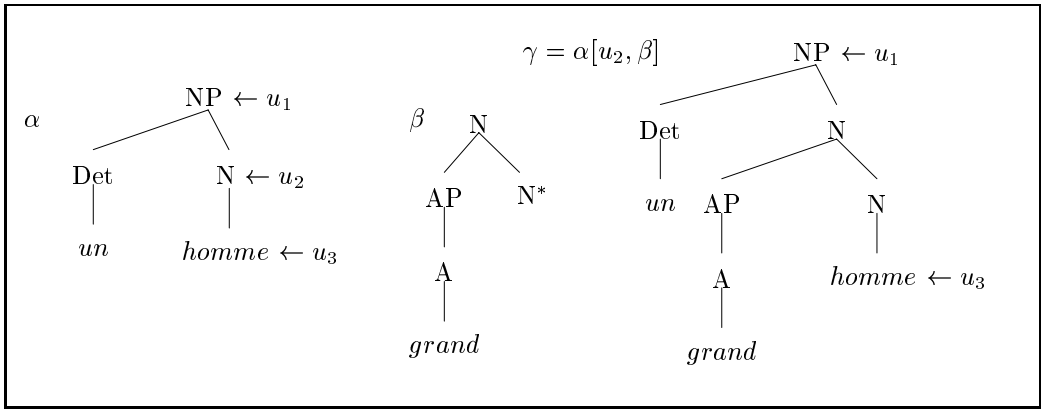


FIGURE 2.15: EXAMPLE FOR NON-MONOTONICITY IN TAG DERVATION

This problem of non-monotonicity arises because TAGs use fully specified structures. To avoid non-monotonicity, Vijay-Shanker (1992) proposes therefore to use tree descriptions, represented by so-called *quasi-trees*, instead of trees.

The idea of quasi-trees is as follows: nodes are replaced by quasi-nodes. These are objects that denote nodes, and two quasi-nodes may denote the same node. Quasi-nodes can be related by the parent relation, linear precedence or dominance. For the example of Fig. 2.15, the corresponding quasi-tree derivation is shown in Fig. 2.16. The node u_2 is replaced by two quasi-nodes n_1, n_2, and the dotted edge between them indicates a dominance relation. This dominance relation signifies that n_1 and n_2 can denote the same node (then there was no adjunction) or there can be something between n_1 and n_2 (corresponding to an adjunction) and in this case n_1 and n_2 would denote different nodes.

If adjunction is interpreted as building the conjunction of two quasi-trees and additional equivalences between quasi-nodes, e.g. $\gamma = \alpha \wedge \beta \wedge n_1 \approx n_3 \wedge n_2 \approx n_6$ in Fig. 2.16, then each

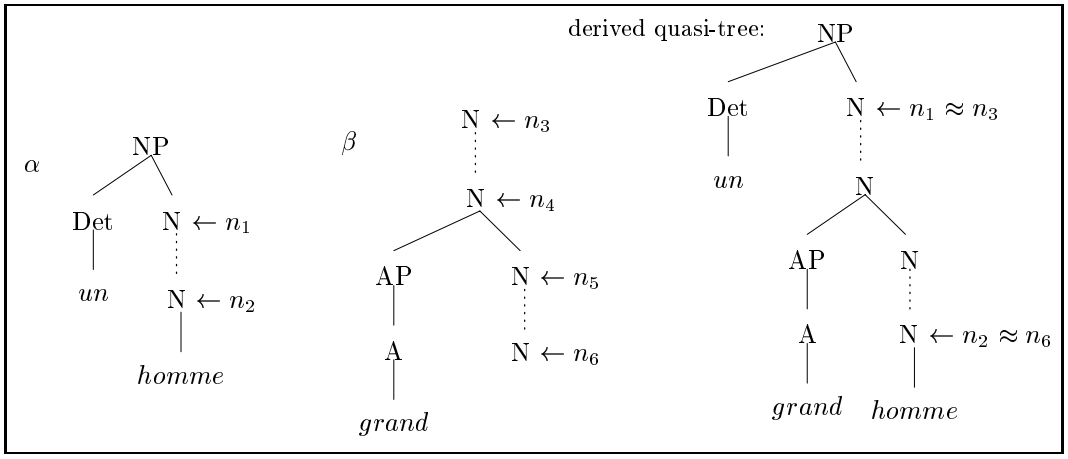


FIGURE 2.16: QUASI-TREE DERIVATION FOR *grand homme*

derivation step increases the derived tree description and decreases the set of trees satisfying this tree description.

Work on the formalization of quasi-trees for TAGs has been done by Rogers and Vijay-Shanker. There are two ways of interpreting quasi-trees. Either quasi-trees can be seen as structures, i.e. as objects. In this case they are models representing partial information, and a definition of subsumption is needed. This approach is pursued in Rogers and Vijay-Shanker 1992. Or quasi-trees can be viewed as descriptions of trees, i.e. as formulas in a tree logic. This second approach can be found in Rogers and Vijay-Shanker 1994 and Rogers 1994. An advantage of the first approach is that it is easier to check whether a quasi-tree satisfies a relation. On the other hand, quasi-trees seen as objects cannot capture many disjunctive consequences. For this reason, in their later formalization, Rogers and Vijay-Shanker take quasi-trees to be sets of formulas.

According to Rogers 1994, a quasi-tree is a partial description of trees such that the set of trees satisfying this description has a unique minimum with respect to that partial order which relates each tree to those trees that can be derived from it by adjunction. The formal definition in Rogers 1994 allows more than these intended quasi-trees, and Rogers and Vijay-Shanker (1994) suppose that additional constraints on the form of quasi-trees result from the adjunction mechanism as defined in the (constraint-based) TAG formalism.

Another problem where a shift from trees to quasi-trees might be useful concerns the representation of the lexicon, i.e. of the TAG. Vijay-Shanker and Schabes (1992) propose the use of a more general form of quasi-trees for a compact organization of the elementary structures of TAGs.

2.5.2 Description Tree Grammars

Description Tree Grammars (D-Tree Grammars or DTGs) are a TAG-variant proposed by Rambow et al. (1995). In this subsection I will present the DTG formalism in an informal way.

Rambow et al. (1995) address a problem that has already been mentioned in Section 2.1, namely that TAGs do not allow a uniform treatment of arguments. DTGs consist of quasi-tree like objects called d-trees, and the *substitution* and *adjunction* operations of TAGs are replaced by *subsertion* and *sister adjunction*. The main idea of these operations is that there should be one complementation operation (subsertion) and one modification operation (sister adjunction).

If one looks at the derivation steps shown in Fig. 2.8 on page 30 and 2.9 on page 32, then it becomes clear that these two complementation derivation steps have the following in common: taking the predicate-argument structure with open slots for the complements (α_1 in Fig. 2.8 and β in Fig. 2.9), the argument is added in the following way: first a subtree of the tree that represents the argument is substituted for the leaf representing the argument slot (in Fig. 2.8 this subtree is the whole tree α_2, and in Fig. 2.9 it is a proper subtree of γ_1). Then the rest of the tree of the argument is inserted somewhere above the subtree that was substituted for the argument slot. This operation is formalized by the definition of subsertion.

The idea underlying d-trees is the same as the one underlying quasi-trees. Each node in an elementary structure where adjunction is possible, is replaced by two different nodes. These two nodes are not connected by a normal edge representing a parent relation. Instead, they are connected by a so-called *d-edge*. Such a d-edge between nodes u_1 and u_2 signifies that either u_1 and u_2 can become equal or there can be a subtree between u_1 and u_2. In other words, these d-edges represent dominance relations.

D-trees are defined as trees with two kinds of edges, so-called *i-edges* (for immediate domination edge) and d-edges (for domination edge) such that for each node u, either u is a leaf or there is exactly one d-edge from u to another node below u or there are (finitely many) i-edges from u to other nodes (to children of u). This means that d-trees consist of fully specified trees with dominance relations between these trees. This idea will also be captured in the definition of descriptions in the next chapter, i.e. in the definition of the elementary objects of Tree Description Grammars. The crucial difference between d-trees and descriptions however is that in the case of descriptions, dominance relations between the subtrees are less restricted than in the case of d-trees. For descriptions, a leaf of one subtree may dominate more than one other subtree, and, furthermore, the root of a subtree may be dominated by more than one other leaf.

Subsertion

Subsertion is an operation taking two d-trees β and α. Roughly said, it consists of deriving a new d-tree γ by substituting a sub-d-tree of the second d-tree α for a node u in β and by inserting the rest of α into the d-edges of β. E.g. in Fig. 2.17, the subtree $\alpha(5)$ is substituted for the leaf u in β. Then $\alpha(4)$ is inserted into the d-edge between $\beta(2)$ and $\beta(3)$, and finally the subtree with $\alpha(1), \alpha(2)$ and $\alpha(3)$ is inserted into the d-edge between $\beta(1)$ and $\beta(2)$.

Subsertion is restricted by constraints called *subsertion-insertion constraints (SIC)*. Each d-edge in an elementary d-tree has such a constraint. These constraints are finite sets of nodes in elementary d-trees. They specify those nodes that are not allowed to appear between the two nodes of the d-edge the constraint is associated with.

Clearly, all substitution operations are also subsertion operations. Therefore subsertion can be used as the operation that takes place to fill argument slots in subcategorization

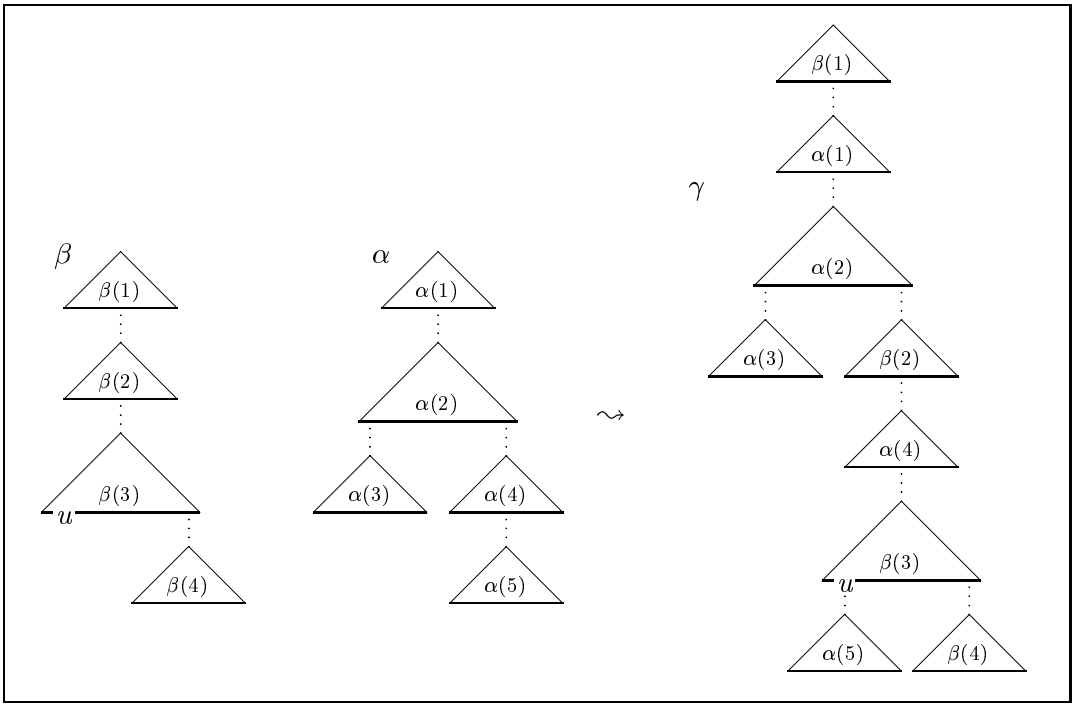


FIGURE 2.17: SUBSERTION

frames. The advantage of subsertion in contrast to substitution is that it even allows an adequate treatment of complement clauses. An analysis of (3) (see page 31) with subsertion is shown in Fig. 2.18. The d-tree α for the complement clause is added by subsertion to the d-tree β representing the argument structure of the matrix verb *penses*. (In contrast to this, see Fig. 2.9 for the analysis of (3) in a TAG.)

Sister-adjunction

The other d-tree operation, *sister adjunction*, consists of adding a new d-tree to a d-tree as right- or left-most subtree of a node with daughters. Therefore sister-adjoining only means putting the two d-trees together and adding one single i-edge. With this definition, in contrast to adjunction in TAGs, in successive derivation steps, several sister-adjunctions can occur at the same node.

Similar to adjunction constraints in TAGs, DTG sister-adjoining constraints (for nodes) specify whether a certain d-tree can be sister-adjoined at a certain node and whether it must be right- or left-adjoined.

2.5.3 Unordered Vector Grammars with Dominance Links

Unordered Vector Grammars with Dominance Links (UVG-DL), introduced in Rambow 1994a,b, can also be seen as a grammar using quasi-trees. This grammar formalism is motivated by the observation that set-local MC-TAGs are not powerful enough to handle

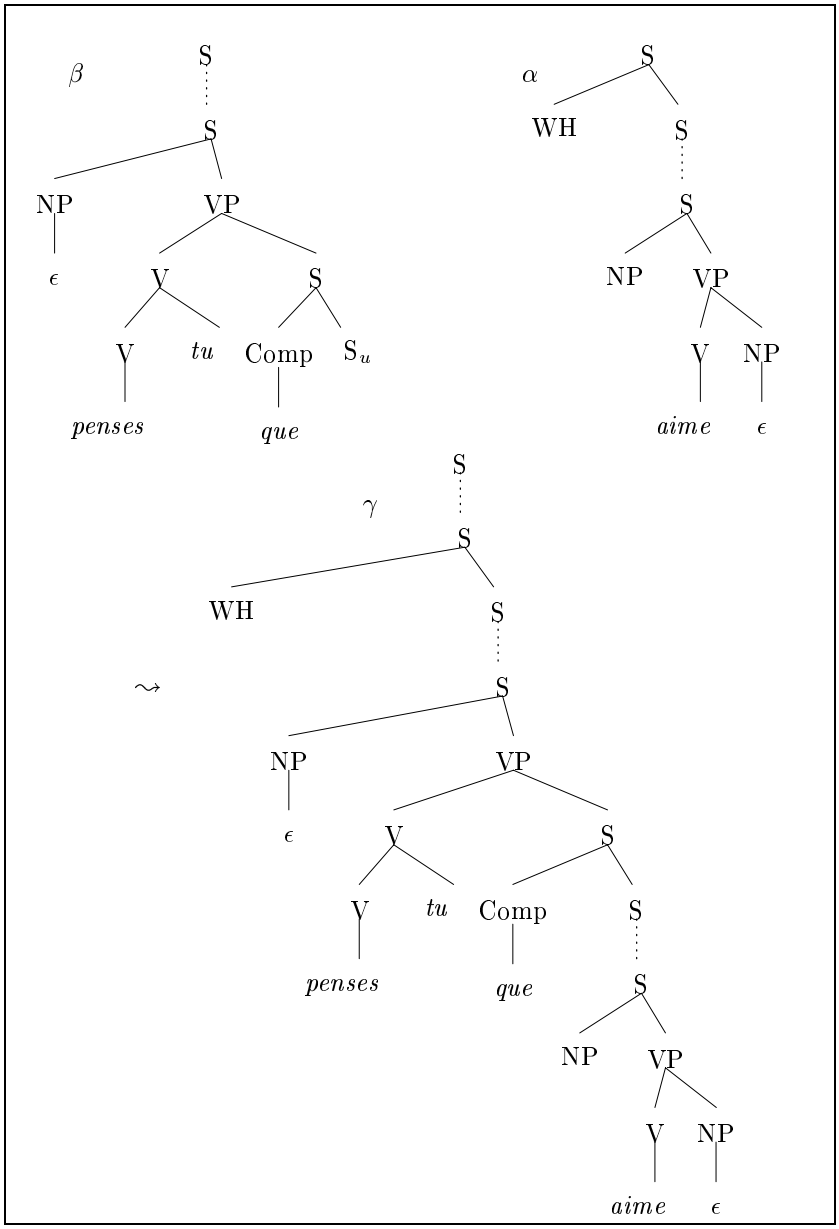


FIGURE 2.18: SUBSERTION OPERATION ADDING A D-TREE FOR THE ANCHOR OF THE COMPLEMENT CLAUSE IN *qui penses-tu que Marie aime? ('whom do you think that Mary loves?')*

$v_1 := (S' \rightarrow \textit{daß}\ VP)$, $dom_{v_1} := \emptyset$
$v_2 := (VP^{(1)} \rightarrow NP^{(1)}_{nom} VP^{(2)}, VP^{(3)} \rightarrow NP^{(1)}_{dat} VP^{(4)}, VP^{(5)} \rightarrow VP^{(6)} VP^{(7)},$
$\quad VP^{(8)} \rightarrow \textit{verspricht})$,
$dom_{v_2} := \{\langle VP^{(2)}, VP^{(8)}\rangle, \langle VP^{(4)}, VP^{(8)}\rangle, \langle VP^{(7)}, VP^{(8)}\rangle\}$
$v_3 := (VP^{(1)} \rightarrow VP^{(2)} VP^{(3)}, \quad VP^{(4)} \rightarrow \textit{zu} \quad \textit{versuchen})$,
$dom_{v_3} := \{\langle VP^{(3)}, VP^{(4)}\rangle\}$
$v_4 := (VP^{(1)} \rightarrow NP^{(1)}_{acc} VP^{(2)}, \quad VP^{(3)} \rightarrow \textit{zu} \quad \textit{reparieren})$,
$dom_{v_4} := \{\langle VP^{(2)}, VP^{(3)}\rangle\}$
$v_5 := (NP^{(1)}_{nom} \rightarrow \textit{der Meister})$, $dom_{v_5} := \emptyset$
$v_6 := (NP^{(1)}_{dat} \rightarrow \textit{niemandem})$, $dom_{v_6} := \emptyset$
$v_7 := (NP^{(1)}_{acc} \rightarrow \textit{den Kühlschrank})$, $dom_{v_7} := \emptyset$

FIGURE 2.19: SAMPLE UVG-DL

scrambling phenomena in German (see 2.4.1, page 35) whereas non-local MC-TAGs are unnecessarily powerful. UVG-DL can be interpreted as a variant of non-local MC-TAGs.

An UVG-DL consists of a nonterminal and a terminal alphabet, a nonterminal start symbol, and a set of pairs $\langle v, dom_v\rangle$ where v is a vector of context-free productions, and dom_v is a binary relation over the occurrences of nonterminals in the productions in v. The pairs in dom_v are called *dominance links*. In order to distinguish between different occurrences of the same nonterminal, the nonterminals in the productions are equipped with superscripts (i) for the i-th occurrence of a certain nonterminal. The relation dom_v must be such that if $\langle A^{(i)}, B^{(j)}\rangle \in dom_v$, then $A^{(i)}$ occurs in the right-hand side of some production in v, and $B^{(j)}$ is the left-hand symbol of some production in v.

The language $L(G)$ generated by some UVG-DL G is defined as follows: $w \in L(G)$ if w is a terminal string derived from the start symbol by applying productions occuring in the vectors of G such that

1. there are vectors $v_1, \ldots, v_m \in V$ and $n_i \in \mathbb{N}$ for $1 \leq i \leq m$ such that in the course of the derivation of w there were exactly n_i applications of each production in v_i, and
2. the dominance links must hold in the derivation tree of w (here derivation tree is the context-free derivation tree).

The second condition signifies the following: if γ is the derivation tree of w, then for each i, $1 \leq i \leq m$, the n_i application of each production in v_i can be ordered in such way that the following holds: if there are productions $p = A^{(i)} \rightarrow w_1 B^{(j)} w_2$ and $q = C^{(k)} \rightarrow u$ in v_i, and if $\langle B^{(j)}, C^{(k)}\rangle \in dom_{v_i}$, then for all l, $1 \leq l \leq n_i$: if u is the node with label B that is on the right side of the l-th application of p and u' is the node with label C that is on the left side of the l-th application of q then $\langle u, u'\rangle \in \mathcal{D}_\gamma$ holds.

An example is the UVG-DL shown in Fig. 2.19 taken from Rambow 1994b with pairs $\langle v_1, dom_{v_1}\rangle, \ldots, \langle v_7, dom_{v_7}\rangle$ (for a definition of this UVG-DL with tree descriptions see Fig. 3.22).

Figure 2.20 shows a sample derivation for (8) with the UVG-DL in Fig. 2.19. A subscript

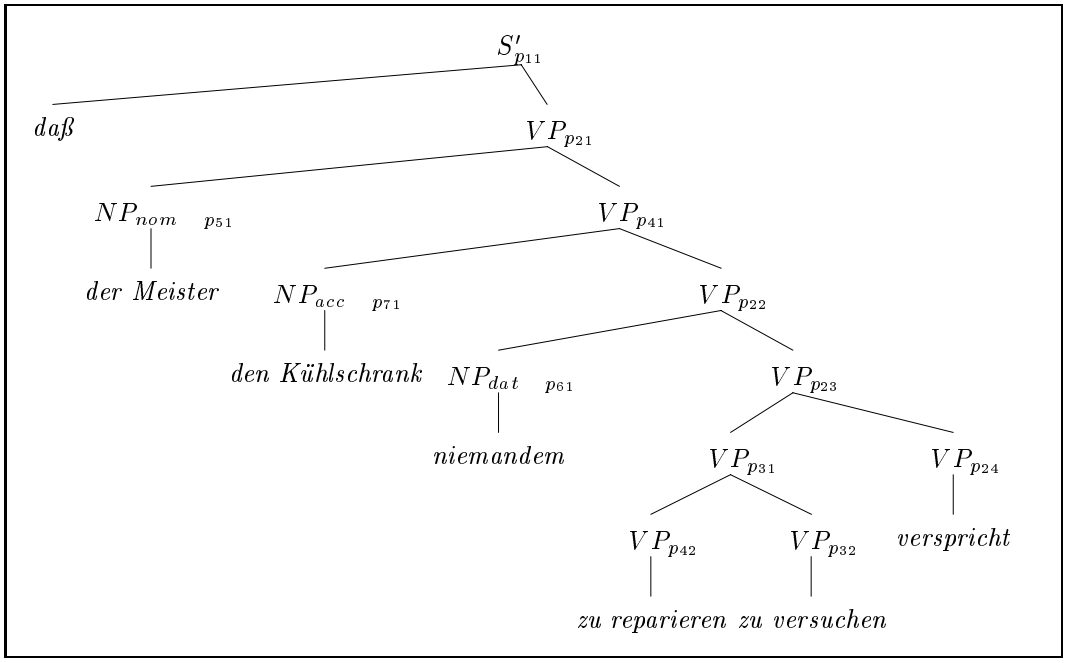


FIGURE 2.20: SAMPLE UVG-DL DERIVATION TREE

p_{ij} in this figure indicates that this node represents the left-hand symbol of the j-th production in v_i.

(8) ... daß der Meister den Kühlschrank niemandem zu reparieren
... that the boss$_{nom}$ the refrigerator$_{acc}$ no-one to repair
zu versuchen verspricht
to try promises
... that the boss promises no-one to try to repair the refrigerator

A definition of the UVG-DL in Fig. 2.19 based on tree descriptions will be shown in Fig. 3.22 on page 102 in Chapter 3.

UVG-DLs show that the introduction of dominance links gives a suitable way to deal with free word order languages such as German. However, Rambow suspects that certain formal languages relevant to natural language syntax cannot be derived in an UVG-DL. In Rambow 1994a, Conjecture 7, p. 78 he argues that it might not be possible to generate the copy-language $\{ww \mid w \in \{a,b\}^*\}$ with UVG-DLs. The difference between UVG-DLs and non-local MC-TAGs is that the application of the productions of a vector in an UVG-DL need not take place simultaneously. This seems to be an important point in the case of the copy language. Using MC-TAGs, this language can be easily generated, whereas with an UVG-DL the order of the application of productions is perhaps too free. If this is true, then this formalism is not adequate to handle Dutch cross-serial dependencies.

To remedy this possible shortcoming of UVG-DLs, Rambow proposes a tree-rewriting variant of UVG-DLs called *Vector Multi-Component Tree Adjoining Grammars with Dominance Links* (V-TAGs, see also Rambow and Lee 1994).

2.6 Conclusion: TAGs and underspecification

In this section the approaches presented above, in particular those based on tree descriptions, are considered with respect to the following question: what properties should a grammar formalism have if it is supposed to allow underspecified representations and, furthermore, to preserve the attractive properties of TAGs and related formalisms?

We have seen that the main idea underlying the shift from context-free grammars to TAGs was an extension of the domain of locality. The advantages of this extension are an increase of the generative capacity, and the possibility to lexicalize a grammar for natural languages without violating the linguistic principles observed in TAGs.

Furthermore, TAGs have been developed as a formalism that is supposed to predict some properties of natural languages. If this view is adopted, then a grammar formalism should in particular be restricted in such way that they generate only semilinear languages.

The requirement that underspecified representations should be allowed, addresses another problem: the relevant criterion here is neither the set of (indexed) strings nor the set of trees generated by the grammar but the way these trees or strings are represented. A tree-generating grammar allowing underspecification must be such that the objects derived by this grammar are descriptions of trees and these descriptions might describe more than one minimal tree. (A suitable definition of minimal trees will be given later. For the considerations here it is sufficient to think of a minimal tree, similar to the definition of *minimal referents* in Rogers 1994, as an object satisfying the actual description ϕ without satisfying any description derivable from ϕ.)

A formalism such as TAG using fully specified trees is obviously not able to generate underspecified representations. In order to capture more than one tree by one derived underspecified representation, the grammar itself must consist of underspecified representations. This is the case for quasi-tree based grammars like D-Tree Grammars and UVG-DLs. However, in both formalisms only the parent relation might be underspecified. But scope relations are transitive and therefore cannot be represented by the parent relation. They must be represented by the dominance relation, i.e. the partial tree descriptions must even allow an underspecification of the dominance relation.

To illustrate this, we will consider the following scope ambiguity:

(9) every man loves a woman

(9) contains two quantifiers, *every man* and *a woman*, and (9) is ambiguous because the relative scope of the two quantifiers is underspecified. The two possible scope orders can be represented by the two first order terms:

- $\forall x((man(x)) \rightarrow (\exists y((woman(y)) \wedge (loves(y)(x)))))$, and
- $\exists y((woman(y)) \wedge (\forall x((man(x)) \rightarrow (loves(y)(x)))))$.

If a bracketed expression is interpreted as a subtree, then the two trees γ_1 and γ_2 in Fig. 2.21 correspond to these two readings. The difference between γ_1 and γ_2 is the following: in γ_1, the parent node of the universal quantifier dominates the parent node of the existential quantifier while in γ_2 the parent node of the existential quantifier dominates the parent node

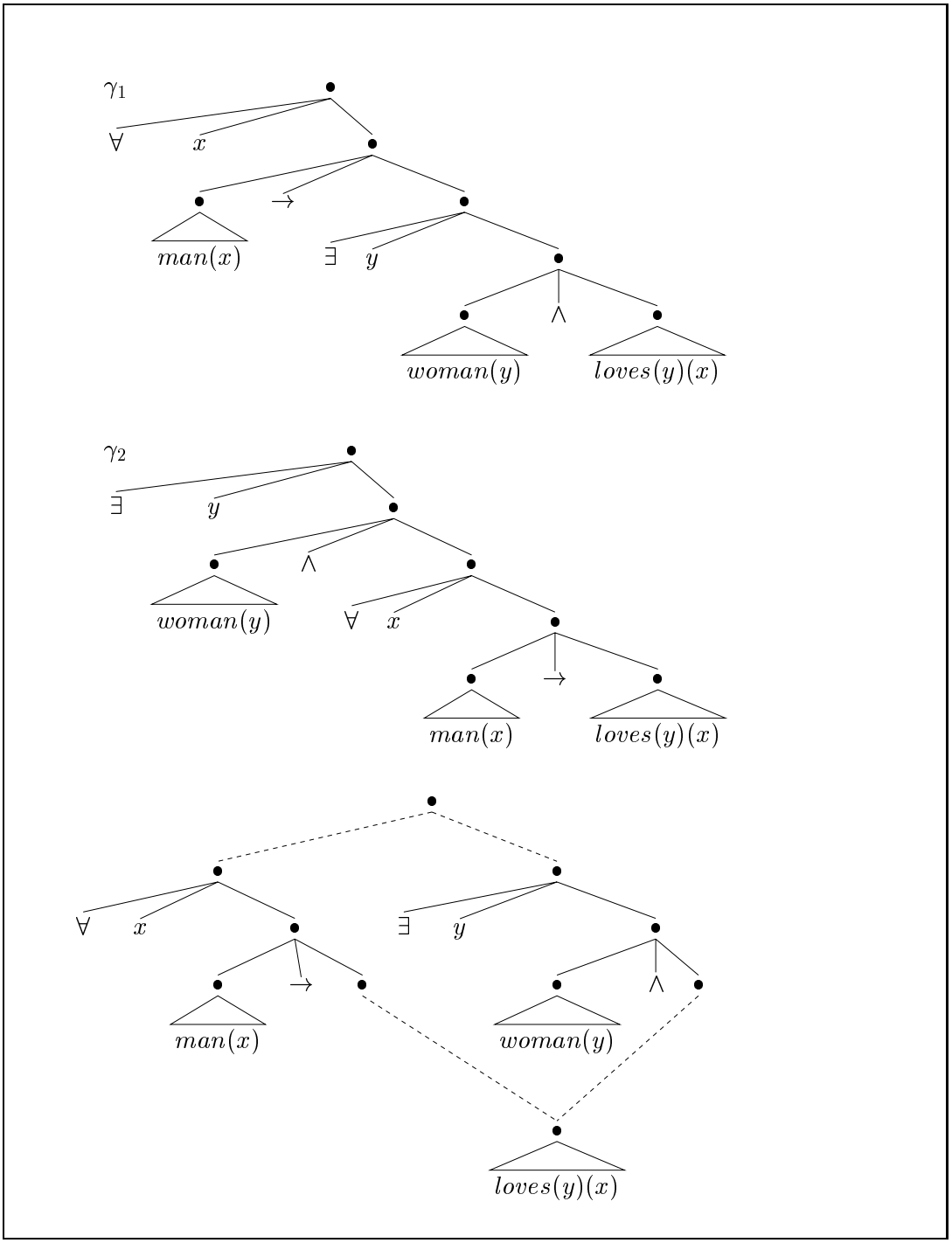


FIGURE 2.21: SCOPE ORDERS AND UNDERSPECIFIED REPRESENTATION FOR *every man loves a woman*

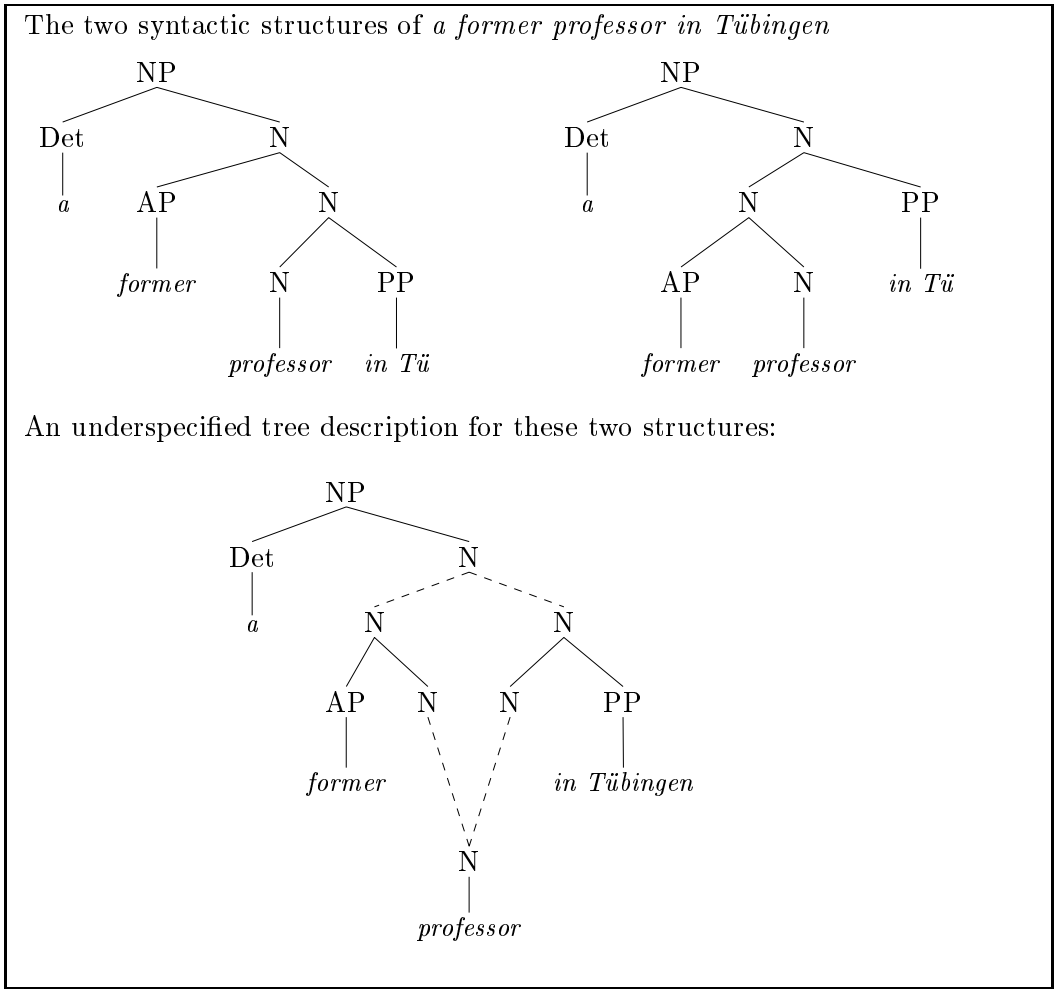


FIGURE 2.22: STRUCTURAL UNDERSPECIFICATION

of the universal quantifier. In other words, γ_1 and γ_2 differ not only in the parent relation but also in the dominance relation. Therefore, a partial description for these two trees must be such that dominance is not fully specified. Fig. 2.21 shows such a tree description. The two trees in Fig. 2.21 are the corresponding two minimal trees. This underspecified representation is similar to those proposed by Muskens (1995) for ambiguous logical forms where underspecified scope relations are also represented by underspecified dominance relations in a tree.

An example of syntactic scope ambiguities where an underspecified representation can be obtained by a relaxation of the dominance relation, was already given in (2) in the introduction. It is repeated here as (10):

(10) a former professor in Tübingen

(10) is ambiguous with respect to its syntactic structure, and, corresponding to this ambiguity, the following two readings are possible:

1. either the NP describes a person that was a professor in Tübingen in some former days,
2. or the NP describes a person that is in Tübingen and that was a professor in some former days.

The two syntactic structures of (10) and an underspecified tree description representing these two structures are shown in Fig. 2.22.

Clearly, representations with an underspecified dominance relation cannot be derived by the TAG-extensions using partial descriptions of trees that were introduced in this chapter. Consequently these formalisms are not able to generate appropriate underspecified representations for scope ambiguities. Instead, a new formalism must be found that allows an underspecified dominance relation and that retains the attractive properties of the TAG variants presented here.

Therefore, in the following two chapters, a TAG-extension will be proposed that is based on the work on quasi-trees by Vijay-Shanker and Rogers and that satisfies the following criteria:

1. it provides an extended domain of locality,
2. it is powerful enough to deal with natural languages, in particular powerful enough to generate the indexed string languages $\{w^{ind}w^{ind} \mid w \in T^*\}$ and SCR^{ind},
3. it generates only semilinear languages,
4. and it allows underspecification not only of the parent relation but also of the dominance relation.

Chapter 3

Tree Description Grammars

The goal of this chapter is to develop an extension of Tree Adjoining Grammars that allows underspecified representations. As we have seen in the previous chapter, a very natural way to introduce underspecification into a tree generating grammar is a relaxation of the dominance relation. The TAG-variants that deal with partial descriptions of trees, namely TAGs based on quasi-trees, D-Tree Grammars and Unordered Vector Grammars with Dominance Links only allow a relaxation of the parent relation but not of the dominance relation in trees. Starting from the work on quasi-trees, a new grammar formalism is presented in the course of this chapter that allows underspecification of the dominance and (as a consequence of this) of the linear precedence relation. These grammars are called *Tree Description Grammars (TDG)*. TDGs also have been presented in Kallmeyer 1996a,b, 1999b.

In principle the idea of TDGs is the following: TDGs consist of tree descriptions, so-called *elementary* descriptions, i.e. TDGs are description-based. Similar to the TAG-variants with tree descriptions presented in the last chapter, TDGs are also derivation-based. This means that starting from the elementary descriptions, more complex descriptions are derived. Each derivation step can be viewed as a conjunction of two formulas and additional equivalences of node names. In contrast to quasi-trees, the dominance relation need not be fully specified in these tree descriptions.

In Fig. 2.21 on page 50 and 2.22 on page 51 we have already seen how underspecified representations could look like which are obtained by relaxing the dominance relation. In a quasi-tree based framework, a derivation of the underspecified tree description proposed in Fig. 2.22 for *former professor in Tübingen* could be of the form shown in Fig. 3.1. In this figure, starting with a tree description γ_1 for *former professor*, the underspecified representation γ_2 is obtained by adding an elementary tree description β for a PP with proposition *in*. In a further derivation step, the elementary description for *Tübingen* must be added.

This example shows the necessity of two conditions on derivation in TDGs. Firstly, there are only certain node names in the new elementary descriptions that occur in the equivalences that are added in this step. If $n_2 \approx n_7$ was also added, the tree description would still be satisfiable. But this is not intended. Therefore, in TDGs it will be possible to mark certain node names in an elementary description such that only for the minimal name (the name dominating all other names, e.g. n_5 in β) and for the marked names (e.g. n_8 in β) new equivalences are added. Marked names are comparable to foot nodes in TAGs since in a derivation step they indicate those parts of the new elementary description that must be

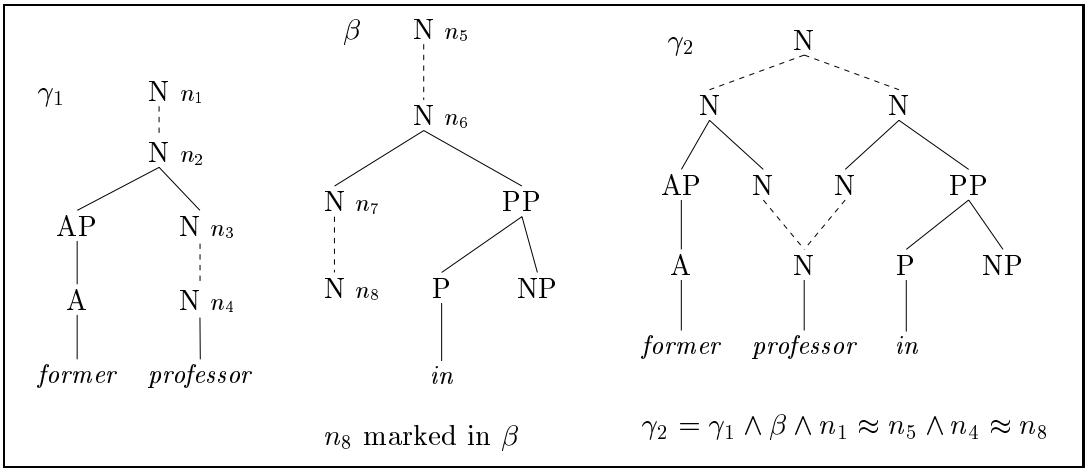


FIGURE 3.1: DERIVATION OF AN UNDERSPECIFIED TREE DESCRIPTION

connected to the old description. A second derivation condition is necessary to select the "most underspecified" derivation result. In the case of γ_1 and β, it would be also possible to add $n_1 \approx n_5 \wedge n_2 \approx n_8$ as new equivalences or $n_3 \approx n_5 \wedge n_4 \approx n_8$. However, these descriptions are not desired since each of them describes only one of the two structures in Fig. 2.22. In contrast to this, the result $\gamma_2 = \gamma_1 \wedge \beta \wedge n_1 \approx n_5 \wedge n_4 \approx n_8$ is satisfied by both structures in Fig. 2.22. When defining the derivation in TDGs, we will see how such a condition of "maximal underspecification" can be formalized.

There are several aspects that must be taken into account for the definition of TDGs. An important question concerns the nature of tree descriptions: are they objects or descriptions (formulas) and how can they be defined? Following Rogers and Vijay-Shanker 1994 and Rogers 1994, I view tree descriptions as formulas in a quantifier-free first order logic. We will see that for TDGs the quasi-tree definition of Rogers and Vijay-Shanker 1994 and Rogers 1994 is not appropiate. Therefore I will introduce so-called *descriptions*. These are formulas that, roughly said, describe certain subtrees together with dominance relations between these subtrees. E.g. γ_1, β and γ_2 in Fig. 3.1 represent descriptions. A TDG then consists of descriptions, and the result of a derivation step in a TDG must also be a description.

After having defined TDGs, a suitable derivation mode must be found. Derivation in TDGs should be such that multicomponent derivations and derivations of underspecified representations are possible. As already mentioned, the idea (as for quasi-trees) is that in each derivation the conjunction of two descriptions and additional equivalences of node names is built. The choice of node names used for these new equivalences can be controlled by marking these names. Furthermore, as illustrated by the example from Fig. 3.1, the result of such a derivation step must be "maximally underspecified".

Finally, since the grammar generates tree descriptions and each tree description is satisfied by an infinite number of trees, a notion of *minimal tree* must be introduced. The idea of the minimal tree is the following: a tree is minimal for a given description if it satisfies this description in such a way that all parent relations in the tree already occur in the description. E.g. the tree description γ_2 in Fig. 3.1 has two minimal trees, the two trees shown in Fig.

2.22. For a given description, a minimal tree can be obtained by adding equivalences between node names that are connected by a dominance relation in the description. The result is a fully specified description of a minimal tree. Rogers and Vijay-Shanker 1994 and Rogers 1994 also define some kind of minimal model for a quasi-tree, the so-called *circumscriptive reading*, but this does not capture the idea of minimal tree.

This chapter is structured as follows: first the tree logic used in TDGs is presented, a simple quantifier-free first order logic mainly based on Rogers 1994. Decidability, soundness and completeness of this logic will be shown. Then descriptions are introduced in Section 3.2 and the TDG formalism is defined. In Section 3.3, I will give a decidable derivation procedure for TDGs, and the decidability of the word problem for lexicalized TDGs will be shown. Finally, in the last section, I will give some examples in order to show that underspecified representations, subsertion-like derivation steps and multicomponent derivations are possible in a TDG.

3.1 A tree logic for TDGs

There are two alternatives with respect to the nature of tree descriptions: either they can be seen as objects (as in Rogers and Vijay-Shanker 1992) or as formulas, i.e. as expressions in a tree logic, as it is the case in Rogers and Vijay-Shanker 1994 and Rogers 1994. I follow the second approach by defining TDGs as sets of formulas.

In this section, the underlying tree logic is introduced together with definitions of satisfiability and of semantic and syntactic consequence. Soundness, completeness and decidability of the logic with respect to these consequence notions are shown. The tree logic is very similar to the one proposed in Chapter 3 of Rogers 1994 for the definition of quasi-trees.

3.1.1 Syntax

A TDG consists of tree descriptions, i.e. of constraints for finite labelled trees. In these trees, leaves are labelled by terminal symbols and internal nodes by finite atomic feature structures, i.e. sets of attribute value pairs. The logic of tree descriptions used for TDGs is a quantifier-free first order logic. It differs from the one proposed in Rogers 1994 in so far as it allows not only the formulation of constraints concerning the tree structure but also the formulation of constraints for the labels of the nodes.

Definition 3.1 ((A,V,T)-formulas) *Let A (the attribute names), V (the attribute values) and T (the terminals) be pairwise disjoint finite sets. (A,V,T)-formulas are words over the following symbols:*

- *the symbols of $V \cup A \cup T \cup \{\epsilon, \bot\}$*
- *a nonempty countable set K of variables called node names,*
- *constants $\lhd, \lhd^*, \prec, \approx$ (for binary relations),*
- *a constant δ (for the labeling function),*
- *logical connectives $\wedge, \vee, \neg$ and*
- *bracket symbols (,).*

(A,V,T)-formulas are inductively defined:

1. *(A,V,T)-terms:*
   (a) *τ is an (A,V,T)-l-term (denoting labels) iff either $\tau \in T \cup \{\epsilon\}$ or there is a $k \in K$ with $\tau = \delta(k)$.*
   (b) *τ is an (A,V,T)-v-term (denoting attribute values) iff either $\tau \in V \cup \{\bot\}$ or there are $a \in A$ and $k \in K$ such that $\tau = a(\delta(k))$.*
   (c) *k is an n-term (denoting nodes) iff $k \in K$.*
2. *(A,V,T)-formulas:*
   (a) *For all τ_1, τ_2, that are either both (A,V,T)-l-terms or both (A,V,T)-v-terms or both n-terms, $\tau_1 \approx \tau_2$ is an (A,V,T)-formula.*
   (b) *For alle n-terms k_1, k_2: $k_1 \lhd k_2$, $k_1 \lhd^* k_2$ and $k_1 \prec k_2$ are (A,V,T)-formulas.*
   (c) *If ϕ is an (A,V,T)-formula, then $\neg\phi$ also is an (A,V,T)-formula.*
   (d) *If ϕ, ψ are (A,V,T)-formulas, then $(\phi \wedge \psi)$ and $(\phi \vee \psi)$ are also (A,V,T)-formulas.*
   (e) *Nothing else is an (A,V,T)-formula.*

(A,V,T)-formulas containing at most one negation and no other logical connective are called elementary (A,V,T)-formulas.
$node(\phi)$ is the set of all node names occuring in an (A,V,T)-formula ϕ.

If the order of logical connectives does not matter as in the case of conjunctions of more than two formulas, or if there is only one possible order, brackets will be omitted.

The constants $\lhd, \lhd^*$ and $\prec$ denote the parent, the dominance and the linear precedence relation respectively.

As an example consider the tree description graphically represented as β in Fig. 3.1. If an attribute *cat* is assumed for syntactic categories, this description is captured by the following (A,V,T)-formula ϕ_β with attributes $A = \{cat\}$, attribute values $V = \{N, PP, P, NP\}$ and terminals $T = \{in\}$:

$$\begin{aligned}\phi_\beta \;=\; & k_1 \lhd^* k_2 \wedge k_2 \lhd k_3 \wedge k_3 \lhd^* k_4 \wedge k_2 \lhd k_5 \wedge k_5 \lhd k_6 \wedge k_6 \lhd k_7 \wedge k_5 \lhd k_8 \\ & \wedge k_3 \prec k_5 \wedge k_6 \prec k_8 \\ & \wedge cat(\delta(k_1)) \approx N \wedge cat(\delta(k_2)) \approx N \wedge cat(\delta(k_3)) \approx N \wedge cat(\delta(k_4)) \approx N \\ & \wedge cat(\delta(k_5)) \approx PP \wedge cat(\delta(k_6)) \approx P \wedge \delta(k_7) \approx in \wedge cat(\delta(k_8)) \approx NP\end{aligned}$$

Since the syntax of the formulas shows which symbols are attribute names, which are attribute values and which are terminal symbols, (A,V,T) will be often omitted.

In contrast to ordinary predicate logic, this logic has three kinds of terms that can be syntactically distinguished: the l-terms denoting labels of nodes, v-terms denoting attribute values and n-terms denoting nodes.

Node names are free variables and not constants, as it is the case in the logic of Rogers 1994. This choice is motivated by the following observation: if the denotation of the node names is given by an interpretation function that is part of the model, i.e. the tree, as in Rogers 1994, then two formulas that only differ in a renaming of the nodes (a bijection of K) are satisfied by different models. In the case of TDGs, formulas differing only in a node renaming shall be equivalent in the sense that their models are the same. Therefore node names are treated as free variables. In this case formulas that are equivalent up to a node renaming, are satisfied by the same trees (only with respect to different assignment functions).

3.1.2 Semantics

The set of intended models for the tree logic introduced above is restricted to terminal finite labelled trees, i.e. finite labelled trees, where each leaf is labelled by a terminal symbol or the empty word and each internal node by a nonterminal.[1] (For the corresponding definitions, see Def. 2.8 on page 16 and Def. 2.10 on page 17 in Chapter 2.) As nonterminal labels I will use finite atomic feature structures.

Definition 3.2 (Finite atomic feature structures) *Let A and V be finite disjoint sets. A finite atomic feature structure is a set $F \subseteq A \times V$ such that*

1. *$F \neq \emptyset$, and*
2. *for each $a \in A$ there is at most one $v \in V$ with $\langle a, v\rangle \in F$.*

A is called the set of attributes and V is called the set of attribute values.
Notation: a set N of finite atomic feature structures with attributes A and values V is written as $N_{A,V}$.

Note that these feature structures are not allowed to be empty, i.e. each feature structure must have at least one attribute value pair.

The notion of satisfiability in this logic is the same as in ordinary first-order predicate logic. Satisfiability of a formula in a model is considered with respect to a node assignment function mapping all node names in K onto nodes in the model.

Definition 3.3 (Satisfiability) *Let $B = (\mathcal{U}, u_0, \mathcal{P}, \mathcal{D}, \mathcal{L}, \mu)$ be a finite labelled tree with finite atomic feature structures $N_{A,V}$ as nonterminals and terminals T. Let $g : K \to \mathcal{U}$ be a function, the node assignment function, and let ϕ, ϕ_1 and ϕ_2 be (A, V, T)-formulas.*
B satisfies ϕ wrt g (written $B \models_g \phi$) is inductively defined:

1. *For all $k_1, k_2 \in K$:*
   - $B \models_g k_1 \lhd k_2$ *iff* $(g(k_1), g(k_2)) \in \mathcal{P}$
   - $B \models_g k_1 \lhd^* k_2$ *iff* $(g(k_1), g(k_2)) \in \mathcal{D}$
   - $B \models_g k_1 \prec k_2$ *iff* $(g(k_1), g(k_2)) \in \mathcal{L}$
   - $B \models_g k_1 \approx k_2$ *iff* $g(k_1) = g(k_2)$
2. *For all l-terms l_1, l_2 (l-terms can either be in $T \cup \{\epsilon\}$ or they have the form $\delta(k)$):*
   $B \models_g l_1 \approx l_2$ *iff* $[\![l_1]\!]_g = [\![l_2]\!]_g$ *where*
   - $[\![\delta(k)]\!]_g = \mu(g(k))$ *for all* $k \in K$,
   - $[\![x]\!]_g = x$ *for all* $x \in T \cup \{\epsilon\}$.
3. *For all v-terms v_1, v_2 (v-terms can either be in $V \cup \{\bot\}$ or they have the form $a(\delta(k))$):*
   $B \models_g v_1 \approx v_2$ *iff* $[\![v_1]\!]_g = [\![v_2]\!]_g$ *where*
   - *for all $k \in K$ and all $a \in A$: if there is a $v \in V$ with $\langle a, v\rangle \in [\![\delta(k)]\!]_g$, then $[\![a(\delta(k))]\!]_g = v$, else $[\![a(\delta(k))]\!]_g = \bot$.*
   - $[\![v]\!]_g = v$ *for all* $v \in V \cup \{\bot\}$.

[1] As suggested by Reinhard Muskens, for the definition of TDGs the restriction to finite trees could even be omitted, since the only trees we are concerned with are the minimal trees of tree descriptions and these trees are always finite because of the way minimal trees will be defined.

4. $B \models_g (\phi_1 \wedge \phi_2)$ *iff* $B \models_g \phi_1$ *and* $B \models_g \phi_2$.
5. $B \models_g (\phi_1 \vee \phi_2)$ *iff* $B \models_g \phi_1$ *or* $B \models_g \phi_2$.
6. $B \models_g \neg\phi_1$ *iff not* $B \models_g \phi_1$.
$B \models \phi$ *iff there is an assignment function* g*, such that* $B \models_g \phi$.
$\phi_1 \models \phi_2$ *iff for all trees* B *and all assignments* g*: if* $B \models_g \phi_1$ *then* $B \models_g \phi_2$.

Henceforward the adjective "finite" will be usually omitted, i.e. "tree" always stands for "finite tree".

3.1.3 Syntactic consequence

In this subsection I will define some syntactic equivalent for the semantic consequence relation "$\models$". This can be done by using inference rules as proposed by Rogers. The idea is as follows: starting from a formula ψ by applying inference rules it is possible to construct a conjunctive normal form of all the consequences of ψ. A syntactic notion of consequence can then be defined for formulas ψ_1, ψ_2 by comparing the two conjunctive normal forms of the consequences of ψ_1 and ψ_2.

This way to define a syntactic consequence was very much influenced by Rogers' work. In Rogers 1994, besides defining such inferences rules, he introduces the notion of a saturated conjunction of disjunctions, which roughly said is a conjunctive normal form that represents all consequences of a formula. Starting from Rogers' work, it is quite straightforward to define the syntactic consequence $\phi_1 \vdash \phi_2$ by comparing the corresponding two saturated conjunctive normal forms of ϕ_1 and ϕ_2.

The syntactic consequence defined in this way is decidable, sound and even complete as will be shown below. Later, when defining TDGs, this notion of syntactic consequence is used for the definition of the derivation mode. Therefore it is important to show not only its soundness and completeness but also its decidability.

Instead of conjunctions of disjunctions of formulas, I will deal with sets of sets of formulas and interprete the whole set (the *branch*) conjunctively and the elements (the *clauses*) disjunctively. This is only a different notation adopted from Rogers which is used in order to see more clearly how inference rules work and in order to facilitate the definition of the syntactic consequence. The use of branches and clauses allows a definition of a syntactic consequence via a subset relation between clauses.

Definition 3.4 (Clause and branch) *Let* B *be a tree and* g *an assignment function.*
1. A clause is a set of formulas $\phi_1, \phi_2, \cdots$*. Notation:* $[\phi_1, \phi_2, \cdots]$
$B \models_g [\phi_1, \phi_2, \cdots]$ *iff there is a* $\phi \in \{\phi_1, \phi_2, \cdots\}$ *such that* $B \models_g \phi$.
2. A branch is a set of clauses $\Phi_1, \Phi_2, \cdots$*. Notation:* $\|\Phi_1, \Phi_2, \cdots\|$
$B \models_g \|\Phi_1, \Phi_2, \cdots\|$ *iff* $B \models_g \Phi$ *for all* $\Phi \in \{\Phi_1, \Phi_2, \cdots\}$.

This definition does not exclude infinite disjunctions or conjunctions. But for our purposes we only need finite clauses and branches.

The inference rules needed for the tree logic for TDGs are very similar to those proposed by Rogers (1994). But Rogers uses these kind of rules without explicitely defining the way they must be applied. Therefore, before introducing specific rules, I will give a general definition of the syntax and semantics of inference rules.

Definition 3.5 (Inference rule) *Let* $V_\Gamma := \{\Gamma_1, \Gamma_2, \cdots\}$, $V_\Phi := \{\Phi_1, \Phi_2, \cdots\}$ *and* $V_\phi := \{\phi_1, \phi_2, \cdots\}$ *be pairwise disjoint countable sets.*

1. *Syntax of inference rules:*
   - (i) *A branch schema can be inductively defined:*
     - (a) *All* $\phi_i \in V_\phi$ *are formula schemata, all formula of the tree logic are formula schemata, and for all formula schemata* F_1 *and* F_2*:* $\neg F_1$ *and* $(F_1 \wedge F_2)$ *and* $(F_1 \vee F_2)$ *are formula schemata.*
       Nothing else is a formula schema.
     - (b) *Each formula schema and each* $\Phi_i \in V_\Phi$ *is a clause schema, and for all clause schemata* C_1 *and* C_2*:* C_1, C_2 *is a clause schema.*
       Nothing else is a clause schema.
     - (c) *For each clause schema* C*,* $[C]$ *is a branch schema, each* $\Gamma_i \in V_\Gamma$ *is a branch schema, and for all branch schemata* B_1 *and* B_2*:* B_1, B_2 *is a branch schema.*
       Nothing else is a branch schema.
   - (ii) *An inference rule consists of two branch schemata* B_{old} *and* B_{new} *such that all* $x \in V_\Gamma \cup V_\Phi \cup V_\phi$ *occuring in* B_{new} *also occur in* B_{old}*.*
     Notation: if $\langle name \rangle$ *is the name of the rule, then the rule is notated as* $\frac{B_{old}}{B_{new}}\langle name \rangle$*.*
2. *Semantics: Let* $\|\Gamma\|$ *be a branch.*
   - (i) $\|\Gamma\|$ *satisfies a branch schema* B*, iff there is a homomorphism* f *such that*
     - $f(\Gamma_i)$ *is a branch for all* $\Gamma_i \in V_\Gamma$*,* $f(\Phi_i)$ *is a clause for all* $\Phi_i \in V_\Phi$ *and* $f(\phi_i)$ *is a formula for all* $\phi_i \in V_\phi$*.*
     - *for all formula* F*:* $f(F) = F$*.*
     - *for all* $\Phi_i \in V_\Phi$*:* $f([\Phi_i]) = \{f(\Phi_i)\}$*.*
     - *for all formula schemata* F_1 *and* F_2*:* $f(\neg F_1) = \neg f(F_1)$ *and* $f((F_1 \wedge F_2)) = (f(F_1) \wedge f(F_2))$ *and* $f((F_1 \vee F_2)) = (f(F_1) \vee f(F_2))$*.*
     - *for all formula schemata* F*:* $f([F]) = \{\{f(F)\}\}$*.*
     - *for each formula schema* F *and each clause schema* C*:* $f([F, C]) = \{\{f(F)\} \cup \{\phi \mid$ *there is a* $\Phi \in f([C])$ *with* $\phi \in \Phi\}\}$*.*
     - *for each* $\Phi_i \in V_\Phi$ *and each clause schema* C*:* $f([\Phi_i, C]) = \{f(\Phi_i) \cup \{\phi \mid$ *there is a* $\Phi \in f([C])$ *with* $\phi \in \Phi\}\}$*.*
     - *for all branch schemata* B_1, B_2*:* $f(B_1, B_2) = f(B_1) \cup f(B_2)$*.*
     - $f(B) = \|\Gamma\|$*.*
   - (ii) *A rule* $\frac{B_{old}}{B_{new}}\langle name \rangle$ *can be applied to* $\|\Gamma\|$*, if there is a function* f *such that* $\|\Gamma\|$ *satisfies* B_{old} *with* $f(B_{old}) = \|\Gamma\|$*. The result of the application of* $\langle name \rangle$ *to* $\|\Gamma\|$ *is then* $f(B_{new})$*.*

The following inference rules in principle correspond to those proposed by Rogers (1994). The logical and structural rules and the resolution rule *res* are exactly as in Rogers 1994 and the tree axiom rules are similar to those in Rogers 1994. Only label and parts of the equality rules do not figure there because Rogers does not consider labelled trees.

Of course only those properties of the models can be considered in inference rules that can be expressed in the tree logic. Properties that cannot be formulated in this logic are the

finiteness of the trees, the uniqueness of the root and the requirement that each leaf must be labelled by a terminal symbol.

For two terms τ_1, τ_2, I will often use an abbreviation for negated equivalence writing $\tau_1 \not\approx \tau_2$ instead of $\neg\tau_1 \approx \tau_2$.

Definition 3.6 (Inference rules for the tree logic) *Let Γ be in V_Γ, Φ, Ψ be in V_Φ and ϕ, ψ be in V_ϕ. Let $A = \{a_1, \cdots a_m\}$ and $V = \{v_1, \cdots, v_l\}$ be alphabets. Let T be an alphabet with $\epsilon \notin T$. $T_\epsilon := T \cup \{\epsilon\} = \{t_1, \cdots, t_n\}$ (ϵ is one of the t_i here which is unusual but which facilitates the formulation of the inference rules).*
The inference rules for the tree logic with respect to T, A and V are the following:

1. *Logical rules (elimination of logical connectives)*

$$\frac{\Gamma, [\Phi, \neg\neg\phi]}{\Gamma, [\Phi, \phi]}\neg\neg \qquad \frac{\Gamma, [\Phi, (\phi \vee \psi)]}{\Gamma, [\Phi, \phi, \psi]}\vee \qquad \frac{\Gamma, [\Phi, \neg(\phi \vee \psi)]}{\Gamma, [\Phi, \neg\phi], [\Phi, \neg\psi]}\neg\vee$$

$$\frac{\Gamma, [\Phi, (\phi \wedge \psi)]}{\Gamma, [\Phi, \phi], [\Phi, \psi]}\wedge \qquad \frac{\Gamma, [\Phi, \neg(\phi \wedge \psi)]}{\Gamma, [\Phi, \neg\phi, \neg\psi]}\neg\wedge$$

2. *Structural rules taut (elimination of tautologies) and abs (absorbtion of a clause by a second clause, that implies the first one)*

$$\frac{\Gamma, [\Phi, \phi, \neg\phi]}{\Gamma}taut \qquad \frac{\Gamma, [\Phi], [\Phi, \Psi]}{\Gamma, [\Phi]}abs$$

3. *Resolution res*

$$\frac{\Gamma, [\Phi, \phi], [\Psi, \neg\phi]}{\Gamma, [\Phi, \phi], [\Psi, \neg\phi], [\Phi, \Psi]}res$$

4. *Tree rules: for all $k_1, k_2, k_3, k_4 \in K$:*

$$\frac{\Gamma}{\Gamma, [\neg k_1 \lhd^* k_2, \neg k_2 \lhd^* k_3, k_1 \lhd^* k_3]}tr_{\mathcal{D}} \qquad \frac{\Gamma}{\Gamma, [k_1 \not\approx k_2, k_1 \lhd^* k_2]}rf_{\mathcal{D}}$$

$$\frac{\Gamma}{\Gamma, [\neg k_1 \lhd^* k_2, \neg k_2 \lhd^* k_1, k_1 \approx k_2]}antis_{\mathcal{D}} \qquad \frac{\Gamma}{\Gamma, [\neg k_1 \lhd k_2, k_2 \not\approx k_1]}irrf_{\mathcal{P}}$$

$$\frac{\Gamma}{\Gamma, [\neg k_1 \prec k_2, \neg k_2 \prec k_3, k_1 \prec k_3]}tr_{\mathcal{L}}$$

$$\frac{\Gamma}{\Gamma, [\neg k_1 \lhd k_2, k_1 \lhd^* k_2]}T_1 \qquad \frac{\Gamma}{\Gamma, [k_1 \lhd^* k_2, k_2 \lhd^* k_1, k_1 \prec k_2, k_2 \prec k_1]}T_2$$

$$\frac{\Gamma}{\Gamma, [\neg k_1 \lhd^* k_2, \neg k_1 \prec k_2]}T_3 \qquad \frac{\Gamma}{\Gamma, [\neg k_2 \lhd^* k_1, \neg k_1 \prec k_2]}T_4$$

$$\frac{\Gamma}{\Gamma, [\neg k_1 \lhd k_2, \neg k_3 \lhd k_2, k_1 \approx k_3]}T_5 \qquad \frac{\Gamma}{\Gamma, [\neg k_1 \prec k_2, \neg k_1 \lhd^* k_3, \neg k_2 \lhd^* k_4, k_3 \prec k_4]}T_6$$

5. *Label rules: for all $k, k_1, k_2 \in K$, for all $t \in T_\epsilon$ and for all $a \in A$:*

$$\frac{\Gamma}{\Gamma, [\delta(k) \approx t_1, \cdots, \delta(k) \approx t_n, a_1(\delta(k)) \approx v_1, \cdots a_m(\delta(k)) \approx v_l]} label_1$$

$$\frac{\Gamma}{\Gamma, [\neg k_1 \lhd k_2, \delta(k_1) \not\approx t]} nont_1 \qquad \frac{\Gamma}{\Gamma, [\delta(k_1) \not\approx t, a(\delta(k_1)) \approx \bot]} term_1$$

$$\frac{\Gamma}{\Gamma, [a(\delta(k)) \approx v_1, \cdots, a(\delta(k)) \approx v_l, a(\delta(k)) \approx \bot]} attr_1$$

6. *Equality rules: for all $k_1, k_2, k_3, k_4 \in K$, for all $t, t' \in T_\epsilon$ with $t \neq t'$, for all $a \in A$ and for all $v, v' \in V \cup \{\bot\}$ with $v \neq v'$:*

$$\frac{\Gamma}{\Gamma, [k_1 \not\approx k_2, k_3 \not\approx k_4, \neg k_1 \lhd^* k_3, k_2 \lhd^* k_4]} equ_{\mathcal{D}}$$

$$\frac{\Gamma}{\Gamma, [k_1 \not\approx k_2, k_3 \not\approx k_4, \neg k_1 \lhd k_3, k_2 \lhd k_4]} equ_{\mathcal{P}}$$

$$\frac{\Gamma}{\Gamma, [k_1 \not\approx k_2, k_3 \not\approx k_4, \neg k_1 \prec k_3, k_2 \prec k_4]} equ_{\mathcal{L}}$$

$$\frac{\Gamma}{\Gamma, [k_1 \not\approx k_2, \delta(k_1) \approx \delta(k_2)]} label_2 \qquad \frac{\Gamma}{\Gamma, [\delta(k_1) \not\approx \delta(k_2), a(\delta(k_1)) \approx a(\delta(k_2))]} attr_2$$

$$\frac{\Gamma}{\Gamma, [t \not\approx t']} term_2 \qquad \frac{\Gamma}{\Gamma, [v \not\approx v']} attr_3$$

$$\frac{\Gamma}{\Gamma, B} nont_2 \qquad with$$

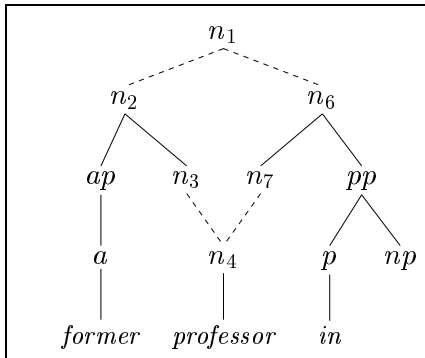
$$B := [a_1(\delta(k_1)) \not\approx a_1(\delta(k_2)), \cdots, a_m(\delta(k_1)) \not\approx a_m(\delta(k_2)),$$
$$\delta(k_1) \approx t_1, \cdots, \delta(k_1) \approx t_n, \delta(k_1) \approx \delta(k_2)]$$

7. *Equivalence rules: for all terms τ_1, τ_2, τ_3 that are either all n-terms or all l-terms or all v-terms (with respect to A, V and T), there are rules*

$$\frac{\Gamma}{\Gamma, [\tau_1 \not\approx \tau_2, \tau_2 \not\approx \tau_3, \tau_1 \approx \tau_3]} tr_= \qquad \frac{\Gamma}{\Gamma, [\tau_1 \not\approx \tau_2, \tau_2 \approx \tau_1]} sy_= \qquad \frac{\Gamma}{\Gamma, [\tau_1 \approx \tau_1]} rf_=$$

The logical rules eliminate logical connectives: they remove double negations, they transform disjunctions into clauses and conjunctions into branches and they apply the de Morgan laws to remove negations of disjunctions or conjunctions. *taut* removes tautologies and *abs* removes disjunctions if a subset of the disjunction already holds. The resolution rule captures the fact that $(\phi_1 \vee \psi) \wedge (\phi_2 \vee \neg\psi)$ implies $\phi_1 \vee \phi_2$ since either ψ or $\neg\psi$ does not hold.

The mechanism of the tree rules, label rules and equality rules is as follows: if such a rule is applied, a clause is added that holds universally for all trees. In further steps, this clause might be reduced using the resolution rule and the structural rules. The first tree axiom rule



$$\begin{aligned}\gamma_2 \quad = \quad & n_1 \lhd^* n_2 \wedge n_2 \lhd n_3 \\ & \wedge n_3 \lhd^* n_4 \wedge n_1 \lhd^* n_6 \\ & \wedge n_6 \lhd n_7 \wedge n_7 \lhd^* n_4 \wedge \cdots\end{aligned}$$

$$\begin{array}{ll}
 & || \quad [\gamma_2] \quad || \\
\overset{log.rules}{\rightarrow} & || \cdots [n_3 \lhd^* n_4], [n_7 \lhd^* n_4] \cdots || \\
\overset{rf_{\overline{=}}}{\rightarrow} & || \cdots [n_3 \lhd^* n_4], [n_7 \lhd^* n_4], [n_4 \approx n_4] \cdots || \\
\overset{rf_{\not\prec}}{\rightarrow} & || \cdots [n_3 \lhd^* n_4], [n_7 \lhd^* n_4], [n_4 \approx n_4], [n_4 \not\approx n_4, n_4 \lhd^* n_4] \cdots || \\
\overset{res}{\rightarrow} & || \cdots [n_3 \lhd^* n_4], [n_7 \lhd^* n_4], [n_4 \approx n_4], [n_4 \not\approx n_4, n_4 \lhd^* n_4], [n_4 \lhd^* n_4] \cdots || \\
\overset{abs}{\rightarrow} & || \cdots [n_3 \lhd^* n_4], [n_7 \lhd^* n_4], [n_4 \approx n_4], [n_4 \lhd^* n_4] \cdots || \\
\overset{T_3}{\rightarrow} & || \cdots [n_3 \lhd^* n_4], [n_7 \lhd^* n_4], [n_4 \approx n_4], [n_4 \lhd^* n_4], [\neg n_4 \lhd^* n_4, \neg n_4 \prec n_4] \cdots || \\
\overset{res}{\rightarrow} & || \cdots [n_3 \lhd^* n_4], [n_7 \lhd^* n_4], [n_4 \approx n_4], [n_4 \lhd^* n_4], [\neg n_4 \lhd^* n_4, \neg n_4 \prec n_4], \\
 & \quad [\neg n_4 \prec n_4] \cdots || \\
\overset{abs}{\rightarrow} & || \cdots [n_3 \lhd^* n_4], [n_7 \lhd^* n_4], [n_4 \approx n_4], [n_4 \lhd^* n_4], [\neg n_4 \prec n_4] \cdots || \\
\overset{T_6}{\rightarrow} & || \cdots [n_3 \lhd^* n_4], [n_7 \lhd^* n_4], [n_4 \approx n_4], [n_4 \lhd^* n_4], [\neg n_4 \prec n_4], \\
 & \quad [\neg n_3 \prec n_7, \neg n_3 \lhd^* n_4, \neg n_7 \lhd^* n_4, n_4 \prec n_4] \cdots || \\
\overset{res,abs}{\rightarrow} & || \cdots [n_3 \lhd^* n_4], [n_7 \lhd^* n_4], [n_4 \approx n_4], [n_4 \lhd^* n_4], [\neg n_4 \prec n_4], \\
 & \quad [\neg n_3 \prec n_7, \neg n_3 \lhd^* n_4, \neg n_7 \lhd^* n_4] \cdots || \\
\overset{res,abs}{\rightarrow} & || \cdots [n_3 \lhd^* n_4], [n_7 \lhd^* n_4], [n_4 \approx n_4], [n_4 \lhd^* n_4], [\neg n_4 \prec n_4], \\
 & [\neg n_3 \prec n_7, \neg n_3 \lhd^* n_4] \cdots || \\
\overset{res,abs}{\rightarrow} & || \cdots [n_3 \lhd^* n_4], [n_7 \lhd^* n_4], [n_4 \approx n_4], [n_4 \lhd^* n_4], [\neg n_4 \prec n_4], [\neg n_3 \prec n_7] \cdots || \\
\overset{T_6,res,abs}{\rightarrow} & || \cdots [n_3 \lhd^* n_4], [n_7 \lhd^* n_4], [n_4 \approx n_4], [n_4 \lhd^* n_4], [\neg n_4 \prec n_4], [\neg n_3 \prec n_7], \\
 & \quad [\neg n_7 \prec n_3] \cdots || \\
\overset{T_2}{\rightarrow} & || \cdots [n_3 \lhd^* n_4], [n_7 \lhd^* n_4], [n_4 \approx n_4], [n_4 \lhd^* n_4], [\neg n_4 \prec n_4], [\neg n_3 \prec n_7], \\
 & \quad [\neg n_7 \prec n_3], [n_3 \lhd^* n_7, n_7 \lhd^* n_3, n_3 \prec n_7, n_7 \prec n_3] \cdots || \\
\overset{res,abs}{\rightarrow} & || \cdots [n_3 \lhd^* n_4], [n_7 \lhd^* n_4], [n_4 \approx n_4], [n_4 \lhd^* n_4], [\neg n_4 \prec n_4], [\neg n_3 \prec n_7], \\
 & \quad [\neg n_7 \prec n_3], [n_3 \lhd^* n_7, n_7 \lhd^* n_3, n_3 \prec n_7] \cdots || \\
\overset{res,abs}{\rightarrow} & || \cdots [n_3 \lhd^* n_4], [n_7 \lhd^* n_4], [n_4 \approx n_4], [n_4 \lhd^* n_4], [\neg n_4 \prec n_4], [\neg n_3 \prec n_7], \\
 & \quad [\neg n_7 \prec n_3], [n_3 \lhd^* n_7, n_7 \lhd^* n_3] \cdots ||
\end{array}$$

FIGURE 3.2: SAMPLE APPLICATION OF INFERENCE RULES

for example expresses the fact that the dominance relation on nodes is transitive, i.e. for two node names k_1, k_2 and a formula ϕ: if $\phi \vdash k_1 \lhd^* k_2 \wedge k_2 \lhd^* k_3$, then $\phi \vdash k_1 \lhd^* k_3$ also holds. This implication is expressed by $\neg(k_1 \lhd^* k_2 \wedge k_2 \lhd^* k_3) \vee k_1 \lhd^* k_3$ which is equivalent to the disjunction $\neg(k_1 \lhd^* k_2) \vee \neg(k_2 \lhd^* k_3) \vee k_1 \lhd^* k_3$. This corresponds to the clause added in the rule $tr_{\mathcal{D}}$.

In a similar way, the rules T_1 to T_6 express the following facts: T_1 signifies that $\phi \vdash k_1 \lhd k_2$ entails $\phi \vdash k_1 \lhd^* k_2$. T_2 captures the condition that two nodes in a tree must be related either by domination or by linear precedence (exhaustiveness). With T_3 and T_4 these two relations are exclusive. T_5 expresses the uniqueness of the parent node, and T_6 captures the inheritance or non-tangling property.

The label rules add the following: with $label_1$, for each node u, the label $\mu(u)$ either is in $T \cup \{\epsilon\}$ or it is a nonempty feature structure, i.e. there is at leat one attribute a and one value v with $\langle a, v\rangle \in \mu(u)$. The rule $nont_1$ signifies that a node that is parent of some other node cannot be labelled by a terminal or the empty word. With $term_1$, for each node with a terminal label all attributes are undefined. $attr_1$ expresses that each attribute value is either in V or undefined.

Equality rules take care of the fact that $\approx$ denotes equality between nodes, labels or attribute values.

The last group of rules, the equivalence rules make sure that $\approx$ is an equivalence relation.

As an example of the application of the inference rules, Fig. 3.2 shows how to deduce from the underspecified tree description γ_2 in Fig. 3.1 the consequence that there are two possibilities for the dominance relation between the node names n_3 and n_7. For the sake of simplicity, all formulas concerning labels are omitted, and node names such as $n_1, n_2, \cdots, ap, pp, \cdots$ are used. By application of inference rules, an additional clause $[n_3 \lhd^* n_7, n_7 \lhd^* n_3]$ is derived, i.e. the tree description implies that either n_3 dominates n_7 (wide scope of *former*) or n_7 dominates n_3 (wide scope of the PP). In the sample deduction, only those clauses are listed that are used to derive $[n_3 \lhd^* n_7, n_7 \lhd^* n_3]$.

Note that, as already mentioned in Chapter 1, objects such as the kind of graph with different edges shown in Fig. 3.2 are pictures of tree descriptions. Tree descriptions themselves are formulas, and these pictures are graphical representations of (some of) the information entailed by the formula in question. The parent relation is depicted by a normal edge and dotted edges represent dominance relations. If a node name is depicted left of another node name, this does not necessarily mean that there is a linear precedence relation between these node names. In γ_2 for example, there is no linear precedence relation between n_2 and n_6, although in the picture n_2 is left of n_6.

It is easy to see that the application of inference rules does not change the set of models (i.e. of trees) satisfying a branch:

Proposition 3.1 (Model invariance) *Let $\|\Gamma_1\|$ and $\|\Gamma_2\|$ be two branches. If $\|\Gamma_2\|$ can be constructed from $\|\Gamma_1\|$ in finitely many steps by applying inference rules, then:*

$$\bigwedge_{\Phi_1 \in \|\Gamma_1\|} \bigvee_{\phi_1 \in \Phi_1} \models \bigwedge_{\Phi_2 \in \|\Gamma_2\|} \bigvee_{\phi_2 \in \Phi_2} \quad \textit{and} \quad \bigwedge_{\Phi_2 \in \|\Gamma_2\|} \bigvee_{\phi_2 \in \Phi_2} \models \bigwedge_{\Phi_1 \in \|\Gamma_1\|} \bigvee_{\phi_1 \in \Phi_1}$$

This can be shown by induction on the number of rule applications.

The following definition is again taken from Rogers 1994. The idea is that a branch is *saturated* if no new consequences can be added by applying inference rules, and a *closed* branch contains one empty clause and is therefore unsatisfiable.

Definition 3.7 (Saturated, open, closed and satisfiable branches) *Let* $\|\Gamma\|$ *be a branch with attributes* A*, attribute values* V *and terminals* T*.*

1. $\|\Gamma\|$ *is saturated, iff:*
   - *No logical rule can be applied to* $\|\Gamma\|$*.*
   - *Neither abs nor taut can be applied to* $\|\Gamma\|$*.*
   - *For all applications of res to* $\|\Gamma\|$ *the new clause introduced by this application can be removed by abs or taut.*
   - *For all tree rules, label rules, equality rules or equivalence rules with alphabets* A*,* V*,* T *and with node names occuring in* $\|\Gamma\|$*: if such a rule is applied to* $\|\Gamma\|$*, then the resulting new clauses can be removed by abs.*
2. $\|\Gamma\|$ *is closed, iff* $\|\Gamma\|$ *contains an empty clause.*
3. $\|\Gamma\|$ *is open, iff it is not closed.*
4. $\|\Gamma\|$ *is satisfiable, iff there is a tree* B *and a node assignment function* g *such that* $B \models_g \|\Gamma\|$*.*

The next proposition says that each formula ψ can be extended to a saturated branch by applying inference rules. (For the syntactic consequence for formulas ψ_1, ψ_2, defined below, the saturated branches of ψ_1 and ψ_2 will be compared.)

Proposition 3.2 *Let* ψ *be an* (A, V, T)*-formula for some attributes* A*, values* V *and terminals* T*, and let* $K_{fin} \subseteq K$ *be a finite set such that node*$(\psi) \subseteq K_{fin}$*. In finitely many steps a unique saturated branch* $\|\Gamma\|$ *can be constructed for* ψ *and* K_{fin}*, such that*

$$\psi \models \bigwedge_{\Phi \in \|\Gamma\|} \bigvee_{\phi \in \Phi} \phi \quad \text{and} \quad \bigwedge_{\Phi \in \|\Gamma\|} \bigvee_{\phi \in \Phi} \not\phi \models \psi$$

Notation: $\Gamma(\psi, K_{fin})$ *is this saturated branch.*

If it does not matter which set K_{fin} was used, I simply write "$\Gamma(\psi)$" instead of "$\Gamma(\psi, K_{fin})$".

Prop. 3.2 can be proven by first giving an algorithm for constructing $\Gamma(\psi, K_{fin})$ for a given ψ and K_{fin} and then showing that the result has the properties mentioned in the proposition. The construction is very similar to the one proposed in Prop. 5 in Rogers 1994, and it consists of an application of the inference rules in a certain order.

Proof: Construction: $\Gamma_1 := \|[\psi]\|$

1. Apply each tree rule, label rule, equality rule and equivalence rule with alphabets A, V and T and node names from K_{fin} once to Γ_1. The result is called Γ_2.
2. Apply the logical rules as often as possible to Γ_2. The result is called Γ_3.
3. Apply *res* repeatedly to Γ_3, as long as new clauses can be added by *res*. The result is called Γ_4.

4. Apply the structural rules as often as possible to Γ_4. The result is $\|\Gamma\|$.

(The order of the first two steps is of no importance because none of the rules applied in the first step adds any logical connectives.)
To show: The construction terminates after finitely many steps and the result is a saturated branch $\|\Gamma\|$, such that $B \models_g \psi$ iff $B \models_g \|\Gamma\|$.

(i) Construction terminates after finitely many steps, i.e. each of the four parts 1. to 4. only consists of a finite number of rule applications:

(a) 1. part: There are only finitely many rules with node names from K_{fin} and symbols from A, V and T that are applied to Γ_1.

(b) 2. part: There is only a finite number of applications of logical rules to Γ_2 because Γ_2 is finite and each logical rule removes some logical connective.

(c) 3. part: Each application of *res* in 3. adds a new clause. But the number of possible clauses containing elementary formulas with node names from K_{fin} and symbols from A, V and T is finite:

There are only finitely many elementary formulas, that can occur as elements of a clause. If n is the number of possible elementary formulas, then there are 2^n possible clauses and consequently at most 2^n applications of *res* in 3.

(d) 4. part: Each structural rule removes a clause. Consequently only finitely many applications of *taut* and *abs* are possible.

(ii) $\|\Gamma\|$ is saturated:

(a) No logical rule can be applied to $\|\Gamma\|$ because no logical connectives are added in 2., 3. or 4.

(b) Because of the 4. part no structural rules can be applied to $\|\Gamma\|$.

(c) Because of $\|\Gamma\| \subseteq \Gamma_4$ all applications of *res*, that are possible in $\|\Gamma\|$ have already been done in 3. Each additional application of *res* either does not change anything or adds a clause that can be removed by *taut* or *abs*.

(d) In the beginning all tree rules, label rules, equality rules and equivalence rules with names from K_{fin} have been applied. The clauses added by these rules are no tautologies and contain only elementary formulas. Because of this they have not been removed by *taut* or by a logical rule. Perhaps they have been reduced by *abs*. Consequently for each result $\|\Gamma, \Phi_1, \Phi_2 \cdots\|$ of an application of a tree axiom rule or a node symbol rule to $\|\Gamma\|$: For each new clause Φ_i there is a clause Ψ_i in $\|\Gamma\|$ with $\Psi_i \subseteq \Phi_i$, i.e. Φ_i can be eliminated by *abs*.

(iii) Model invariance: Holds because of $\psi = \bigwedge_{\Phi \in \|[\psi]\|} \bigvee_{\phi \in \Phi}$ and Prop. 3.1.

□

Closed branches are not satisfiable. Therefore (with Prop. 3.1) for all ϕ: if $\Gamma(\phi)$ is closed, then ϕ is not satisfiable. This means that a necessary condition for the satisfiability of a formula ϕ is that $\Gamma(\phi)$ must be open. We will see that this is even sufficient, i.e. if $\Gamma(\phi)$ is open, then ϕ is satisfiable.

By comparing the saturated branches $\Gamma(\phi_1)$ and $\Gamma(\phi_2)$ of two formulas ϕ_1 and ϕ_2 a syntactic notion of consequence $\phi_1 \vdash \phi_2$ can be defined. The idea is as follows: for two

formulas ϕ_1, ϕ_2, the saturated branches $\Gamma(\phi_1)$ and $\Gamma(\phi_2)$ (for the same sets of node names) represent conjunctive normal forms of the consequences of ϕ_1 and ϕ_2 respectively. If ϕ_1 implies ϕ_2, this means that for each clause (i.e. each conjunct) Ψ_2 in $\Gamma(\phi_2)$, there must be a clause (conjunct) Ψ_1 in $\Gamma(\phi_1)$ such that Ψ_1 implies Ψ_2. Since Ψ_2 and Ψ_1 are clauses, i.e. disjunctions, each disjunct in Ψ_1 must then occur in Ψ_2, in other words Ψ_1 must be a subset of Ψ_2.

Definition 3.8 (Syntactic consequence $\phi_1 \vdash \phi_2$) *Let ϕ_1 and ϕ_2 be formulas, $\Gamma_1 := \Gamma(\phi_1, K_{fin})$ and $\Gamma_2 := \Gamma(\phi_2, K_{fin})$ with $K_{fin} := node(\phi_1) \cup node(\phi_2)$.*

$$\phi_1 \vdash \phi_2 \textit{ iff for all } \Psi_2 \in \Gamma_2 \textit{ there is a } \Psi_1 \in \Gamma_1 \textit{ such that } \Psi_1 \subseteq \Psi_2$$

$\vdash$ is transitive and because of Prop. 3.2 decidable.

3.1.4 Soundness and Completeness

If $\phi_1 \vdash \phi_2$, then for all trees B and all node assignment functions g: If $B \models_g \phi_1$, then $B \models_g \phi_2$. This follows immediately from Def. 3.8 together with Prop. 3.2 and Def. 3.4. Consequently the following holds:

Proposition 3.3 (Soundness) *For all ϕ_1, ϕ_2: If $\phi_1 \vdash \phi_2$, then $\phi_1 \models \phi_2$.*

In order to prove completeness, the satisfiability of open saturated branches first must be proven. This proposition will then be used in the completeness proof.

Proposition 3.4 (Satisfiability of open saturated branches) *For all formulas ψ and all $\|\Gamma\| = \Gamma(\psi)$:*

$$\|\Gamma\| \textit{ is open iff } \|\Gamma\| \textit{ is satisfiable (iff } \psi \textit{ is satisfiable)}$$

With this proposition the openness of a saturated branch is not only a necessary but also a sufficient condition for satisfiablility.

The proof of the satisfiability of an open saturated branch is as follows: first, $\Gamma(\psi, K_{fin})$ is extended to a branch Γ' such that in Γ' linear precedence, the parent relation, dominance, equality and the node labels are completely specified for the node names occuring in $\Gamma(\psi)$. The saturatedness of Γ' is shown, and then, similar to the construction in Rogers 1994, p. 29, a tree satisfying Γ' and also $\Gamma(\psi)$ is constructed. This tree is constructed by taking as nodes the equivalence classes of the relation $\approx_n \subseteq K_{fin} \times K_{fin}$ with $k_1 \approx_n k_2$ iff $[k_1 \approx k_2] \in \Gamma'$.

Proof: Let ψ be a (A, V, T)-formula, and let K_{fin} be a finite set with $node(\psi) \subseteq K_{fin} \subseteq K$. Since closed branches are not satisfiable, only the following must be shown: if $\Gamma := \Gamma(\psi, K_{fin})$ is open, then ψ is satisfiable.
Let Γ be open.

(i) First Γ is extended to a branch Γ', such that all node labels are completely described in Γ' and for all $k_1, k_2 \in K_{fin}$:
$[k_1 \prec k_2] \in \Gamma'$ or $[\neg k_1 \prec k_2] \in \Gamma'$, $[k_1 \lhd k_2] \in \Gamma'$ or $[\neg k_1 \lhd k_2] \in \Gamma'$, $[k_1 \lhd^* k_2] \in \Gamma'$ or $[\neg k_1 \lhd^* k_2] \in \Gamma'$ and $[k_1 \approx k_2] \in \Gamma'$ or $[\neg k_1 \approx k_2] \in \Gamma'$ (in general Γ' is not unique).
Construction:

(a) Definition of Γ':

$$\begin{aligned}\Gamma_0 := \Gamma \quad &\cup \quad \{[k_1 \mathrm{R} k_2, \neg k_1 \mathrm{R} k_2] \mid \mathrm{R} \in \{\prec, \lhd, \lhd^*, \approx\}, k_1, k_2 \in K_{fin}\} \\ &\cup \quad \{[\delta(k) \approx t, \neg\delta(k) \approx t] \mid t \in T \cup \{\epsilon\}, k \in K_{fin}\} \\ &\cup \quad \{[a(\delta(k)) \approx v, \neg a(\delta(k)) \approx v] \mid a \in A, v \in V \cup \{\bot\}, k \in K_{fin}\}\end{aligned}$$

$\Gamma' := \Gamma_0$

(b) Apply first *res* (as long as no tautology is introduced) and then *abs* as often as possible to Γ' (only finitely many applications).

(c) If $|[\Phi]| = 1$ for all $[\Phi] \in \Gamma'$, then the construction is terminated.

Else: Choose one $[\Phi] \in \Gamma'$, such that $|[\Phi]| > 1$, and $[\Phi]$ was added in (a), and then choose one $\phi \in [\Phi]$. $\Gamma'' := \Gamma' \cup \{[\phi]\}$.

Apply *res* as often as possible to Γ'', as long as *res* does not introduce any tautologies (these applications of *res* add only subsets of clauses that are already in Γ'').

Apply then *abs* as often as possible to Γ'' (this only removes clauses that are supersets of $[\phi]$ or of the clauses just added by *res*). Let Γ''' be the resulting branch. $\Gamma' := \Gamma'''$.

Continue with (c).

The construction terminates after finitely many steps because no new clauses are introduced and in each step at least one $[\Phi]$ is reduced.

No clause is completely removed. Consequently for each tautology added to Γ in the beginning, a subset containing one formula is in Γ'.

(ii) To show: Γ' is saturated.

- No logical rule can be applied to Γ.
- *res* and *abs* have been applied as often as possible.
- *taut* cannot be applied to Γ' because $|[\Phi]| = 1$ holds for all $[\Phi] \in \Gamma'$.
- $\Gamma \subseteq \Gamma_0$ and for all $[\Psi] \in \Gamma_0$ there is a $[\Phi] \in \Gamma'$, such that $[\Phi] \subseteq [\Psi]$.

  Consequently, for all $[\Psi] \in \Gamma$ there is a $[\Phi] \in \Gamma''$, such that $[\Phi] \subseteq [\Psi]$. This means that all clauses, that can be added to Γ' by tree, label or equality rules, can be removed by *abs*, and that for all trees B and all node assignments g: if $B \models_g \Gamma'$, then $B \models_g \Gamma$.

(iii) To show: Γ' is satisfiable:

Construction of a tree $B = (\mathcal{U}, u_0, \mathcal{P}, \mathcal{D}, \mathcal{L}, \mu)$ satisfying Γ':

- Nodes and relations $\mathcal{P}, \mathcal{D}$ and $\mathcal{L}$:

  $\mathcal{U}_{\Gamma'} := \{u \subseteq K_{fin} \mid$ for all $k_1, k_2 \in u : [k_1 \approx k_2] \in \Gamma'\}$.

  Let u_0 be a new node, $u_0 \notin \mathcal{U}_{\Gamma'}$. u_0 is the root of B.

$$\begin{aligned}\mathcal{D} \quad := \quad &\{\langle u_1, u_2\rangle \mid u_1, u_2 \in \mathcal{U}_{\Gamma'} \text{ and there are } k_1 \in u_1 \text{ and } k_2 \in u_2 \\ &\qquad \text{with } [k_1 \lhd^* k_2] \in \Gamma'\} \\ \cup \quad &\{\langle u_0, u\rangle \mid u \in \mathcal{U}_{\Gamma'} \cup \{u_0\}\} \\ \mathcal{P} \quad := \quad &\{\langle u_1, u_2\rangle \mid u_1, u_2 \in \mathcal{U}_{\Gamma'} \text{ and there are } k_1 \in u_1 \text{ and } k_2 \in u_2 \\ &\qquad \text{with } [k_1 \lhd k_2] \in \Gamma'\} \\ \cup \quad &\{\langle u_0, u\rangle \mid u \in \mathcal{U}_{\Gamma'} \text{ and there is no } u' \in \mathcal{U}_{\Gamma'}, u' \neq u \text{ with } \langle u', u\rangle \in \mathcal{D}\}\end{aligned}$$

$$\mathcal{L} := \{\langle u_1, u_2\rangle \mid u_1, u_2 \in \mathcal{U}_{\Gamma'} \text{ and there are } k_1 \in u_1 \text{ and } k_2 \in u_2 \text{ such that } [k_1 \prec k_2] \in \Gamma'\}$$

$\mathcal{U}_{new} := \emptyset$. For all $u \in \mathcal{U}_{\Gamma'}$: if there is no u' with $\langle u', u\rangle \in \mathcal{P}$, and if u_d is such that $\langle u_d, u\rangle \in \mathcal{D}$ and there is no u', $u \neq u'$ and $u' \neq u_d$ with $\langle u_d, u'\rangle, \langle u', u\rangle \in \mathcal{D}$, then

- choose a u'_d with $u'_d \notin \mathcal{U}_{\Gamma'} \cup \mathcal{U}_{new} \cup \{u_0\}$, and add u'_d to $\mathcal{U}_{new}$,
- add $\langle u_d, u'_d\rangle$ and $\langle u'_d, u\rangle$ to $\mathcal{P}$
- add $\langle u'_d, u'_d\rangle$ to $\mathcal{D}$, and then add $\langle u_1, u'_d\rangle$ to $\mathcal{D}$ for all u_1 with $\langle u_1, u\rangle \in \mathcal{D}$, and then add $\langle u'_d, u_2\rangle$ to $\mathcal{D}$ for all u_2 with $\langle u_d, u_2\rangle \in \mathcal{D}$.
- add $\langle u_1, u'_d\rangle$ to $\mathcal{L}$ for all u_1 with $\langle u_1, u\rangle \in \mathcal{L}$, and then add $\langle u'_d, u_2\rangle$ to $\mathcal{L}$ for all u_2 with $\langle u, u_2\rangle \in \mathcal{L}$.

$\mathcal{U} := \mathcal{U}_{\Gamma'} \cup \{u_0\} \cup \mathcal{U}_{new}$.

- Labels: Let T be the terminals.
  Choose a new attribute $a_{new} \notin A$ and a new value $v_{new} \notin V$.
  $N := \{X \mid X \subseteq (A \cup \{a_{new}\}) \times (V \cup \{v_{new}\})$ with $\langle a_{new}, v_{new}\rangle \in X\}$.
  Without loss of generality suppose that T, N and $\mathcal{U}$ are pairwise disjoint.
  - For all $t \in T \cup \{\epsilon\}$ and $u \in \mathcal{U}_{\Gamma'}$:
    $\mu(u) := t$ iff $[\delta(k) \approx t] \in \Gamma'$ for all $k \in u$.
  - For all $u \in \mathcal{U}_{new} \cup \{u_0\}$ define $\mu(u) := \{\langle a_{new}, v_{new}\rangle\}$.
  - For all $u \in \mathcal{U}_{\Gamma'}$ such that there is no $t \in T \cup \{\epsilon\}$ with $[\delta(k) \approx t] \in \Gamma'$ for some $k \in u$:
    $\mu(u) := \{\langle a, v\rangle \mid a \in A, v \in V$ and $[a(\delta(k)) \approx v] \in \Gamma'$ for all $k \in u\} \cup \{\langle a', v'\rangle\}$.

Clearly, there is at least one B that can be constructed in such a way.

To show: B is a finite labelled tree.
Since K_{fin} is finite, $\mathcal{U}$ is also finite.
The axioms listed under *3.* in Def. 2.8 on page 16 in Chapter 2 must be shown:

(a) $\mathcal{P}$ is irreflexive because of the inference rule $irrf_{\mathcal{P}}$.
Because of the additional pairs added to $\mathcal{P}$ with nodes in $\mathcal{U}_{new}$, for each $u \in \mathcal{U} \setminus \{u_0\}$, there is a u' with $\langle u', u\rangle \in \mathcal{P}$.
Furthermore, because of T_5 for each $u \in \mathcal{U}$, there is at most one u' with $\langle u', u\rangle \in \mathcal{P}$.

(b) $\mathcal{D}$ is reflexive, transitive and antisymmetric because of the rules $rf_{\mathcal{D}}$, $tr_{\mathcal{D}}$ and $antis_{\mathcal{D}}$.
With T_1 and because of the construction of $\mathcal{D}$, $\mathcal{P} \subset \mathcal{D}$ holds.
Suppose that $\langle u_1, u_2\rangle \in \mathcal{D}$ holds for some $u_1, u_2 \in \mathcal{U}$ with $u_1 \neq u_2$ and $\langle u_1, u_2\rangle \notin \mathcal{P}$.
Then (because of the relations added with nodes in $\mathcal{U}_{new}$) there is a u_3 with $\langle u_1, u_3\rangle \in \mathcal{D}$ and $\langle u_3, u_2\rangle \in \mathcal{P}$.
Consequently, $\mathcal{D}$ is the reflexive transitive closure of $\mathcal{P}$.

(c) $\mathcal{L}$ is transitive because of $tr_{\mathcal{L}}$ and the new pairs added to $\mathcal{L}$ with nodes in $\mathcal{U}_{new}$.

(d) Because of rule T_2 and the new pairs added to $\mathcal{D}$ and $\mathcal{P}$ with nodes from $\mathcal{U} \setminus (\mathcal{U}_{new} \cup \{u_0\})$, for all $u_1, u_2 \in \mathcal{U}$: $\langle u_1, u_2\rangle \in \mathcal{D}$ or $\langle u_2, u_1\rangle \in \mathcal{D}$ or $\langle u_1, u_2\rangle \in \mathcal{L}$ or $\langle u_2, u_1\rangle \in \mathcal{L}$.

(e) For all $u_1, u_2 \in \mathcal{U}$ with $\langle u_1, u_2\rangle \in \mathcal{L}$: with T_3 and T_4 and the construction of $\mathcal{L}$: $\langle u_1, u_2\rangle \notin \mathcal{D}$ and $\langle u_2, u_1\rangle \notin \mathcal{D}$.

(f) Axiom (f) also holds because of rule T_6 and the new pairs with nodes from $\mathcal{U}_{new}$ added to $\mathcal{D}$ and $\mathcal{L}$.

Because of the rules $label_2$, $attr_2$, $term_2$ and $attr_3$, μ is a function, i.e. μ is unique and the values of attributes are also unique.
With the form of Γ' and the definition of μ, $\mu(u)$ is defined for all $u \in \mathcal{U}$,
and because of rule $nont_1$, for each node u that is no leaf, $\mu(u) \in N$ holds.

Clearly, B is constructed in such a way that it satisfies Γ' with each assignment function g such that for all $u \in \mathcal{U}_{\Gamma'}$: $g(k) = u$ for all $k \in u$.

□

With this proposition completeness can be shown:

Proposition 3.5 (Completeness) *For all ϕ_1, ϕ_2: If $\phi_1 \models \phi_2$, then $\phi_1 \vdash \phi_2$.*

This proposition can be shown by induction on the structure of ϕ_2 where ϕ_2 is supposed to be in conjunctive normal form, i.e. to be a conjunction of disjunctions of elementary formulas.

In order to render the completeness proof more readable, a special case is first considered separately in the following lemma:

Lemma 3.1 *For all formulas $\phi = \psi_1 \vee \cdots \psi_{n-1}$ and ψ_n, such that ψ_i is an elementary formula for $1 \leq i \leq n$: $\phi \vdash \phi \vee \psi_n$.*

Proof of Lemma 3.1: Let ϕ and ψ_n be as in the lemma, $\Gamma := \Gamma(\phi, node(\phi \vee \psi_n))$ and $\Gamma' := \Gamma(\phi \vee \psi_n, node(\phi \vee \psi_n))$.
If ϕ is a tautology, then $\Gamma = \Gamma'$ and therefore $\phi \vdash \phi \vee \psi_n$ holds.
Let ϕ be no tautology.
Suppose that, when constructing Γ and Γ', the logical rules have been applied first.
The result of the applications of all logical rules to $\|[\phi]\|$ and $\|[\phi \vee \psi_n]\|$ respectively are $\|[\psi_1, \cdots, \psi_{n-1}]\|$ and $\|[\psi_1, \cdots, \psi_{n-1}, \psi_n]\|$.
in the course of the construction of Γ', let Γ_n be the result of applying n rules to $\|[\psi_1, \cdots, \psi_{n-1}, \psi_n]\|$.
Claim: For all $\Psi_n \in \Gamma_n$, that cannot be removed by *taut*, there is a $\Phi \in \Gamma$ such that $\Phi \subseteq \Psi_n$.
Induction on n:

(i) Induction start $n = 0$: since Γ is constructed from $\|[\psi_1, \cdots, \psi_{n-1}]\|$ and *taut* cannot be applied to any subset of this branch, Γ still contains a subset of $[\psi_1, \cdots, \psi_{n-1}]$ (and also of $[\psi_1, \cdots, \psi_{n-1}, \psi_n]$).

(ii) Induction step $n \Rightarrow n+1$: possible rules that can be applied to Γ_n to construct Γ_{n+1}:

1. Tree, label, equality or equivalence rules: since Γ is saturated, the claim still holds for the result of applying such a rule.
2. *res*: let $[\Phi, \phi], [\Psi, \neg\phi] \in \Gamma_n$, *res* adds the clause $[\Phi, \Psi]$.
   There are $[\Phi'] \subseteq [\Phi, \phi]$ and $[\Psi'] \subseteq [\Psi, \neg\phi]$, such that $[\Phi'], [\Psi'] \in \Gamma$ (induction claim).
   - 1. case: $[\Phi'] \subseteq [\Phi]$ or $[\Psi'] \subseteq [\Psi] \Rightarrow [\Phi'] \subseteq [\Phi, \Psi]$ or $[\Psi'] \subseteq [\Phi, \Psi]$.

- 2. case: $[\Phi'] = [\Phi'', \phi]$ and $[\Psi'] = [\Psi'', \neg\phi]$
  Consequently (Γ saturated) there is either a subset of $[\Phi'', \Psi'']$ (and of $[\Phi, \Psi]$) in Γ, or $[\Phi'', \Psi'']$ (and consequently also $[\Phi, \Psi]$) is a tautology and can be removed by *taut*.

3. *taut* and *abs* only remove clauses, i.e. claim still holds for the result of applying *taut* or *abs*.

□

Proof of Prop. 3.5: Let ϕ_1, ϕ_2 be formulas with $K_{fin} = node(\phi_1) \cup node(\phi_2)$ such that $\phi_1 \models \phi_2$. Without loss of generality let ϕ_2 be in conjunctive normal form. Define $\Gamma_1 := \Gamma(\phi_1, K_{fin})$ and $\Gamma_2 := \Gamma(\phi_2, K_{fin})$.

If ϕ_1 is not satisfiable, then $\Gamma_1 = \|[\ ]\|$ and therefore $\phi_1 \vdash \phi_2$ holds.

Let ϕ_1 be satisfiable. Then (because of $\phi_1 \models \phi_2$) ϕ_2 is also satisfiable.
To show: $\phi_1 \vdash \phi_2$.
Induction on the structure of ϕ_2:

(i) Induction start: Let ϕ_2 be an elementary formula. Then $[\phi_2] \in \Gamma_2$.
Define $\phi_2^{neg} := \neg\phi_2$ if ϕ_2 does not contain negations, $\phi_2^{neg} := \phi_2'$, if $\phi_2 = \neg\phi_2'$.
$[\phi_2] \in \Gamma_1$ holds because:

Suppose that $[\phi_2] \notin \Gamma_1$.
Then, starting from $\Gamma_1 \cup \{[\phi_2^{neg}]\}$ no closed branch can be constructed by applying *res*.
Therefore (with Prop. 3.4) $\phi_1 \wedge \neg\phi_2$ is satisfiable.
Contradiction to $\phi_1 \models \phi_2$.

Γ_1 and Γ_2 are saturated, and (because of $[\phi_2] \in \Gamma_1$), all rule applications that were done to construct Γ_2, were done as well (with the same instantiation f of the V_Γ, V_Φ and V_ϕ) to construct Γ_1.
Consequently $\phi_1 \vdash \phi_2$.

(ii) Induction step for disjunctions: $\phi_2 = \psi_1 \vee \psi_2 \cdots \vee \psi_n$ with elementary formulas ψ_i, $1 \leq i \leq n$.
Because of $\phi_1 \models \phi_2$ either $\phi_1 \models \psi_1 \vee \psi_2 \cdots \vee \psi_{n-1}$ or $\phi_1 \models \psi_n$ hold.
Consequently (induction claim) $\phi_1 \vdash \psi_1 \vee \psi_2 \cdots \vee \psi_{n-1}$ or $\phi_1 \vdash \psi_n$.
$\psi_1 \vee \psi_2 \cdots \vee \psi_{n-1} \vdash \psi_1 \vee \psi_2 \cdots \vee \psi_{n-1} \vee \psi_n$ (Lemma 3.1) and $\psi_n \vdash \psi_1 \vee \psi_2 \cdots \vee \psi_{n-1} \vee \psi_n$ (transitivity of $\vdash$).
Consequently (transitivity of $\vdash$) $\phi_1 \vdash \psi_1 \vee \psi_2 \cdots \vee \psi_{n-1} \vee \psi_n$.

(iii) Induction step for conjunctions: $\phi_2 = \psi_1 \wedge \psi_2 \cdots \wedge \psi_n$ and $\phi_1 \models \phi_2$.
$\Rightarrow \phi_1 \models \psi_1 \wedge \cdots \psi_{n-1}$ and $\phi_1 \models \psi_n$.
Consequently (induction claim) $\phi_1 \vdash \psi_1 \wedge \cdots \psi_{n-1}$ and $\phi_1 \vdash \psi_n$.
$\Gamma_{n-1} := \Gamma(\psi_1 \wedge \cdots \psi_{n-1}, K_{fin}), \Gamma' := \Gamma(\psi_n, K_{fin})$.
For all $\Psi \in \Gamma_{n-1} \cup \Gamma'$ there is a $\Phi \in \Gamma_1$, such that $\Phi \subseteq \Psi$ (induction claim).
$\Gamma_2 = \Gamma(\phi_2, K_{fin})$ can be constructed from $\Gamma_{n-1} \cup \Gamma'$ by repeatedly applying *res*, *taut*

and *abs*. Let $\Gamma_{n,m}$ be the branch obtained after m applications of *res*, *taut* and *abs* to $\Gamma_{n-1} \cup \Gamma'$.
To show: For all $\Psi \in \Gamma_{n,m}$ there is a $\Phi \in \Gamma_1$, such that $\Phi \subseteq \Psi$.
Induction on m:

- Induction start $m = 0$: Claim holds because of $\phi_1 \vdash \psi_1 \wedge \cdots \psi_{n-1}$ and $\phi_1 \vdash \psi_n$.
- Induction step $m \rightarrow m + 1$: Possible rule applications:
  res: Let *res* add a clause to $\Gamma_{n,m}$, that is no tautology.
  Consequently there are $[\Psi_1, \psi], [\Psi_2, \neg\psi] \in \Gamma_{n,m}$, such that $[\Psi_1, \Psi_2] \in \Gamma_{n,m+1}$.
  There are subsets Ξ_1 of $[\Psi_1, \psi]$ and Ξ_2 of $[\Psi_2, \neg\Psi]$ in Γ_1.
  1. case: $\psi \notin \Xi_1$ or $\neg\psi \notin \Xi_2 \Rightarrow \Xi_1 \subseteq [\Psi_1, \Psi_2]$ or $\Xi_2 \subseteq [\Psi_1, \Psi_2]$.
  3. case: $\Xi_1 = [\Psi'_1, \psi], \Xi_2 = [\Psi'_2, \neg\psi] \Rightarrow$ there is a subset of $[\Psi'_1, \Psi'_2]$ in Γ_1 and $[\Psi'_1, \Psi'_2] \subseteq [\Psi_1, \Psi_2]$ holds.
  Structural rules: *taut* and *abs* only remove clauses.

Consequently $\phi_1 \vdash \psi_1 \wedge \cdots \psi_{n-1} \wedge \psi_n$ also holds.

□

3.2 TDG-Formalism

In this section, the tree logic presented above is used to propose a grammar formalism based on quasi-tree like formulas called *descriptions*. In the first part, the notions of description and *Tree Description Grammar (TDG)* are defined. Then derivations in TDGs are introduced, and afterwards the notions of *description language* (the set of descriptions generated by a grammar), *tree language* and *string language* are defined.

3.2.1 Tree Description Grammar

First, the form of the formulas allowed in a TDG must be defined. It is not possible to take the quasi-tree definition given in Rogers and Vijay-Shanker 1994 and Rogers 1994 because this definition does not allow a relaxation of the dominance relation: for a formula ϕ that is a quasi-tree, the following holds: for all $k_1, k_2 \in node(\phi)$, either $\phi \vdash k_1 \lhd^* k_2$ or $\phi \vdash k_2 \lhd^* k_1$ or $\phi \vdash \neg k_1 \lhd^* k_2 \wedge \neg k_2 \lhd^* k_1$. In other words, dominance must be fully specified. Therefore some of the tree descriptions that should be generated by TDGs are no quasi-trees, e.g. the tree description γ_2 in Fig. 3.1 (p. 54) is no quasi-tree.

A common idea of the TAG variants mentioned in Chapter 2 is that each elementary object in the grammar represents several trees (sets of trees in MC-TAGs, tree descriptions in DTGs and vectors of trees in UVG-DLs and V-TAGs). If an elementary object is added in the course of the derivation, then all trees of this elementary object are added. I want to keep this idea for Tree Description Grammars and therefore, the form of the formulas used in Tree Description Grammars is restricted. The form of the tree descriptions allowed in TDGs is sketched in Fig. 3.3. In these formulas certain subtrees are completely described. Together with the descriptions of these subtrees, dominance relations between leaves and roots of the subtrees are specified. There must be (at least) one node name dominating all other node

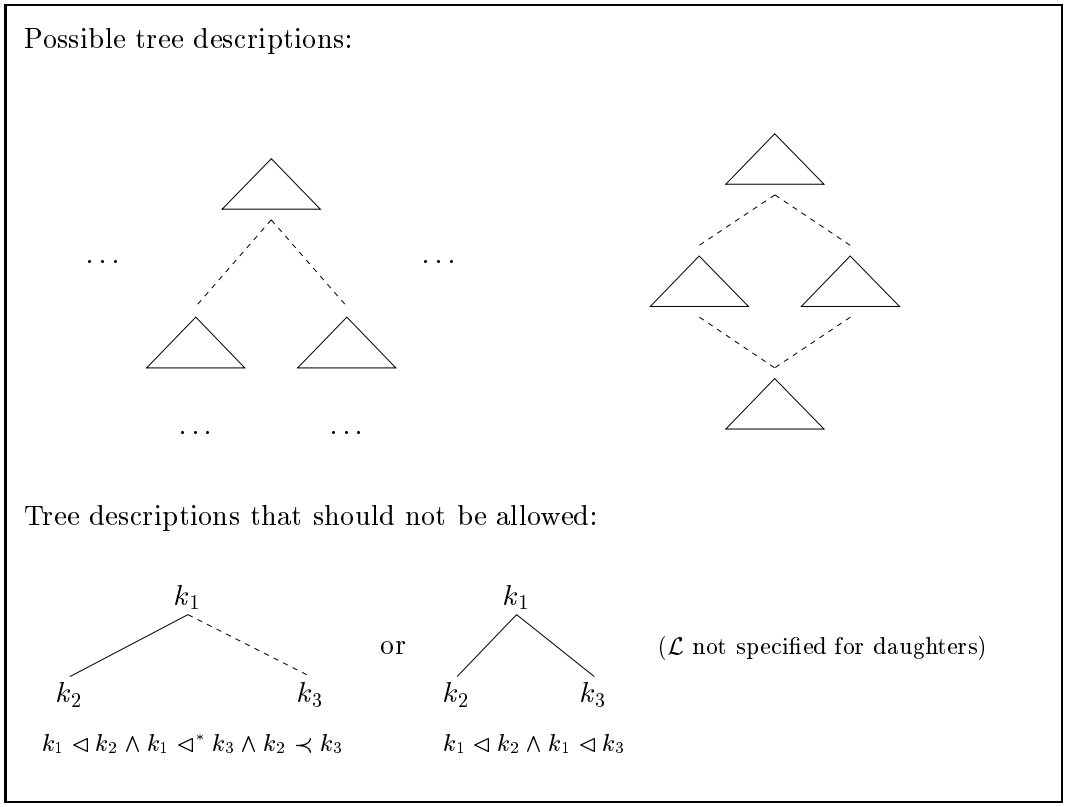


FIGURE 3.3: TREE DESCRIPTIONS IN TDGS

names. Neither negations nor disjunctions are allowed for these formulas, and the formulas must be satisfiable. This is formalized by the definition of *description*.

Before defining descriptions, a few notions are introduced, that will be frequently used afterwards: a *minimal name* in a formula is a name dominating all other names. If a tree satisfies the formula, then the minimal name denotes the root of that subtree in the tree that is covered by the description. There can be more than one minimal name in a formula, but all minimal names must be equivalent. Those dominances that are represented by dotted edges in the graphical representations are called *strong dominances*. These dominances are not implied by other dominance, parent or equivalence relations. A *leaf name* is a name dominating only itself and all equivalent names. A name k is *between* two other names, if one of the two names dominates k and k dominates the other name.

Definition 3.9 (Minimal and leaf name, strong dominance, between names) *Let ϕ be a formula and $k, k_1, k_2 \in node(\phi)$.*

1. *k is called a minimal name in ϕ iff $\phi \vdash k \lhd^* k'$ for all $k' \in node(\phi)$.*
2. *k is a leaf name in ϕ, iff for all $k' \in node(\phi)$: if $\phi \vdash k \lhd^* k'$ then $\phi \vdash k \approx k'$.*
3. *$k_1 \lhd^* k_2$ is called a strong dominance in ϕ, if*
   - *$\phi \vdash k_1 \lhd^* k_2$,*

- $\phi \not\vdash k_1 \approx k_2$ *and* $\phi \not\vdash k_1 \lhd k_2$,
- *and for all* $k_3 \in K$*: if* $\phi \vdash k_1 \lhd^* k_3 \wedge k_3 \lhd^* k_2$*, then* $\phi \vdash k_1 \approx k_3 \vee k_3 \approx k_2$.

Notation: $\phi \vdash_s k_1 \lhd^* k_2$

4. *k is between k_1 and k_2 (and between k_2 and k_1), if $k \neq k_1$, $k \neq k_2$ and either $\phi \vdash k_1 \lhd^* k \wedge k \lhd^* k_2$ or $\phi \vdash k_2 \lhd^* k \wedge k \lhd^* k_1$.*

Definition 3.10 (Description) *An (A, V, T)-formula ϕ is called an (A, V, T)-description, iff:*

1. *ϕ is satisfiable.*
2. *There is at least one minimal name in ϕ.*
3. *ϕ does not contain any negations or any disjunctions.*
4. *For all $k_1, k_2, k_3 \in K$: if $\phi \vdash k_1 \lhd k_2 \wedge k_1 \lhd^* k_3$, then either $\phi \vdash k_1 \approx k_3$ holds or there is a k_4, such that $\phi \vdash k_1 \lhd k_4 \wedge k_4 \lhd^* k_3$.*
5. *For all $k_1, k_2, k_3 \in K$: if $\phi \vdash k_1 \lhd k_2 \wedge k_1 \lhd k_3$, then either $\phi \vdash k_2 \approx k_3$ or $\phi \vdash k_2 \prec k_3$ or $\phi \vdash k_3 \prec k_2$ holds.*

The first condition holds iff $\Gamma(\phi)$ is open (see Prop. 3.4), i.e. this condition also can be formulated in a syntactic way. In particular, this means that the satisfiability of a formula is decidable. Since $node(\phi)$ is finite, the question whether a formula is a description or not, is then also decidable.

Conditions *4.* and *5.* make sure that for each node name k: either k is a leaf name, or k is a "mother node" in a fully specified subtree description, or k strongly dominates another subtree description.

Because of the existence of a minimal name in descriptions, it is easy to see that the following lemma holds:

Lemma 3.2 *For each description ϕ and each $k \in node(\phi)$:*

1. *k is minimal in ϕ or there is a k' such that $\phi \vdash k' \lhd k$ or $\phi \vdash_s k' \lhd^* k$.*
2. *k is leaf name in ϕ or there is a k' such that $\phi \vdash k \lhd k'$ or $\phi \vdash_s k \lhd^* k'$.*

Proof Let ϕ be a description, $k \in node(\phi)$.

1. Let k_1 be minimal in ϕ.
   Suppose that k is not minimal in ϕ (i.e. $\phi \not\vdash k_1 \approx k$), and that there is no k_p with $\phi \vdash k_p \lhd k$.
   Since k_1 is minimal, $\phi \vdash k_1 \lhd^* k$ holds. Then either $\phi \vdash_s k_1 \lhd^* k$ or there is a k_2 with $\phi \not\vdash k_2 \approx k_1$ and $\phi \not\vdash k_2 \approx k$ and $\phi \vdash k_1 \lhd^* k_2 \wedge k_2 \lhd^* k$.
   Inductively (because $node(\phi)$ is finite), it follows that there are $k_1, \cdots k_n$ such that $\phi \vdash k_1 \lhd^* k_2 \wedge \cdots \wedge k_{n-1} \lhd^* k_n \wedge k_n \lhd^* k$, for all $1 \leq i < n$, $\phi \not\vdash k_{i+1} \approx k_i$ and $\phi \not\vdash k_{i+1} \approx k$ and there is no k_{n+1} with $\phi \vdash k_n \lhd^* k_{n+1} \wedge k_{n+1} \lhd^* k$ and $\phi \not\vdash k_n \approx k_{n+1}$ and $\phi \not\vdash k_{n+1} \approx k$.
   $\Rightarrow \phi \vdash_s k_n \lhd^* k$.
2. The second claim can be shown in a very similar way.

□

In principle, a Tree Description Grammar (TDG) consists of alphabets A, V and T (attributes, attribute values and terminals), a set of (A, V, T)-descriptions and a specific start

description. In a description in a TDG certain node names may be marked. The descriptions occurring in a TDG are called elementary descriptions. Marked names are important for the derivation process: in a derivation step where an elementary description ψ is added, equivalent names must be found for the minimal names and all marked names in ψ. Roughly said, the marked names in ψ indicate the points where ψ must be connected to the old description. In this respect they resemble to foot nodes in TAGs. E.g. in the description β in Fig. 3.1, the name n_8 must be marked. Since a marked name can be viewed as a kind of open slot (as a substitution or foot node in a TAG), marked names must not have terminal labels.

Definition 3.11 (Tree Description Grammar) *A Tree Description Grammar (TDG) is a tuple $G = (A, V, T, D, \phi_S)$, such that:*

1. *A, V and T are pairwise disjoint alphabets, the attributes, attribute values and the terminal symbols.*
2. *D is a finite set of pairs $\langle \psi, K_\psi \rangle$ where ψ is an (A,V,T)-description and $K_\psi \subseteq node(\psi)$ such that for none of the $k_\psi \in K_\psi$ there is a $t \in T \cup \{\epsilon\}$ with $\psi \vdash \delta(k_\psi) \approx t$.*
   Each $\langle \psi, K_\psi \rangle \in D$ is called an elementary description.
3. *ϕ_S is an (A,V,T)-description, the start description.*

For each elementary description (ψ, K_ψ) define $M_\psi := K_\psi \cup \{k_m \,|\, k_m$ minimal in $\psi\}$. A $k \in K_\psi$ is called a marked name (in ψ).

3.2.2 Derivation in a TDG

The following equivalence relation is necessary for technical reasons. It will be used to make sure that in each TDG derivation step descriptions with disjoint sets of node names can be chosen. This is desirable because identifications of node names should be controlled by the marking of certain names and by the derivation process. I.e. they should not be caused by accidental occurrences of equal node names in different descriptions.

Definition 3.12 (K-equivalent) *Let A, V, T be alphabets.*

1. *Let $f : K \to K$ be a function. f is homomorphically extended to (A,V,T)-formulas by defining $\hat{f}(\phi)$ inductively for some (A,V,T)-formula ϕ:*
   (a) *For all l- or v-terms τ_1, τ_2, $\hat{f}(\tau_1 \approx \tau_2) = \hat{f}(\tau_1) \approx \hat{f}(\tau_2)$, where*
      - *$\hat{f}(x) := x$ for all $x \in T \cup V \cup \{\epsilon, \bot\}$ and*
      - *$\hat{f}(a(\delta(k))) := a(\delta(f(k)))$ for all $k \in K$ and $a \in A$.*
   (b) *For all $k_1, k_2 \in K$ and $\mathrm{R} \in \{\lhd, \lhd^*, \prec, \approx\}$: $\hat{f}(k_1 \,\mathrm{R}\, k_2) := f(k_1)\,\mathrm{R}\, f(k_2)$.*
   (c) *$\hat{f}(\neg\phi_1) := \neg\hat{f}(\phi_2)$, $\hat{f}(\phi_1 \wedge \phi_2) := \hat{f}(\phi_1) \wedge \hat{f}(\phi_2)$ and $\hat{f}(\phi_1 \vee \phi_2) := \hat{f}(\phi_1) \vee \hat{f}(\phi_2)$ for all formulas ϕ_1, ϕ_2.*
2. *Let ϕ, ϕ' be (A,V,T)-formulas. $\phi \approx_K \phi'$ iff there is a bijection $f : K \to K$ such that $\phi' = \hat{f}(\phi)$. ϕ and ϕ' are then called K-equivalent.*
3. *Let $\langle \psi_1, K_{\psi_1} \rangle, \langle \psi_2, K_{\psi_2} \rangle$ be pairs such that ψ_1, ψ_2 are (A,V,T)-formulas and $K_{\psi_1} \subseteq node(\psi_1)$ and $K_{\psi_2} \subseteq node(\psi_2)$.*
   $\langle \psi_1, K_{\psi_1} \rangle \approx_K \langle \psi_2, K_{\psi_2} \rangle$ iff there is a bijection $f : K \to K$ such that $\psi_2 = \hat{f}(\psi_1)$ and $K_{\psi_2} = \{k \in K |$ there is a $k' \in K_{\psi_1}$ such that $k = f_K(k')\}$.

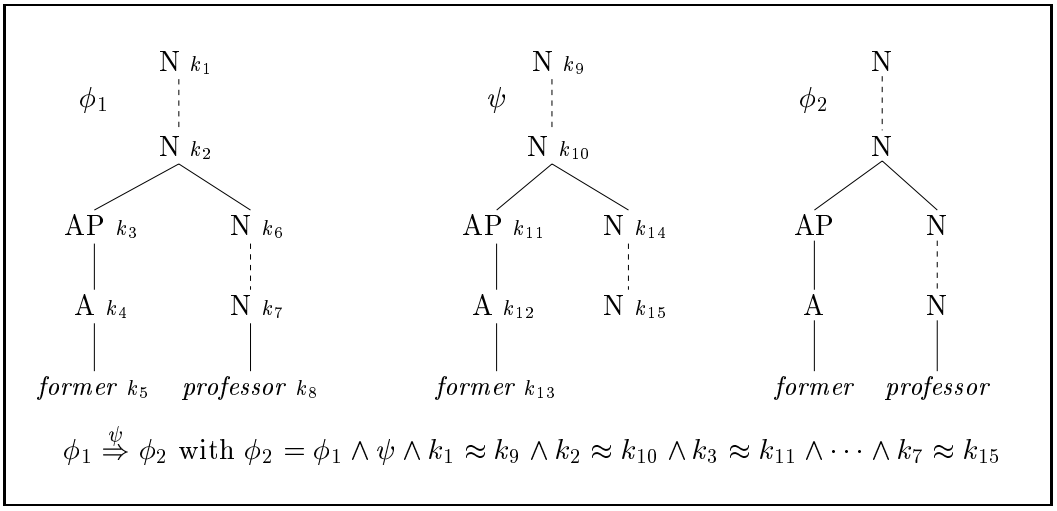


FIGURE 3.4: A QUASI-TREE LIKE DERIVATION THAT IS NOT ALLOWED IN TDGS

$\langle \psi_1, K_{\psi_1} \rangle$ and $\langle \psi_2, K_{\psi_2} \rangle$ are then called K-equivalent.

Clearly, $\approx_K$ is an equivalence relation because f is a bijection. For two K-equivalent formulas ϕ and ϕ', the following holds for all trees B: $B \models \phi$ iff $B \models \phi'$.

A derivation in a TDG starts with the start description ϕ_S. In each derivation step the description that is already derived is extended by adding a new elementary description. One can think of a TDG derivation step as building the conjunction of the old description, a new elementary description and additional node name equivalences. These equivalences are such that for a name k in the new elementary description, an equivalent name is found in the old description iff k is marked or minimal.

The new elementary description added in a derivation step should really represent a further restriction of the old description. A derivation as shown in Fig. 3.4 where ϕ_2 is derived from ϕ_1 by adding an elementary description ψ should not be allowed, since in this case the new subtree description in ψ was identified with a subtree description in ϕ_1 and thereby ψ did not cause a further restriction of the derived description (all trees satisfying ϕ_1 satisfy ϕ_2 as well). To avoid such cases, the two descriptions involved in a derivation step should be put together in such a way that a sort of linearity condition is respected: each subtree description in one of the two descriptions should occur in the result of the derivation step and therefore it should not be possible to identify two parent relations with each other as it is the case in Fig. 3.4. On the other hand, the derivation definition must ensure that the result of a derivation step is determined by the two old descriptions in the sense that it does not describe more subtrees than the two single descriptions together. In particular, new parent relations must not be added.

Finally, as already observed when the example of Fig. 3.1 on page 54 was considered, the result of a derivation step must be as underspecified as possible. Roughly said, this means that it must denote as many trees as possible. To illustrate this, consider again the case of Fig. 3.1. The desired result is $\phi_1 := \gamma_1 \wedge \beta \wedge n_1 \approx n_5 \wedge n_4 \approx n_8$. An alternative result that must be

excluded by the condition of maximal underspecification is $\phi_2 := \gamma_1 \wedge \beta \wedge n_1 \approx n_5 \wedge n_2 \approx n_8$. A direct comparison via syntactic consequence does not work since neither $\phi_1 \vdash \phi_2$ nor $\phi_2 \vdash \phi_1$ holds (e.g. $\phi_1 \vdash n_2 \not\approx n_8$ and $\phi_2 \vdash n_2 \approx n_8$). In order to render such a comparison possible, the new equivalences must be first "eliminated". This can be done by replacing in the new description each occurrence of a marked or minimal name from the elementary description by the equivalent name from the old derived description. In the case of ϕ_1 and ϕ_2 this would eliminate n_5 and n_8, and the results would be $\phi_1' = \gamma_1 \wedge n_1 \lhd^* n_6 \wedge \cdots \wedge n_7 \lhd^* n_4$ and $\phi_2' = \gamma_1 \wedge n_1 \lhd^* n_6 \wedge \cdots \wedge n_7 \lhd^* n_2$ respectively. After this modification, a comparison via syntactic consequence is possible and it captures the intuitive notion of "more" or "less underspecified". In the case of ϕ_1' and ϕ_2', $\phi_2' \vdash \phi_1'$ and $\phi_1' \nvdash \phi_2'$ holds which signifies that ϕ_1 is more underspecified than ϕ_2. The condition of maximal underspecification is formalized by axiom *(A5)* in the following definition.

Definition 3.13 (Derivation in a TDG) *Let $G = (A, V, T, D, \phi_S)$ be a TDG, ϕ_1, ϕ_2 (A, V, T)-descriptions and $\langle \psi_e, K_{\psi_e} \rangle \in D$.*

1. *ϕ_2 can be derived in one step from ϕ_1 using $\langle \psi_e, K_{\psi_e} \rangle$ (written $\phi_1 \overset{\psi_e}{\Rightarrow} \phi_2$ or $\phi_1 \overset{\langle \psi_e, K_{\psi_e} \rangle}{\Rightarrow} \phi_2$), iff there is a pair $\langle \psi, K_\psi \rangle$ with $\langle \psi, K_\psi \rangle \approx_K \langle \psi_e, K_{\psi_e} \rangle$ such that $node(\phi_1)$ and $node(\psi)$ are disjoint, and*

   (A1) $\phi_2 \vdash \phi_1 \wedge \psi$.

   (A2) *For all $k_\psi \in node(\psi)$:*
   - *there is a $k \in node(\phi_1)$ with $\phi_2 \vdash k_\psi \approx k$ iff $k_\psi \in M_\psi$ and*
   - *if there are $k_1, k_2 \in node(\phi_1)$ with $\phi_2 \vdash k_1 \approx k_\psi \wedge k_2 \approx k_\psi$, then $\phi_1 \vdash k_1 \approx k_2$.*

   (A3) *For all $k_1 \in node(\phi_1), k_2 \in node(\psi)$, such that $\phi_2 \vdash k_1 \approx k_2$:*
   there are no $k_1' \in node(\phi_1), k_2' \in node(\psi)$, such that $\phi_1 \vdash k_1 \lhd k_1'$ and $\psi \vdash k_2 \lhd k_2'$.

   (A4) *For all k_1, k_2, such that $\phi_2 \vdash k_1 \lhd k_2$: there are k_3, k_4, such that $\phi_2 \vdash k_1 \approx k_3 \wedge k_2 \approx k_4$ and $\phi_1 \wedge \psi \vdash k_3 \lhd k_4$.*

   (A5) *For all ϕ_3 fulfilling (A1) - (A4):*
   Let $f_1, f_2 \in K^K$ be mappings with $f_1(k) = f_2(k) = k$ for all $k \notin M_\psi$, and for all $k_m \in M_\psi$, there are $k_1, k_2 \in node(\phi_1)$ with $\phi_2 \vdash k_1 \approx k_m$ and $\phi_3 \vdash k_2 \approx k_m$ such that $f_1(k_m) = k_1$ and $f_2(k_m) = k_2$.
   Then the following holds: if $\hat{f}_1(\phi_2) \vdash \hat{f}_2(\phi_3)$, then $\hat{f}_2(\phi_3) \vdash \hat{f}_1(\phi_2)$.

2. *ϕ_2 can be derived in one step from ϕ_1, $\phi_1 \Rightarrow \phi_2$, iff there is an elementary description ψ such that $\phi_1 \overset{\psi}{\Rightarrow} \phi_2$.*
3. *ϕ' can be derived from ϕ (in finitely many steps), $\phi \overset{*}{\Rightarrow} \phi'$, iff there are descriptions $\phi_1, \cdots \phi_n$, such that $\phi = \phi_1 \Rightarrow \phi_2 \cdots \Rightarrow \phi_n = \phi'$.*

With *(A1)* and *(A2)* the result of a derivation step implies a conjunction of the two descriptions involved in this step and additional equivalences of node names.

According to *(A2)* exactly for those names in the new elementary description ψ that are minimal or marked names, there must be equivalent names in the old description ϕ_1. Furthermore, for each marked or minimal name k in ψ, the equivalent name in the old ϕ_1 must be unique up to $\approx$, i.e. unique in the following sense: there may be two names k_1 and k_2 in the old ϕ_1 that are both equivalent to k in the new derived description. But if this is

the case, then k_1 and k_2 must already be equivalent to each other in ϕ_1. One can say that the marking of names somehow specifies how to put the two descriptions (the old ϕ_1 and the new elementary ψ) together.

(A3) states that two node names cannot be identified if they both describe internal nodes in completely described subtrees (e.g. n_2 and n_6 in Fig. 3.1 cannot be identified). At least one of them must be a leaf name or on the left side of a strong dominance in the old descriptions. This formalizes the linearity condition mentioned above. As a further consequence of *(A3)*, the following holds for all $k_1 \in node(\phi_1)$ and $k_2 \in node(\psi)$ with $\phi_2 \vdash k_1 \approx k_2$: there are no $k_1' \in node(\phi_1), k_2' \in node(\psi)$, such that $\phi_1 \vdash k_1' \lhd k_1$ and $\psi \vdash k_2' \lhd k_2$. If there were such k_1', k_2', then (because of the uniqueness of the parent node, inference rule T_5) $\phi_2 \vdash k_1' \approx k_2'$ would also holds and this would be a contradiction to *(A3)*.

(A4) simply says that in a derivation step no new $\lhd$ relations can be added. Therefore the result of the derivation cannot describe more subtrees than the two single descriptions do.

Because of *(A1)* and the transitivity of the relation "$\vdash$" the following holds:

Proposition 3.6 (Monotony of the derivation) *If* $\phi_1 \stackrel{*}{\Rightarrow} \phi_2$, *then* $\phi_2 \models \phi_1$.

Therefore, in a TDG derivation $\phi_1 \stackrel{*}{\Rightarrow} \phi_2$, each tree satisfying ϕ_2 also satisfies ϕ_1, i.e. in contrast to TAGs, the derivation process in TDGs is monotonic in the sense that in the course of the derivation, the derived description becomes more and more specific and at the same time the set of trees satisfying the description decreases in a monotonic way.

The examples in Fig. 3.5 and 3.7 show substitution-like and adjunction-like derivation steps in TDGs. (Since the syntax-semantics interface presented in Chapter 5 is developed for a fragment of French, most of the examples in the following will be in French, too.) In the graphical representations, marked names are equipped with an asterisk.

For a derivation step similar to substitution (see Fig. 3.5) an elementary description $\langle \psi, K_\psi \rangle$ with $K_\psi = \emptyset$ must be added. Then only for the minimal names in ψ there are equivalent node names in the old description. However, an empty set of marked names is not sufficient to guarantee that the derivation step is a kind of substitution operation i.e. that the minimal name of the elementary description becomes equivalent to a leaf name. The elementary description ψ in the derivation step shown in Fig. 3.6 for example has no marked names. But in this derivation step, the minimal name of ψ is not identified with a leaf name in the old description. Instead, it is identified with a name that is part of a strong dominance. Since the result of the derivation step must be a description and therefore in particular satisfy the forth condition in the definition of descriptions, Def. 3.10 on page 73, a further dominance must be added besides the new node name equivalence. In Section 3.3, the derivation process in TDGs will be examined more closely, and I will show which dominances must be added in addition to the old descriptions.

For an adjunction-like derivation step $\langle \psi, K_\psi \rangle$ must be such that all marked names are equivalent and they are all leaf names. This corresponds to the foot node in an auxiliary tree in a TAG. In the old description ϕ_1 there must be a strong dominance $k_1 \lhd^* k_2$. Then a derivation step adding ψ can consist of putting ψ between k_1 and k_2, which is like an adjunction in a TAG (see Fig. 3.7). Furthermore if $\phi_1 \vdash \delta(k_1) \not\approx \delta(k_2)$, then this is comparable to an OA-condition (obligatory adjunction) in a TAG: since $\phi_1 \vdash k_1 \not\approx k_2$ holds, for each tree B satisfying ϕ_1, there is a subtree between the node denoted by k_1 and the node denoted

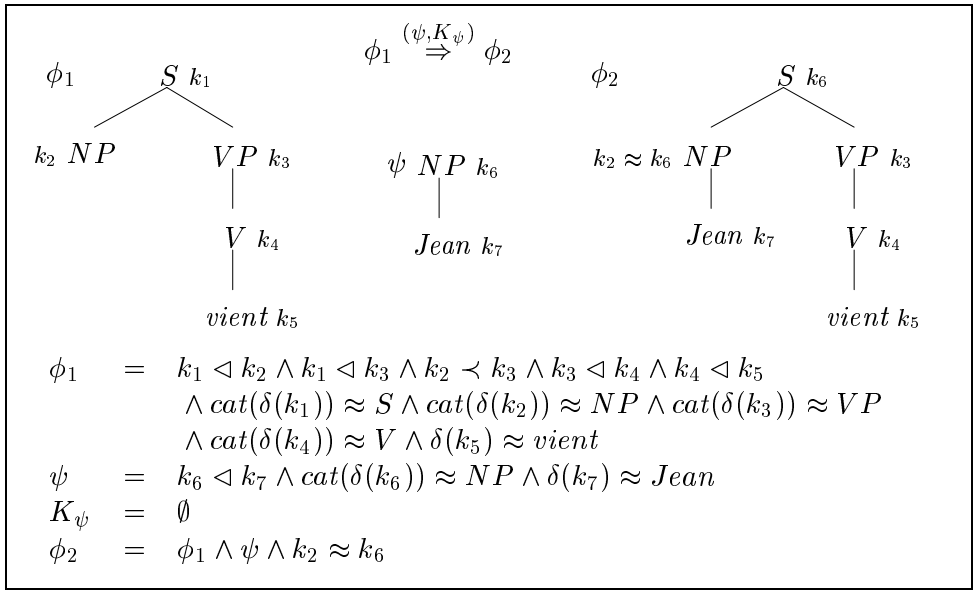


FIGURE 3.5: *Jean vient ('Jean comes')*: SUBSTITUTION-LIKE DERIVATION STEP

by k_2. In other words, between these two node names, some further part of an elementary description must be inserted in order to obtain a description that has a minimal tree (the definition of minimal trees will be given in subsection 3.2.3). Derivation steps as shown in Fig. 3.7 correspond to the quasi-tree adjunction proposed in Vijay-Shanker 1992 (see also Chapter 2). However, such an elementary description as ψ in Fig 3.7 does not guarantee that the derivation step is like an adjunction. E.g. the result of the derivation step in Fig. 3.1 is the underspecified γ_2.

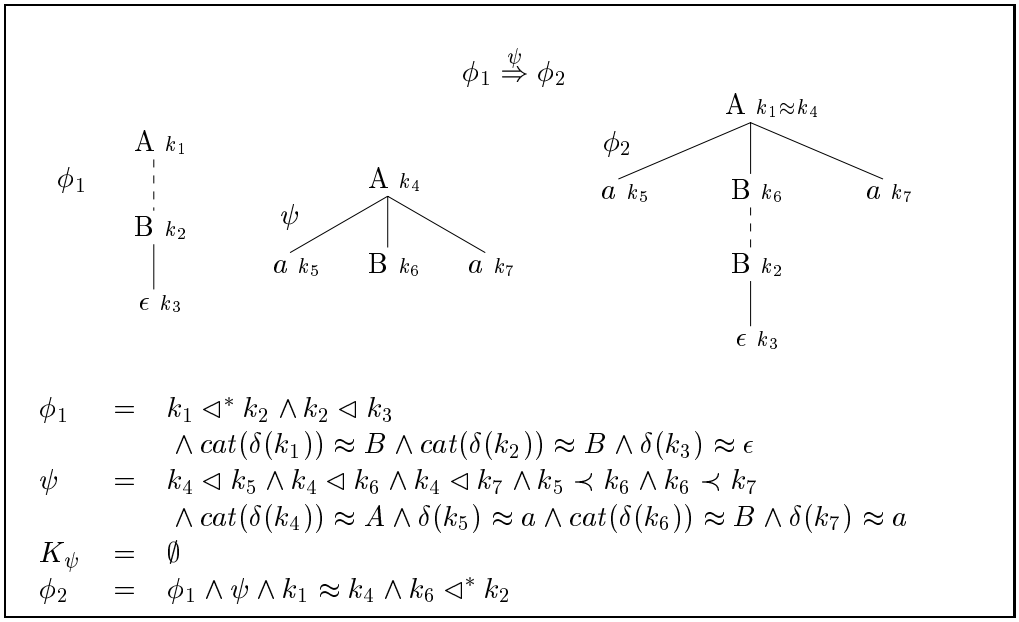
3.2.3 Language generated by a TDG

Since a TDG consists of descriptions and the derivation mode is concerned with descriptions, the set generated by a TDG is a set of descriptions. This set is called the *description language* of the grammar. Depending on its description language, the tree language of a TDG is defined as the set containing the *minimal* trees of the derived descriptions. The string language is then the set of strings yielded by these trees.

First, the description language is defined as the set of all descriptions derived from the start description. A possible start description for example is $\phi_S = cat(\delta(k_1)) \approx S, cat \in A, S \in V$ in order to specify that the root has the category S.

Definition 3.14 (Description language of a TDG) *Let $G = (A, V, T, D, \phi_S)$ be a TDG. The description language of G is $L_D(G) := \{\phi \mid \phi_S \overset{*}{\Rightarrow} \phi\}$.*

Now the minimal trees of a description ϕ must be defined. The idea is that a minimal tree represents one of the readings that are all described in the (possibly underspecified) description. If there is a strong dominance $k_1 \lhd^* k_2$ in the description such that the labels do



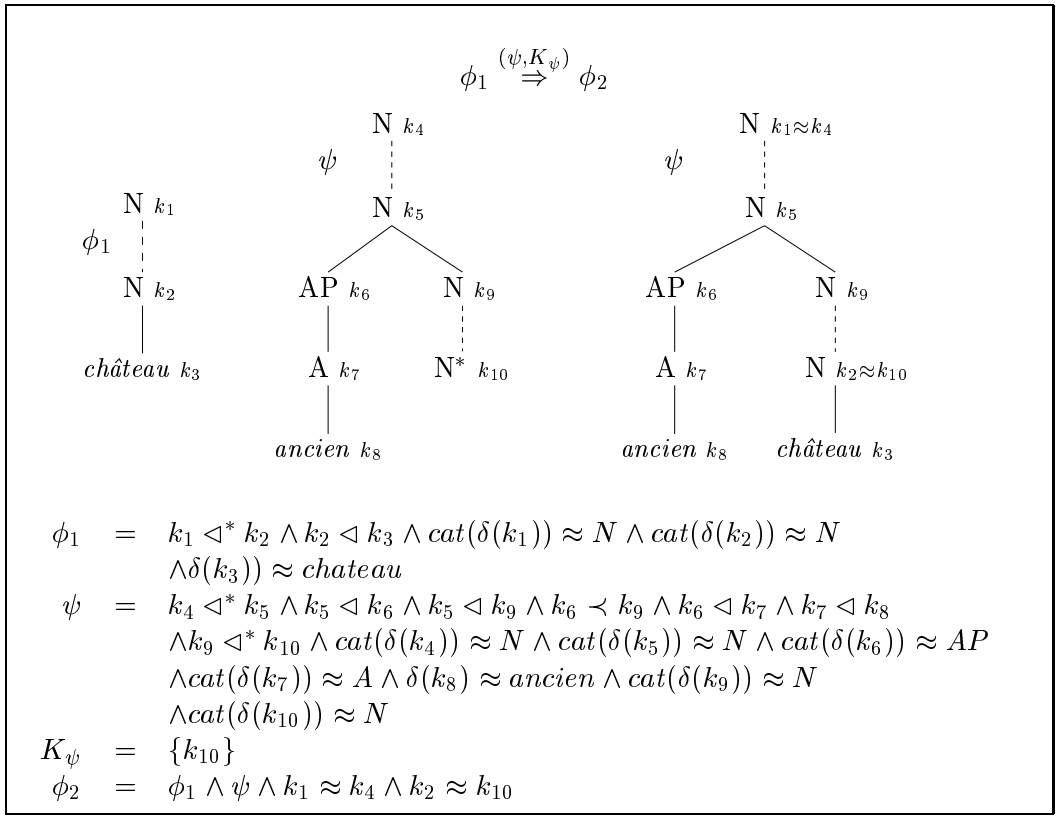
$$
\begin{array}{lcl}
\phi_1 & = & k_1 \lhd^* k_2 \wedge k_2 \lhd k_3 \\
 & & \wedge\, cat(\delta(k_1)) \approx B \wedge cat(\delta(k_2)) \approx B \wedge \delta(k_3) \approx \epsilon \\
\psi & = & k_4 \lhd k_5 \wedge k_4 \lhd k_6 \wedge k_4 \lhd k_7 \wedge k_5 \prec k_6 \wedge k_6 \prec k_7 \\
 & & \wedge\, cat(\delta(k_4)) \approx A \wedge \delta(k_5) \approx a \wedge cat(\delta(k_6)) \approx B \wedge \delta(k_7) \approx a \\
K_\psi & = & \emptyset \\
\phi_2 & = & \phi_1 \wedge \psi \wedge k_1 \approx k_4 \wedge k_6 \lhd^* k_2
\end{array}
$$

FIGURE 3.6: A DERIVATION STEP WITHOUT MARKED NAMES THAT DOES NOT RESEMBLE TO SUBSTITUTION

not allow an equivalence between k_1 and k_2 (i.e. $\phi \vdash k_1 \not\approx k_2$), then ϕ is supposed to have no minimal tree. The strong dominance $k_1 \lhd^* k_2$ is a kind of obligatory adjunction constraint, and therefore it is intended to signify that something must be inserted into this strong dominance in order to obtain a description with minimal trees. In this respect, the definition of *minimal referents* in Rogers 1994 and Rogers and Vijay-Shanker 1994 is too general for our purposes. The concept of minimal referents is such that each satisfiable description has at least one minimal referent.

Minimal trees of a description ϕ are defined as follows: such a tree must be terminal, i.e. all leaves must have terminal labels, it must satisfy ϕ, and it has to be minimal in the sense that its $\mathcal{P}$ relation is already described in ϕ. Furthermore, two node names k_1, k_2 in ϕ_1 with $\phi_1 \not\vdash k_1 \approx k_2$ can denote the same node in the tree, but if this is the case, then there must be no k_3, k_4 such that either $\phi \vdash k_1 \lhd k_3 \wedge k_2 \lhd k_4$ or $\phi \vdash k_3 \lhd k_1 \wedge k_4 \lhd k_2$. In other words, if two node names are not equivalent in the description but describe the same node in a minimal tree, then they both have to be part of some strong dominance in the description. This is similar to the axiom *(A3)* for the derivation in TDGs. As a consequence, two subtree descriptions in ϕ must describe two different subtrees of the tree. E.g. in Fig. 3.8 the trees B_1 and B_2 both satisfy ϕ in such a way that all parent relations in the trees are described in ϕ. However, neither B_1 nor B_2 are intended to be minimal for ϕ because ϕ is supposed to express that there are two different subtrees of the form of B_1 and some other node dominating these two subtrees.

Definition 3.15 (Minimal tree) *Let $G = (A, V, T, D, \phi_S)$ be a TDG, $\phi \in L_D(G)$. Let B be a finite labelled tree. B is a minimal tree of ϕ iff B is a terminal tree wrt the nonterminal*



$$
\begin{array}{rcl}
\phi_1 & = & k_1 \lhd^* k_2 \wedge k_2 \lhd k_3 \wedge cat(\delta(k_1)) \approx N \wedge cat(\delta(k_2)) \approx N \\
 & & \wedge \delta(k_3)) \approx chateau \\
\psi & = & k_4 \lhd^* k_5 \wedge k_5 \lhd k_6 \wedge k_5 \lhd k_9 \wedge k_6 \prec k_9 \wedge k_6 \lhd k_7 \wedge k_7 \lhd k_8 \\
 & & \wedge k_9 \lhd^* k_{10} \wedge cat(\delta(k_4)) \approx N \wedge cat(\delta(k_5)) \approx N \wedge cat(\delta(k_6)) \approx AP \\
 & & \wedge cat(\delta(k_7)) \approx A \wedge \delta(k_8) \approx ancien \wedge cat(\delta(k_9)) \approx N \\
 & & \wedge cat(\delta(k_{10})) \approx N \\
K_\psi & = & \{k_{10}\} \\
\phi_2 & = & \phi_1 \wedge \psi \wedge k_1 \approx k_4 \wedge k_2 \approx k_{10}
\end{array}
$$

Figure 3.7: *ancien château ('former castle')*: adjunction-like derivation step

alphabet $Pow(A \times V) \setminus \emptyset$ *and the terminal alphabet* T*, and there is an assignment function* g *such that*

1. $B \models_g \phi$.
2. *For all* u_1, u_2 *in* $\mathcal{U}$*: if* $\langle u_1, u_2 \rangle \in \mathcal{P}$*, then there are* $k_1, k_2 \in K$ *such that* $\phi \vdash k_1 \lhd k_2$ *and* $g(k_1) = u_1, g(k_2) = u_2$.
3. *For all* k_1, k_2 *with* $g(k_1) = g(k_2)$*: if* $\phi \not\vdash k_1 \approx k_2$*, then there are no* k_3, k_4 *such that* $\phi \vdash k_1 \lhd k_3 \wedge k_2 \lhd k_4$.

A description $\phi \in L_D(G)$ not only can have more than one minimal tree (e.g. γ_2 in Fig. 3.1) but it can even have no minimal tree. This is for example the case for the description ϕ in Fig. 3.8.

In order to find the set of minimal trees for a given description ϕ, a fully specified description (i.e. a description without strong dominances) is first constructed. For this purpose, it is sufficient to add term equivalences to ϕ, i.e. equivalences between node names and equivalences between labels or attribute values. I will examine this process more closely in the following section where decidability of the set of minimal trees for a given description will be shown. Here I want to restrict myself to an example. In order to obtain a minimal tree B for the description ϕ shown in Fig. 3.9, the strong dominances in this description must be

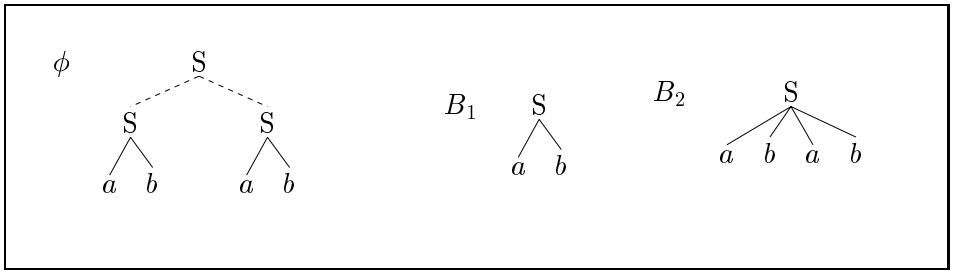


FIGURE 3.8: TREES THAT ARE NOT MINIMAL

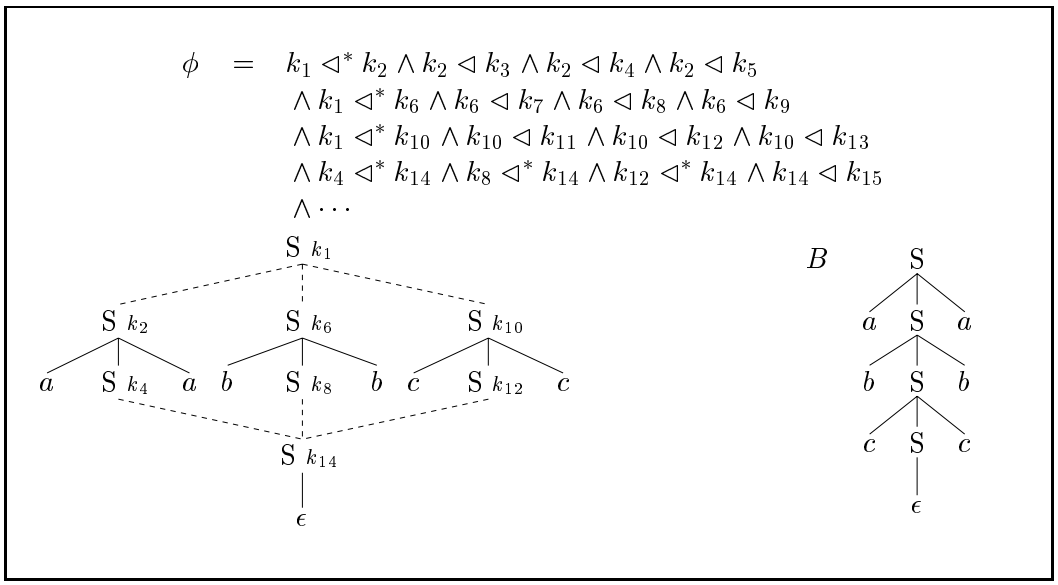


FIGURE 3.9: SAMPLE DESCRIPTION WITH ONE OF ITS MINIMAL TREES

eliminated by adding further equivalences between node names. First, an equivalent name for k_1 must be found. This can be either k_2 or k_6 or k_{10}. Suppose that $k_1 \approx k_2$ was added. This signifies that B satisfies $\phi \wedge k_1 \approx k_2$. Because of the third condition for minimal trees, B must then also satisfy $k_4 \lhd^* k_6 \wedge k_4 \lhd^* k_{10}$. Then B might satisfy either $k_4 \approx k_6$ or $k_4 \approx k_{10}$ and therefore one of these equivalences must be added. Suppose that $k_4 \approx k_6$ is added, i.e. B now satisfies $\phi \wedge k_1 \approx k_2 \wedge k_4 \approx k_6 \wedge k_4 \lhd^* k_{10}$. With the third condition for minimal trees, B must also satisfy $k_8 \lhd^* k_{10}$, and therefore the last two equivalences that must be added are $k_8 \approx k_{10}$ and $k_{12} \approx k_{14}$. The resulting description $\phi \wedge k_1 \approx k_2 \wedge k_4 \approx k_6 \wedge k_8 \approx k_{10} \wedge k_{12} \approx k_{14}$ fully specifies one of the minimal trees of ϕ, namely the tree B shown in Fig. 3.9. B is not the only minimal tree of ϕ. Altogether, ϕ has $3! = 6$ minimal trees.

An interesting question when thinking about minimal trees is whether there is a corresponding notion of minimality with respect to some ordering relation on the models of the description in question. In other words whether it is possible to define the minimal trees of a description ϕ based only on a partial order defined on the models of ϕ. As far as I see

this is not possible for the following reasons: First of all, a general definition of a partial order on the models of a description ϕ that is independent from ϕ such that the minimal trees of ϕ only depend on this order cannot exist, since there are descriptions that have the same sets of models but whose minimal trees are different. E.g., the description $\phi' := k_1 \lhd^* k_2 \wedge k_2 \lhd k_3 \wedge k_2 \lhd k_4 \wedge k_3 \prec k_4 \wedge cat(\delta(k_1)) \approx S \wedge cat(\delta(k_2)) \approx S \wedge \delta(k_3)) \approx a \wedge \delta(k_4)) \approx b$ is satisfied by the same models as the description ϕ in Fig. 3.8 but, in contrast to ϕ, ϕ' has a minimal tree, namely the tree B_1 in Fig. 3.8. Even a partial order depending on the description in question that defines minimal trees probably does not exist in general: If one supposes that such an order exists, then this order must express something like "has less parent-relations" or "has less sister-relations (linear precedence with a common parent)". Furthermore, the minimal trees are not always unique. Consequently, a tree B would be defined as being minimal for a description ϕ if there was no smaller model of ϕ ("smaller" in the sense of the partial order). But then there would arise a problem with descriptions that do not have a minimal tree. In these cases for each model of the description there should exist a smaller model, and this seems to be contradictory. These considerations indicate that for the definition of minimal trees of a description given above, there is probably no corresponding notion of minimality with respect to some partial order on trees.

The tree language of a TDG is defined as the set of minimal trees of the descriptions derived from the start description. The string language is the set of strings yielded by the trees in the tree language:

Definition 3.16 (Tree language of a TDG) *Let $G = (A, V, T, D, \phi_S)$ be a TDG. The tree language of G is $L_T(G) := \{B \mid B$ is minimal tree of a $\phi \in L_D(G)\}$.*

Definition 3.17 (String language of a TDG, TDL)

1. *Let $G = (A, V, T, D, \phi_S)$ be a TDG. The string language of G is $L_S(G) := \{w \mid$ there is a $B \in L_T(G)$ such that $w = yield(B)\}$.*
2. *A language $L \subseteq T^*$ is a Tree Description Language (TDL), iff there is a TDG G such that $L_S(G) = L$.*

Examples of TDGs are given in Fig. 3.10 and 3.11. Fig. 3.10 shows a schema for TDGs generating the string languages $L_n := \{a_1^k a_2^k ... a_n^k \mid 0 \leq k\}$, and Fig. 3.11 shows a schema for TDGs for $L^n_{copy} := \{w^n \mid w \in T^*\}$. For $n \geq 5$, the string languages L_n are no TALs and for $n \geq 3$, L^n_{copy} is no TAL: with the pumping lemma for TALs proven by Vijay-Shanker (1987) (see Prop. 2.1, page 23) it can be shown that for $n \geq 5$ the languages $\{a_1^k a_2^k ... a_n^k \mid 0 \leq k\}$ are no TALs. Together with the closure of TALs with respect to intersection with regular languages, the pumping lemma can also be used to show that for $n \geq 3$, $\{w^n \mid w \in T^*\}$ are no TALs. These two examples show that there are TDLs that are no TALs.

At a first glance, one might even suspect that TALs are a subset of TDLs. However, it is not at all obvious how to construct an equivalent TDG for a given TAG. If nodes allowing adjunction are simply replaced by strong dominances (as in Fig. 2.15 on page 42 and 2.16 on page 43 in Chapter 2), then one usually gets a superset of the original string language. The problem is that the tree description corresponding to an auxiliary tree need not necessarily be inserted into one single strong dominance. Whereas in TAGs adjunction takes place at one single node, TDGs are non-local in the sense that there can be arbitrary many node

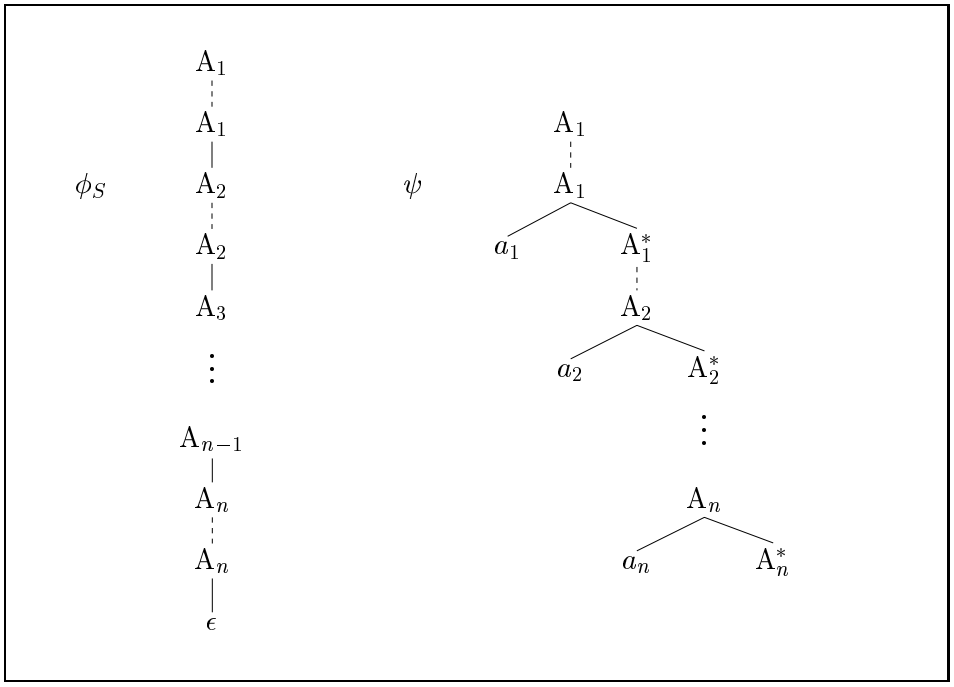


FIGURE 3.10: TDG FOR $\{a_1^k \dots a_n^k \mid k \in \mathbb{N}\}$

names between the two names that are chosen as equivalent names for the minimal and the marked name in such a description corresponding to an auxiliary tree. Therefore, I cannot say anything about a subset relation between TALs and TDLs. In Chapter 4 we will see that this is different with local TDGs. Local TDGs generate a true superset of TALs, since it is no problem to construct an equivalent local TDG for each given TAG (even for each given set-local MC-TAG).

3.3 Formal properties of TDG derivations

In this section I will first show some properties of descriptions derived in TDGs. Then, I will give a derivation algorithm, and show the decidability of lexicalized TDLs.

3.3.1 Description language

In the following, some properties of the result of a TDG derivation step are proven.

With the definition of the derivation, in particular with *(A4)* and *(A5)*, it can be shown that a derivation step does not add any new node names, i.e. that the result of the derivation contains only node names occuring in the single descriptions involved in that derivation.

Proposition 3.7 *If* $\phi_1 \overset{(\psi,K_\psi)}{\Rightarrow} \phi_2$ *in a TDG, then* $node(\phi_2) = node(\phi_1) \cup node(\psi)$.

Proof: Let $\phi_1 \overset{(\psi,K_\psi)}{\Rightarrow} \phi_2$ hold.

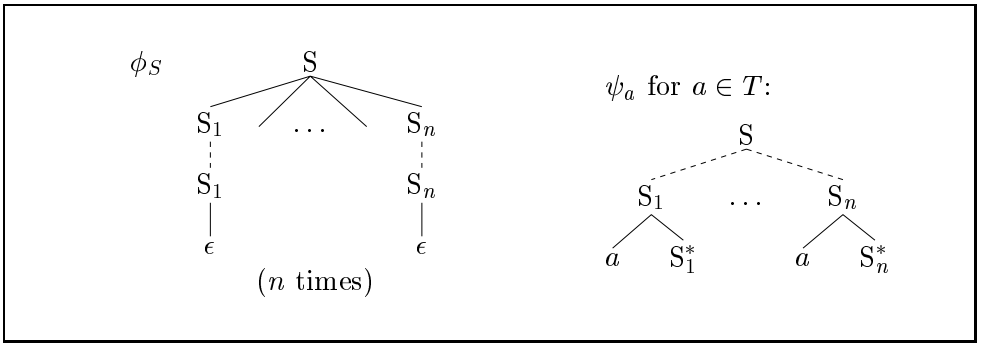
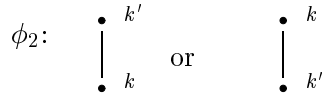
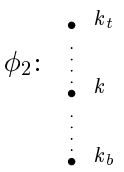


FIGURE 3.11: TDG FOR $\{w^n \mid w \in T^*\}$

1. To show: $node(\phi_1) \cup node(\psi) \subseteq node(\phi_2)$.
   ϕ_1 and ψ are descriptions
   $\Rightarrow$ there is a $k_r^{\phi_1}$ such that $\phi_1 \vdash k_r^{\phi_1} \lhd^* k_{\phi_1}$ for all $k_{\phi_1} \in node(\phi_1)$, and there is a k_r^{ψ} such that $\psi \vdash k_r^{\psi} \lhd^* k_{\psi}$ for all $k_{\psi} \in node(\psi)$.
   $\Rightarrow$ (transitivity of $\vdash$) $\phi_2 \vdash k_r^{\phi_1} \lhd^* k_{\phi_1}$ for all $k_{\phi_1} \in node(\phi_1)$ and $\phi_2 \vdash k_r^{\psi} \lhd^* k_{\psi}$ for all $k_{\psi} \in node(\psi)$.
   $\Rightarrow node(\phi_1) \cup node(\psi) \subseteq node(\phi_2)$.
2. To show: $node(\phi_2) \subseteq node(\phi_1) \cup node(\psi)$.
   Suppose there is a $k \in node(\phi_2) \setminus (node(\phi_1) \cup node(\psi))$.
   (i) 1. case: there is a $k' \in node(\phi_1) \cup node(\psi)$ such that $\phi_2 \vdash k \approx k'$.
   Let ϕ_2' be the result of replacing all occurrences of k in ϕ_2 by k'. ϕ_2' is a description satisfying $(A1)$ - $(A4)$, and $\phi_2 \vdash \phi_2'$ and $\phi_2' \not\vdash \phi_2$ hold.
   Contradiction to $(A5)$.
   (ii) 2. case: There is no $k' \in node(\phi_1) \cup node(\psi)$ such that $\phi_2 \vdash k \approx k'$, and there is a $k' \in node(\phi_2)$ such that $\phi_2 \vdash k' \lhd k$ or $\phi_2 \vdash k \lhd k'$. Contradiction to $(A4)$.



   (iii) 3. case: There is no $k' \in node(\phi_1) \cup node(\psi)$ such that $\phi_2 \vdash k \approx k'$, and there is a k_t such that $\phi_2 \vdash_s k_t \lhd^* k$, and k either is a leaf name in ϕ_2 or there is a k_b such that $\phi_2 \vdash_s k \lhd^* k_b$.



   Let ϕ_2' be the result of first adding to ϕ_2 all conjuncts of the form k_1 R k_2 for R $\in \{\lhd^*, \prec\}$ such that $\phi_2 \vdash k_1$ R k_2 and $k_1, k_2 \neq k$ and then removing all conjuncts containing an occurrence of k.

ϕ_2' is a description, because:

1. ϕ_2' is satisfiable because $\phi_2 \vdash \phi_2'$ holds and ϕ_2 is satisfiable.
2. each name that is minimal in ϕ_2 is also minimal in ϕ_2'.
3. ϕ_2' is negation and disjunction free since ϕ_2 is negation and disjunction free.
4. For all $k_1, k_2, k_3 \in node(\phi_2')$: If $\phi_2' \vdash k_1 \lhd k_2 \wedge k_1 \lhd^* k_3$, then $\phi_2 \vdash k_1 \lhd k_2 \wedge k_1 \lhd^* k_3$ also holds (because of $\phi_2 \vdash \phi_2'$). Therefore either $\phi_2 \vdash k_1 \approx k_3$ or there is a $k_4 \in node(\phi_2)$ with $\phi_2 \vdash k_1 \lhd k_4 \wedge k_4 \lhd^* k_3$. In the second case, $k_4 \in node(\phi_2')$ holds because of (ii). Then (construction of ϕ_2'), either $\phi_2' \vdash k_1 \approx k_3$ or there is a $k_4 \in node(\phi_2')$ with $\phi_2' \vdash k_1 \lhd k_4 \wedge k_4 \lhd^* k_3$.
5. For all $k_1, k_2, k_3 \in node(\phi_2')$: If $\phi_2' \vdash k_1 \lhd k_2 \wedge k_1 \lhd k_3$, then (with (ii)) $k_1, k_2, k_3 \in node(\phi_2)$, and $\phi_2 \vdash k_1 \lhd k_2 \wedge k_1 \lhd k_3$. Therefore either $\phi_2 \vdash k_2 \approx k_3$ and $\phi_2' \vdash k_2 \approx k_3$, or $\phi_2 \vdash k_2 \prec k_3$ and $\phi_2' \vdash k_2 \prec k_3$, or $\phi_2 \vdash k_3 \prec k_2$ and $\phi_2' \vdash k_3 \prec k_2$.

$(A1)$ - $(A4)$ hold for ϕ_2', and $\phi_2 \vdash \phi_2'$ and $\phi_2' \nvdash \phi_2$ hold. Contradiction to $(A5)$.

(iv) 4. case: There is no $k' \in node(\phi_1) \cup node(\psi)$ such that $\phi_2 \vdash k \approx k'$, and k is minimal in ϕ_2: because of (i)-(iii), for all $k' \in node(\phi_2) \setminus (node(\phi_1) \cup node(\psi))$, $\phi_2 \vdash k \approx k'$ holds. Let ϕ_2' be the result of removing all conjuncts with occurrences of any element of $node(\phi_2) \setminus (node(\phi_1) \cup node(\psi))$ from ϕ_2. ϕ_2' is a description where each name that is minimal in ϕ_1, is also minimal in ϕ_2'.

$(A1)$ - $(A4)$ hold for ϕ_2', and $\phi_2 \vdash \phi_2'$ and $\phi_2' \nvdash \phi_2$ hold. Contradiction to $(A5)$.

□

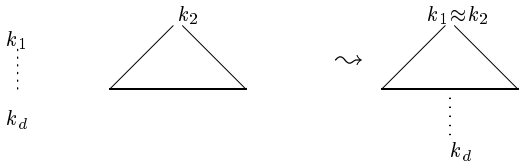
The following three lemmata are concerned with the new conjuncts that must be added in a derivation step besides the old descriptions and the new node name equivalences. Lemma 3.3 states that no new linear precedence relations must be added in order to find the result of a derivation step. In other words, all linear precedence relations in the resul of a derivation step are consequences of linear precedence relations in the old description and dominance relations in the resulting description. This lemma is a consequence of the derivation axiom *(A5)*. The proof of Lemma 3.3 can be found in the appendix on page 233.

Lemma 3.3 *If $\phi_1 \overset{\psi}{\Rightarrow} \phi_2$ is a derivation step in a TDG, then*

(LP) *for all k_1, k_2: $\phi_2 \vdash k_1 \prec k_2$ iff there are k_1', k_2' with $\phi_2 \vdash k_1' \lhd^* k_1 \wedge k_2' \lhd^* k_2$ and $\phi_1 \wedge \psi \vdash k_1' \prec k_2'$.*

In contrast to linear precedence, there may be new dominance relations in the result of a derivation step that are not implied by the old descriptions. In other words, in a derivation procedure, some new dominances must be added in addition to those implied by the old descriptions ϕ_1 and ψ and the new node name equivalences. This is necessary because the result must be a description and therefore in particular, it must satisfy the forth condition of Def. 3.10, page 73. According to this condition, for a node name k in the new description ϕ_2 there must not be k_1, k_2 with $\phi_2 \vdash k \lhd k_1$ and $\phi_2 \vdash_s k \lhd^* k_2$, i.e. roughly said k must not be on the left side of a parent relation and of a strong dominance.

Suppose that a new node name equivalence $k_1 \approx k_2$ is added in a derivation step, that for one of these two names, say k_2, there is a daughter k_p described in ϕ_1 or ψ and that for the second name only a strong dominance with a k_d on the right side is described in ϕ_1 or ψ. This means that for the result ϕ_2 of the derivation step with $\phi_2 \vdash \phi_1 \wedge \psi \wedge k_1 \approx k_2$ also $\phi_2 \vdash k_2 \lhd k_p \wedge k_2 \lhd^* k_d$ holds. Therefore, because of condition *4.* for descriptions either $\phi_2 \vdash k_2 \approx k_d$ must hold or there must be a k_2' with $\phi_2 \vdash k_2 \lhd k_2' \wedge k_2' \lhd^* k_d$. Since $\phi_2 \not\vdash k_1 \approx k_d$ follows from axiom *(A2)* for TDG derivations (Def. 3.13), the second one must be the case. With axiom *(A4)*, no new parent relation may be added in a derivation step, and therefore k_2' must be such that $\phi_1 \wedge \psi \vdash k_2 \lhd k_2'$ holds, i.e. in ϕ_2 k_d must be dominated by some daughter of k_2 in ϕ_1 or ψ. In the derivation step shown in Fig. 3.6 on page 79 for example, the new dominance $k_6 \lhd^* k_2$ was added for this reason.



This picture might be misleading, since it shows only one of the possible results: in the result ϕ_2, k_d is not necessarily strongly dominated by some other name. If there is no daughter described for k_d, then k_d might also be identified with a node name that has a mother in ϕ_2.

The new dominances added in cases as sketched above are the only new dominances, i.e. the only dominances in the result of a derivation step that are not implied by the old descriptions and the new node name equivalences:

Lemma 3.4 *If $\phi_1 \overset{\psi}{\Rightarrow} \phi_2$ is a derivation step in a TDG, and $\phi_\approx := \bigwedge\{k_\phi \approx k_\psi \,|\, k_\phi \in node(\phi_1), k_\psi \in node(\psi), \phi_2 \vdash k_\phi \approx k_\psi\}$, then*

(D) *for all k_1, k_2 with $\phi_2 \vdash_s k_1 \lhd^* k_2$:*

(a) either $\phi_1 \wedge \psi \wedge \phi_\approx \vdash k_1 \lhd^* k_2$

(b) or there are $k_m \in M_\psi$, $k \in node(\phi_1)$ and $k' \in node(\phi_2)$ such that

- *$\phi_2 \vdash k_m \approx k$,*
- *either $\psi \vdash_s k_m \lhd^* k_2$ and $\phi_1 \vdash k \lhd k'$, or $\phi_1 \vdash_s k \lhd^* k_2$ and $\psi \vdash k_m \lhd k'$,*
- *and $\phi_2 \vdash k' \lhd^* k_1$.*

The proof of this lemma is quite technical and therefore it is put into the appendix. It can be found on page 234.

The following lemma states that no new formulas with v- or l-terms must be added. It is obvious, that this lemma holds.

Lemma 3.5 *If $\phi_1 \overset{\psi}{\Rightarrow} \phi_2$ is a derivation step in a TDG, then*

(L) *for all $k \in K$, $a \in A$, $v \in V \cup \{\bot\}$ and $t \in T \cup \{\epsilon\}$:*

- *$\phi_2 \vdash a(\delta(k)) \approx v$ iff there is a k' with $\phi_2 \vdash k \approx k'$ and $\phi_1 \wedge \psi \vdash a(\delta(k')) \approx v$, and*
- *$\phi_2 \vdash \delta(k) \approx t$ iff there is a k' with $\phi_2 \vdash k \approx k'$ and $\phi_1 \wedge \psi \vdash \delta(k') \approx t$.*

3.3.2 A derivation procedure

The derivation definition 3.13 given in Section 3.2 on page 76 defines a derivation step by specifying properties that the result of the derivation must have. However, this definition does not tell us how to obtain the result of a derivation step. In the following, this problem will be addressed. I will describe the TDG derivation process by proposing a derivation procedure. For an old description and a new elementary description this procedure gives the set of all results that may be obtained when these two descriptions are put together in one derivation step.

Giving this derivation procedure I intend to show that the TDG derivation process is decidable and I want to examine more closely the way derivation works. However, I do not intend to present an algorithm that is attractive with respect to its complexity. Obviously, from this point of view, there are better derivation procedures than the one presented here.

Clearly, for each TDG G, there is a strongly equivalent TDG G' that allows even derivations of similar tree descriptions such that two names that are equivalent in one of the descriptions in G' are even equal. "Similar tree descriptions" means that there is a description ϕ derived in G iff there is a ϕ' derived in G' such that there is a mapping $f : node(\phi) \rightarrow node(\phi)$ with $\phi \vdash k_1 \approx k_2$ for all k_1, k_2 with $f(k_1) = f(k_2)$ and $\phi' = \hat{f}(\phi)$. In particular, ϕ and ϕ' have the same minimal trees. Therefore, without loss of generality, for the derivation algorithm a TDG $G = \langle A, V, T, D, \phi_S \rangle$ is supposed to be such that for all elementary or start descriptions ψ in G and for all $k_1, k_2 \in node(\psi)$: $\psi \vdash k_1 \approx k_2$ iff $k_1 = k_2$.

This kind of descriptions ϕ are called *K-minimal* since $|node(\phi)|$ is minimal in a certain sense.

Definition 3.18 (K-minimal) *A description ϕ is K-minimal iff for all $k_1, k_2 \in node(\phi)$: $\phi \vdash k_1 \approx k_2$ iff $k_1 = k_2$.*

The derivation algorithm takes a derived description ϕ_1 and an elementary description ψ with $node(\phi_1) \cap node(\psi) = \emptyset$ The output is a set $D^{\psi}_{\phi_1}$ of descriptions ϕ. These descriptions are such that for each result ϕ_2 of a derivation step $\phi_1 \stackrel{\psi}{\Rightarrow} \phi_2$, there is a corresponding ϕ'_2 in $D^{\psi}_{\phi_1}$. This ϕ'_2 can be obtained from ϕ_2 by replacing all $k_m \in M_\psi$ by the equivalent name from ϕ_1.

Roughly, the algorithm is as follows:

- first, the conjunction $\phi_1 \wedge \psi$ is built, new node name equivalences for marked and minimal names in ψ are added, and dominances are added such that **(D)** (see Lemma 3.4) is satisfied The result is a set of descriptions satisfying *(A1)* to *(A4)*, **(LP)**, **(D)** and **(L)**.
- Then for all ϕ and ϕ' in this set, it is checked whether one description is "more underspecified" than the other in the sense of derivation axiom *(A5)*, and in this case the "less underspecified" is removed.

In the proof of Prop. 3.2 I have shown how to construct an equivalent saturated branch for a given formula ϕ. This can be done in finitely many steps, i.e. it is decidable. In order to check whether a formula ϕ is satisfiable, it is sufficient to check whether the saturated branch

```
Input:  two names k1, k2 and a K-minimal description φ
Output:  true or false

begin
    if k1 = k2, then dom(k1, k2, φ) = true,
    else:
       if there is a k, k ≠ k2 with dom(k1, k, φ) = true
          and either k ◁ k2 in φ or k ◁* k2 in φ,
       then return dom(k1, k2, φ) := true,
       else return dom(k1, k2, φ) := false;
end
```

FIGURE 3.12: PROCEDURE `dom`

$\Gamma(\phi, node(\phi))$ contains an empty clause. If this is the case, then ϕ is not satisfiable. Whereas, if this is not the case, then ϕ is satisfiable according to Prop. 3.4. Therefore I will suppose for the following that there is a predicate `satisfiable` for formulas such that `satisfiable(`ϕ`) = true` iff ϕ is satisfiable.

Before coming to the main algorithm, the following procedures are introduced:

1. a procedure `dom` checking for two node names k_1, k_2 in a K-minimal description ϕ whether the first name dominates the second, i.e. whether $\phi \vdash k_1 \lhd^* k_2$ holds,

2. a procedure `strong-dom` checking for two node names k_1, k_2 in a K-minimal description ϕ whether the first name strongly dominates the second, i.e. whether $\phi \vdash_s k_1 \lhd^* k_2$ holds,

3. a procedure `left-of` checking for two node names k_1, k_2 in a K-minimal description ϕ whether the first name is left of the second, i.e. whether $\phi \vdash k_1 \prec k_2$ holds,

4. a procedure `terminal` checking for a name k in a K-minimal description ϕ and for a terminal t whether $\phi \vdash \delta(k) \approx t$ holds,

5. a procedure `attr-closure` adding to a K-minimal description ϕ equivalences between v-terms that are entailed by ϕ,

6. a procedure `attr-value` checking for a name k in a K-minimal description ϕ and for an attribute a and a value v whether $\phi \vdash a(\delta(k)) \approx v$ holds,

7. and a procedure `more-underspecified` checking for two K-minimal descriptions whether one of them is "more underspecified" in the sense of *(A5)* than the other.

In the algorithms, `L1, L2, ...` are labels. Partly they are used as goals for jump commands and partly they are only used to refer to certain parts of the algorithm when proving that the desired result is obtained.

```
Input:  two names k1, k2 and a K-minimal description φ
Output:  true or false

begin
    if dom(k1, k2, φ) = true and k1 ≠ k2 and k1 ◁ k2 not in φ and
       there is no k ∈ node(φ), k ∉ {k1, k2} with
          dom(k1, k, φ) = true and dom(k, k2, φ) = true,
    then return strong-dom(k1, k2, φ) := true,
    else return strong-dom(k1, k2, φ) := false;
end
```

FIGURE 3.13: PROCEDURE `strong-dom`

The dominance checking procedure shown in Fig. 3.12 is very simple. Since the input ϕ is satisfiable and for $k \neq k'$, $\phi \vdash k \lhd^* k' \wedge k' \lhd^* k$ cannot hold, the recursion terminates after finitely many steps. Obviously, for two names k_1 and k_2 and a description ϕ satisfying the conditions specified for the input: $\phi \vdash k_1 \lhd^* k_2$ iff `dom`(k_1, k_2, ϕ) `= true`.

Besides `dom`, a predicate `strong-dom` for strong dominance is also needed. The algorithm is shown in Fig. 3.13.

With the Def. 3.9 of strong dominance for descriptions on page 72, it is clear that for a K-minimal description ϕ and names $k_1, k_2 \in node(\phi)$: `strong-dom`(k_1, k_2, ϕ) `= true` iff $\phi \vdash_s k_1 \lhd^* k_2$.

For the procedure `left-of` in Fig. 3.14 checking linear precedence, the input description ϕ is also supposed to be K-minimal. In the procedure, first a set L will be constructed containing all linear precedences $k_1 \prec k_2$ with $\phi \vdash k_1 \prec k_2$. Then it will be tested whether this set contains the linear precedence in question.

There are two inference rules that can cause the adding of new formulas $k_l \prec k_r$ as consequences, when a saturated branch is constructed for some description ϕ. These are the rules $tr_{\mathcal{L}}$ and T_6. $tr_{\mathcal{L}}$ expresses the transitivity of $\mathcal{L}$, and T_6 expresses the fact that if there are node names k_1, k_2, k_3, k_4 with $\phi \vdash k_1 \prec k_2 \wedge k_1 \lhd^* k_3 \wedge k_2 \lhd^* k_4$, then $\phi \vdash k_3 \prec k_4$. For both rules, the consequences are added to L in `left-of`, and therefore for all descriptions ϕ and $k_1, k_2 \in node(\phi)$: `left-of`(k_1, k_2, ϕ) `= true` iff $\phi \vdash k_1 \prec k_2$.

The following two procedures are used in `more-underspecified`. The first procedure checks for a node name k in a K-minimal description ϕ and for a terminal t whether $\phi \vdash \delta(k) \approx t$ holds. This procedure, called `terminal`, is shown in Fig. 3.15. The second checks for a $k \in node(\phi)$, an attribute $a \in A$ and a value $v \in V \cup \{\bot\}$ whether $\phi \vdash a(\delta(k)) \approx v$ holds. This is called `attr-closure` and is shown in Fig. 3.16.

Since all conjuncts with l-terms occurring in descriptions are either of the form $t \approx t$ or $\delta(k_1) \approx \delta(k_2)$ or of the form $\delta(k) \approx t$ or $t \approx \delta(k)$ for some $k, k_1, k_2 \in K$ and $t \in T \cup \{\epsilon\}$, `terminal`$(k, t, \phi)$ `= true` holds iff $\phi \vdash \delta(k) \approx t$.

In the procedure `attr-closure`, first (see label `L1`) conjuncts $a(\delta(k)) \approx \bot$ for all $a \in A$ and all k with terminal label are added to the K-minimal input description ϕ. In `L2` all

```
Input:  names k_1, k_2 and a K-minimal description φ
Output:  true or false

begin
    L := ∅;
    for all k_l, k_r ∈ node(φ):  if k_l ≺ k_r in φ, then add k_l ≺ k_r to L;
L1: if there are k_l, k, k_r with k_l ≺ k ∈ L and k ≺ k_r ∈ L and k_l ≺ k_r ∉
    L,
      then add k_l ≺ k_r to L and go to L1;
    if there are k_l, k_r, k'_l, k'_r with k_l ≺ k_r ∉ L, k'_l ≺ k'_r ∈ L and
         dom(k'_l, k_l, φ) = dom(k'_r, k_r, φ) = true,
      then add k_l ≺ k_r to L and go to L1;
    if k_1 ≺ k_2 ∈ L, then return left-of(k_1, k_2, φ) := true,
      else return left-of(k_1, k_2, φ) := false;
end
```

FIGURE 3.14: PROCEDURE left-of

```
Input:  a name k, a t ∈ T ∪ {ε} and a K-minimal description φ
Output:  true or false

begin
    if δ(k) ≈ t or t ≈ δ(k) in φ, then terminal(k, t, φ) = true,
    else:
      if there are k_1, ···, k_n for some n, 1 ≤ n ≤ |node(φ)| with
        δ(k) ≈ δ(k_1) or δ(k_1) ≈ δ(k) in φ,
        δ(k_i) ≈ δ(k_{i+1}) or δ(k_{i+1}) ≈ δ(k_i) in φ for 1 ≤ i < n,
        and δ(k_n) ≈ t or t ≈ δ(k_n) in φ,
      then return terminal(k, t, φ) := true,
      else return terminal(k, t, φ) := false;
end
```

FIGURE 3.15: PROCEDURE terminal

```
Input:  a K-minimal description φ
Output:  a description

begin
    φ' := φ;
L1: for all k ∈ node(φ):
      if there is a t ∈ T ∪ {ε} with δ(k) ≈ t in φ or t ≈ δ(k) in φ,
      then:  for all a ∈ A:
        if neither a(δ(k)) ≈ ⊥ nor ⊥ ≈ a(δ(k)) are in φ',
        then add a(δ(k)) ≈ ⊥ to φ';
L2: for all k1, k2 ∈ node(φ) with δ(k1) ≈ δ(k2) in φ:  begin
       for all a ∈ A such that a(δ(k1)) ≈ a(δ(k2)) is not in φ':
         add the conjunct a(δ(k1)) ≈ a(δ(k2)) to φ';
       remove δ(k1) ≈ δ(k2) from φ';
    end;
L3: for all k1, k2 ∈ node(φ) and a1, a2 ∈ A:
      if a1(δ(k1)) ≈ a2(δ(k2)) in φ' and a2(δ(k2)) ≈ a1(δ(k1))
         is no conjunct in φ',
      then add a2(δ(k2)) ≈ a1(δ(k1)) to φ';
L4: for all k1, k2 ∈ node(φ), a ∈ A and v ∈ V ∪ {⊥}:  if
      either a(δ(k1)) ≈ v or v ≈ a(δ(k1)) in φ,
      and either a(δ(k2)) ≈ v or v ≈ a(δ(k2)) in φ,
    then add a(δ(k1)) ≈ a(δ(k2)) and a(δ(k2)) ≈ a(δ(k1)) to φ';
L5: if there are k1, k2, k3 ∈ node(φ) and a1, a2, a3 ∈ A with
      a1(δ(k1)) ≈ a2(δ(k2)) and a2(δ(k2)) ≈ a3(δ(k3)) in φ'
      and a1(δ(k1)) ≈ a3(δ(k3)) not in φ',
    then begin
        add a1(δ(k1)) ≈ a3(δ(k3)) to φ';
        add a3(δ(k3)) ≈ a1(δ(k1)) to φ';
        go to L5;
    end;
    return attr-closure(φ) := φ';
end
```

FIGURE 3.16: PROCEDURE attr-closure

```
Input:  a name k, a K-minimal description φ, an a ∈ A
and a v ∈ V ∪ {⊥}
Output:  true or false

begin
    φ' := attr-closure(φ);
    if
       a(δ(k)) ≈ v or v ≈ a(δ(k)) in φ'
       or there is a a' ∈ A and a k' ∈ node(φ) with
          a(δ(k)) ≈ a'(δ(k')) in φ'
          and either a'(δ(k')) ≈ v or v ≈ a'(δ(k')) in φ',
    then return attr-value(k, a, v, φ) := true,
    else return attr-value(k, a, v, φ) := false;
end
```

FIGURE 3.17: PROCEDURE `attr-value`

equivalences $\delta(k_1) \approx \delta(k_2)$ are replaced by all conjuncts of the form $a'(\delta(k_1)) \approx a'(\delta(k_2))$ for all $a' \in A$. Then, in `L3`, all conjuncts of the form $a_1(\delta(k_1)) \approx a_2(\delta(k_2))$ for $k_1, k_2 \in node(\phi), a_1, a_2 \in A$ are added that follow from the symmetry of the equivalence relation. In `L4` equivalences between attributes with equal values are added, and in `L5` equivalences are added that follow from the transitivity of the equivalence relation.

Clearly, for $\phi' :=$`attr-closure(`ϕ`)`, $\phi \vdash \phi'$ and $\phi' \vdash \phi$ holds, and ϕ' is such that

- for all $k \in node(\phi)$, $a \in A$ and $v \in V \cup \{\bot\}$: $\phi \vdash a(\delta(k)) \approx v$ iff either $a(\delta(k)) \approx v$ or $v \approx a(\delta(k))$ in ϕ',
- and for all $a_1, a_2 \in A$ and $k_1, k_2 \in node(\phi)$: $\phi \vdash a_1(\delta(k_1)) \approx a_2(\delta(k_2))$ iff $a_1(\delta(k_1)) \approx a_2(\delta(k_2))$ is conjunct in ϕ'.

The procedure `attr-value` (Fig. 3.17) tests whether $\phi \vdash a(\delta(k)) \approx v$ holds for a K-minimal description ϕ, a certain $k \in node(\phi), a \in A$ and $v \in V \cup \{\bot\}$. With `attr-closure`, this test is quite simple.

For the test of axiom *(A5)* in the main derivation algorithm, the two descriptions ϕ_2 and ϕ_3 are already such that for each new node name equivalence $k \approx k_m$ with k_m marked or minimal in the new elementary description, all occurrences of k_m were replaced by k. Therefore the functions f_1 and f_2 in *(A5)* can be omitted (they are both the identity mapping). For two descriptions ϕ_1, ϕ_2, the procedure `more-underspecified` testing whether the first one implies the second or the reverse, is very simple (see Fig. 3.18). `more-underspecified(`ϕ_1, ϕ_2`) = true` (i.e. $\phi_2 \vdash \phi_1$) holds if all conjuncts in ϕ_1 are implied by ϕ_2.

Clearly, for two K-minimal descriptions ϕ_1, ϕ_2: `more-underspecified(`ϕ_1, ϕ_2`) = true` iff $\phi_2 \vdash \phi_1$.

The main derivation procedure `derivation` in Fig. 3.19 describes the construction of a set $D^{\psi}_{\phi_1}$ for a given ϕ_1 and a given ψ as described above. For this set $D^{\psi}_{\phi_1}$, it will be shown

```
Input:  two K-minimal (A,V,T)-descriptions φ1 and φ2
Output:  true or false

begin
    If for all k1,k2 ∈ node(φ1) with k1 ≠ k2, a ∈ A, v ∈ V ∪ {⊥}, t ∈
    T ∪ {ε}:
      1. if k1 ◁ k2 in φ1, then k1 ◁ k2 in φ2,
      2. if k1 ≺ k2 in φ1, then left-of(k1, k2, φ2) = true,
      3. if k1 ◁* k2 in φ1, then dom(k1, k2, φ2) = true,
      4. if δ(k1) ≈ t or t ≈ δ(k1) in φ1, then terminal(k1,t,φ2) =
         true
      5. if δ(k1) ≈ δ(k2) in φ1,
         then
            either δ(k1) ≈ δ(k2) or δ(k2) ≈ δ(k1) in φ2
            or there is a t ∈ T ∪ {ε} with
               terminal(k1,t,φ2) = terminal(k2,t,φ2) = true
            or
               there are a' ∈ A and v' ∈ V with
                     attr-value(k1,a',v',φ2) = true,
               and for all a'' ∈ A:
                     a''(δ(k1)) ≈ a''(δ(k2)) in attr-closure(φ)
      6. if a(δ(k1)) ≈ v or v ≈ a(δ(k1)) in φ1,
         then attr-value(k1,a,v,φ2) = true
    then more-underspecified(φ1, φ2) = true,
    else more-underspecified(φ1, φ2) = false;
end
```

FIGURE 3.18: PROCEDURE more-underspecified

that $\phi_1 \overset{\psi}{\Rightarrow} \phi_2$ iff there is a $\phi_2' \in D_{\phi_1}^{\psi}$ such that the following holds: there is a mapping $f : node(\phi_2) \to node(\phi_2')$ mapping all marked or minimal node names in ψ to their equivalent name in ϕ_1 and all other names to themselves. For this mapping, $\hat{f}(\phi_2)$ is equivalent to ϕ_2' (i.e. $\hat{f}(\phi_2) \vdash \phi_2'$ and $\phi_2' \vdash \hat{f}(\phi_2)$ hold).

Starting from the conjunction of ϕ_1 and ψ (see `L1`), for all marked or minimal names in ψ, equivalent names in ϕ_1 must be found (see `L5`) such that axiom *(A2)* and *(A3)* are respected (see `L8` and `L9`). Since the result must be a description, further dominances must be addded such that **(D)** holds, and the result must be satisfiable (see `L10`). Finally it must be checked whether there is no other possibility for the new equivalences that is more underspecified (see `L3`).

In this procedure, `begin`s and `end`s are equipped with numbers in order to show the structure of the procedure. A pair of `begin` and `end` corresponding to each other have the same number.

The following proposition states that the result of this procedure represents really the set of all descriptions that might be derived from ϕ_1 by adding ψ.

Proposition 3.8 (Derivation procedure) *Let $G = \langle A, V, T, D, \phi_S \rangle$ be a TDG. Let ϕ_1 be an (A, V, T)-description and let $\langle \psi, k_\psi \rangle$ be K-equivalent to an elementary description in G such that $node(\phi_1) \cap node(\psi) = \emptyset$ and ϕ_1 and ψ are both K-minimal.*
$\phi_1 \overset{\psi}{\Rightarrow} \phi_2$ iff there is a $\phi_2' \in$ `derivation`(ϕ_1, ψ) such that there is a mapping $f : node(\phi_2) \to node(\phi_2')$ with

1. *$f(k) := k$ for all $k \in node(\phi_2) \setminus M_\psi$,*
2. *for all $k \in M_\psi$ with $\phi_2 \vdash k \approx k'$ for $k' \in node(\phi_1)$: $f(k) := k'$,*
3. *and $\hat{f}(\phi_2) \vdash \phi_2'$ and $\phi_2' \vdash \hat{f}(\phi_2)$.*

Proof Let $G = \langle A, V, T, D, \phi_S \rangle$ be a TDG. Let ϕ_1 and ψ be as specified for the input of `derivation`.
To show: $\phi_2' \in$ `derivation`$(\phi_1,\ \psi)$ iff there is a ϕ_2 with $\phi_1 \overset{\psi}{\Rightarrow} \phi_2$ and with a mapping f as in Prop. 3.8.

The following holds: $\phi_1 \overset{\psi}{\Rightarrow} \phi_2$ iff ϕ_2 is a description satisfying **(L)**, **(LP)**, **(D)**, and *(A1)* to *(A5)*.

To show: $\phi_2' \in D_{\phi_1}^{\psi}$ holds in `L3` iff the corresponding ϕ_2 is a description satisfying *(A1)* to *(A4)* and **(L)**, **(LP)**, and **(D)**.

$\phi_2' \in D_{\phi_1}^{\psi}$ holds in `L3` iff for ϕ_2' and a corresponding ϕ_2 that can be obtained by adding all new equivalences as conjuncts to ϕ_2'

- ϕ_2' (and then also ϕ_2) is a description,
- ϕ_2 satisfies *(A1)* because of `L1` and because none of the conjuncts was removed in the course of the construction of $D_{\phi_1}^{\psi}$,
- ϕ_2 satisfies *(A2)* because
  (a) with the condition in `L2` and with `L5`, for all $k_m \in M_\psi$ there is exactly one $k \in node(\phi_1)$ with $\phi_2 \vdash k_m \approx k$, and

```
Input:  a description φ1 and a pair ⟨ψ, Kψ⟩ such that Kψ ⊆ node(ψ),
node(φ1) ∩ node(ψ) = ∅ and φ1 and ψ are both K-minimal
Output:  a set of descriptions

begin(1)
    Mψ := {k; k minimal in ψ or k ∈ Kψ};
L1: D^ψ_φ1 := {φ1 ∧ ψ};
L2: if Mψ = ∅, then begin(2)
L3:     for all φ, φ' ∈ D^ψ_φ1:  if more-underspecified(φ, φ') = true
              and more-underspecified(φ', φ) = false,
           then remove φ' from D^ψ_φ1;
L4:     return derivation(φ1, ψ) := D^ψ_φ1;
    end(2),
    else begin(3)
L5:     select an (arbitrary) km ∈ Mψ and remove km from Mψ;
        Dnew := ∅;
L6:     for all φold ∈ D^ψ_φ1:
L7:         for all k ∈ node(φ1):  begin(4)
              φ := φold;
              if there are no kp, k'p such that
L8:             either kp ◁ k in φ and k'p ◁ km in φ,
L9:             or k ◁ kp in φ and km ◁ k'p in φ,
              then begin(5)
                replace all occurrences of km in φ by k;
L10:            if there are kd, ks with k ◁ kd in φ and
                  strong-dom(k, ks, φ) = true,
                then:  for all k'd such that dom(kd, k'd, φ) = true and
                  there is no kp with k'd ◁ kp in φ:
                  begin(6)
                    add k'd ◁* ks to φ;
                    if satisfiable(φ) = true, then add φ to Dnew;
                  end(6);
              end(5);
            end(4);
L11:    D^ψ_φ1 := Dnew;
L12:    go to L2;
    end(3);
end(1)
```

FIGURE 3.19: PROCEDURE derivation

(b) because of the condition in L8, for no $k_\psi \in node(\psi) \setminus M_\psi$, there is a $k \in node(\phi_1)$ with $\phi_2 \vdash k_\psi \approx k$:
if such a k existed, then the equivalence would have been added by one of the inference rules $tr_=, sy_=, rf_=, antis_\mathcal{D}$ or T_5. The first four rules cannot add equivalences because ϕ_1 and ψ are K-minimal, and T_5 cannot add equivalences because of the condition in L8.

- ϕ_2 satisfies *(A3)* because of the condition in L9,
- ϕ_2 satisfies *(A4)* because no parent relations were added,
- ϕ_2 satisfies **(L)**, **(LP)**, and **(D)** since only dominances were added in L10 according to **(D)**.

Consequently, the following holds:
$\phi_2' \in$ `derivation`(ϕ_1, ψ)
iff ϕ_2 is a description satisfying *(A1)* to *(A5)* with respect to ϕ_1 and ψ (and consequently also satisfying **(L)**, **(LP)**, and **(D)**)
iff $\phi_1 \overset{\psi}{\Rightarrow} \phi_2$.

□

3.3.3 Decidability of lexicalized TDGs

In order to deal with natural languages in a way similar to TAGs, I will now define *lexicalized TDGs* (for a definition of lexicalized grammars in general see Schabes 1990). A TDG is called lexicalized if there is at least one leaf name with a terminal label in each elementary description. Furthermore, this terminal is preceded by a kind of "preterminal", i.e. for the leaf name with the terminal label, there must be another name describing the parent of this leaf.

Definition 3.19 (Lexicalized TDG) *A TDG G is a lexicalized TDG (LTDG), iff for every elementary description ψ in G there are $k, k_t \in node(\psi)$ and there is a $t \in T$, such that $\psi \vdash k \lhd k_t \wedge \delta(k_t) \approx t$.*

Note that, as in the case of TAGs, this definition only captures the requirement to have lexical items in each elementary description, but it does not say anything about the uniqueness of an anchor.

To show the decidability of the word problem for lexicalized TDGs, the decidability of minimal trees first must be proven. The result shown for the derivation algorithm then tells us that the language generated by a lexicalized TDG is decidable.

In order to find the set of minimal trees for a given description, it is sufficient to add equivalences between n-terms, l-terms or v-terms to ϕ. Therefore, the set of minimal trees of a given description is decidable:

Proposition 3.9 (Decidability of minimal trees) *Let G be a TDG. For each $\phi \in L_D(G)$ the set of minimal trees of ϕ can be constructed in finitely many steps.*

More precisely representatives of classes of isomorphic minimal trees will be constructed (see Def. 2.15 for the notion of a tree isomorphism, only the constraint functions C and O are omitted in the case of minimal trees). Henceforward I will not distinguish between isomorphic trees. A tree will be identified with the equivalence class it represents.

As already mentioned, the idea of the proof of Prop. 3.9 is to construct for a derived description ϕ first some description ϕ' by adding equivalences such that there are no strong dominances in ϕ' and all node labels are completely specified in ϕ'. Then there is a direct correspondence between these descriptions ϕ' and the minimal trees of ϕ.

Proof or Prop. 3.9: Let G be a TDG and $\phi \in L_D(G)$.
In order to find a minimal tree of ϕ, a description ϕ' must be constructed such that $\phi' \vdash \phi$, $node(\phi) = node(\phi')$ and for all $k_1, k_2 \in node(\phi)$:

(a) $\phi' \vdash_s k_1 \lhd^* k_2$ does not hold.

(b) $\phi' \vdash k_1 \approx k_2$ or $\phi' \vdash \neg k_1 \approx k_2$.

(c) $\phi' \vdash k_1 \lhd k_2 \Rightarrow$ there are k_3, k_4 such that $\phi' \vdash k_1 \approx k_3 \wedge k_2 \approx k_4$ and $\phi \vdash k_3 \lhd k_4$ (no new parent relations, see second axiom for minimal trees).

(d) If $\phi' \vdash k_1 \approx k_2$ and $\phi \nvdash k_1 \approx k_2$, then there are no k_3, k_4 such that $\phi \vdash k_1 \lhd k_3 \wedge k_2 \lhd k_4$ (see third axiom for minimal trees).

(e) For all $k \in node(\phi')$:
if k is a leaf name in ϕ, then there is a $t \in T \cup \{\epsilon\}$ with $\phi' \vdash \delta(k) \approx t$ (tree must be terminal),
else (k no leaf name) for all $a \in A$ there is a $v \in V \cup \{\bot\}$ with $\phi' \vdash a(\delta(k)) \approx v$, and there is at least one $\langle a, v\rangle \in A \times V$ with $\phi' \vdash a(\delta(k)) \approx v$ (at least one attribute with a value $v \neq \bot$).

Construction of such a ϕ':

- $\phi' := \phi$
- Add arbitrary conjuncts $k_1 \approx k_2$ to ϕ' with $k_1 \neq k_2$ and $k_1, k_2 \in node(\phi)$.
- Add arbitrary conjuncts $\delta(k) \approx t$ to ϕ' with $k \in node(\phi)$ and $t \in T \cup \{\epsilon\}$.
- For all $k \in node(\phi)$ where no formula $\delta(k) \approx t$ was added and for all $a \in A$, choose a $v \in V \cup \{\bot\}$ and add the conjunct $a(\delta(k)) \approx v$ to ϕ'.
- If the result ϕ' is a description such that conditions (a) - (e) hold (this is decidable), then it is one of the desired descriptions.

The construction is non-deterministic. But as $node(\phi)$, A, V and T are finite and "$\vdash$" is decidable, there are only finitely many possibilities to construct a ϕ' in this way. Consequently all possible ϕ' can be found in finitely many steps.

Now it must be shown that the resulting ϕ' correspond to the minimal trees:

(i) Each of the ϕ' uniquely describes a minimal tree $B_{\phi'}$ of ϕ with respect to nonterminals NA, V and terminals T:
$B_{\phi'} = (\mathcal{U}_{\phi'}, u_{0\,\phi'}, \mathcal{P}_{\phi'}, \mathcal{D}_{\phi'}, \mathcal{L}_{\phi'}, \mu)$ such that
- $\mathcal{U}_{\phi'} = \{u \mid u \subset node(\phi') \text{ is equivalence class wrt } \{\langle k_1, k_2\rangle \mid \phi' \vdash k_1 \approx k_2\}\}$
- $u_{0\phi'}$ is the node with $k \in u_{0\phi'}$ iff k is minimal in ϕ'.
- $\mathcal{P}_{\phi'} = \{\langle u_1, u_2\rangle \mid \phi' \vdash k_1 \lhd k_2 \text{ for all } k_1 \in u_1 \text{ and } k_2 \in u_2\}$

- $\mathcal{D}_{\phi'} = \{\langle u_1, u_2\rangle \mid \phi' \vdash k_1 \lhd^* k_2 \text{ for all } k_1 \in u_1 \text{ and } k_1 \in u_2\}$
- $\mathcal{L}_{\phi'} = \{\langle u_1, u_2\rangle \mid \phi' \vdash k_1 \prec k_2 \text{ for all } k_1 \in u_1 \text{ and } k_2 \in u_2\}$
- For all $u \in \mathcal{U}_{\phi'}$ that are no leaf in $B_{\phi'}$:
  $\mu(u) := \{\langle a, v\rangle \mid a \in A, v \in V \text{ and } \phi' \vdash a(\delta(k)) \approx v \text{ for all } k \in u\}$.
- For all leaves $u \in \mathcal{U}_{\phi'}$ and all $t \in T \cup \{\epsilon\}$:
  $\mu(u) := t$ iff $\phi' \vdash \delta(k) \approx t$ for all $k \in u$.

Because of the inference rules, it is easy to see that $B_{\phi'}$ is a tree.

To show: $B_{\phi'}$ is a minimal tree of ϕ:

1. $B \models_g \phi$ holds for all assignments g such that for each $u \in \mathcal{U}_{\phi'}$: $g(k) = u$ for all $k \in u$.
2. With the definition of $\mathcal{P}_{\phi'}$, for all $\langle u_1, u_2\rangle \in \mathcal{P}_{\phi'}$ there are $k_1, k_2 \in node(\phi)$ such that $\phi' \vdash k_1 \lhd k_2$ and $g(k_1) = u_1, g(k_2) = u_2$.
   Consequently (with condition (c) for ϕ') there are $k_3, k_4 \in node(\phi)$ such that $\phi \vdash k_3 \lhd k_4$ and $\phi' \vdash k_1 \approx k_3 \wedge k_2 \approx k_4$
   Consequently (with definition of $\mathcal{U}_{\phi'}$) there are $k_3, k_4 \in node(\phi)$ such that $\phi \vdash k_3 \lhd k_4$ and $g(k_3) = u_1, g(k_4) = u_2$.
3. Let $g(k_1) = g(k_2)$ and $\phi \nvdash k_1 \approx k_2$ hold.
   Consequently $\phi' \vdash k_1 \approx k_2$ and $\phi \nvdash k_1 \approx k_2$
   Consequently (with cond. (d)) there are no k_3, k_4 with $\phi \vdash k_1 \lhd k_3 \wedge k_2 \lhd k_4$.

(ii) For each minimal tree B of ϕ there is a ϕ' that can be constructed from ϕ such that B is isomorphic to $B_{\phi'}$:
Let B be a minimal tree of $\phi \in L_D(G)$ wrt the node assignment g.
For all $k, k_1, k_2 \in node(\phi)$:

1. if $k_1 \neq k_2$ and $g(k_1) = g(k_2)$, then add $k_1 \approx k_2$ as a conjunct to ϕ,
2. if there is a $t \in T \cup \{\epsilon\}$ such that $\mu(g(k)) = t$, then add $\delta(k) \approx t$ as a conjunct to ϕ, and
3. if $g(k)$ is no leaf, then for all $a \in A$: if there is a $v \in V$ with $\langle a, v\rangle \in \mu(g(k))$, then add the conjunct $a(\delta(k)) \approx v$ to ϕ, else add the conjunct $a(\delta(k)) \approx \bot$.

The resulting description ϕ' is one of the descriptions, that can be constructed from ϕ in the way described above.
Obviously the corresponding tree $B_{\phi'}$ is isomorphic to B.

□

With Prop. 3.9 and 3.8 the following holds for LTDGs:

Proposition 3.10 *(Decidability)*
For LTDGs the word problem is decidable.

Proof: With Prop. 3.8, for a given description ϕ_1 and an elementary ψ the set
$\{\phi \mid$ there is a ϕ_2 with $\phi_1 \overset{\psi}{\Rightarrow} \phi_2$ and a $f : node(\phi_2) \to node(\phi)$
with $\hat{f}(\phi_2) = \phi$ and $\phi_2 \vdash k_1 \approx k_2$ for all k_1, k_2 with $f(k_1) = f(k_2)\}$

is finite and can be constructed in finitely many steps (i.e. is decidable).
Since G is lexicalized, in each derivation step at least one terminal is added to the yield of a possible minimal tree. Therefore, in order to decide for a given $w = t_1 \cdots t_n$, $t_i \in T$, whether $w \in L_S(G)$ holds, the following is sufficient: all derivations $\phi_S \stackrel{*}{\Rightarrow} \phi$ must be performed where for each of the t_i exactly one elementary description ψ with $\psi \vdash \delta(k) \approx t_i$ for one node name k is added. In this way in finitely many steps a finite set of descriptions is obtained. After that the minimal trees of these descriptions must be constructed (decidable with Prop. 3.9). If $w = yield(B)$ holds for one of these trees B, then $w \in L(G)$.

□

3.4 TDGs for natural language

In this section TDGs, more precisely LTDGs will be considered with respect to natural languages, and they will be compared to TAG variants mentioned in Chapter 2, in particular to MC-TAGs, D-Tree Grammars and Unordered Vector Grammars with Dominance Links. We will see that TDGs allow "multicomponent" derivation because one elementary description can describe several subtrees that have to be added at the same time. TDGs are even powerful enough to generate the indexed string language SCR^{ind} which is not the case for tree-local MC-TAGs. As shown by Becker et al. (1992), a grammar that is adequate to handle scrambling phenomena in German must be able to generate SCR^{ind} (see also Section 2.4.1 in Chapter 2).

Furthermore, with TDGs arguments can be treated in a uniform way because subsertion-like derivation steps are possible. In the previous chapter, we have seen that this is one of the problems for TAGs where derivation is restricted to substitution and adjunction and therefore a uniform treatment of arguments is not possible.

With respect to the main purpose of the introduction of TDGs, the generation of underspecified representations, we will see that, since underspecification of the dominance relation is possible in TDGs, representations as the one in the introductory example in Fig. 1.2 on page 6 and Fig. 2.22 on page 51 for nominal modifier ambiguities can be derived.

3.4.1 Multicomponent derivation

Kroch (1987) and Kroch and Joshi (1987) have argued that the derivational generative capacity of TAGs is not sufficient to analyze certain nonlocal dependencies, such as extraction out of noun phrases as in (11), in an appropriate way. They propose the use of tree-local multicomponent TAGs for these cases. An MC-TAG analysis of (11) was given in Chapter 2.

(11) which painting$_i$ did you buy a copy of ϵ_i?

Vijay-Shanker (1992) already points out that in a grammar based on quasi-trees such multicomponent derivations are possible because more than one subtree can be represented in one tree description. This is the case in particular for TDGs, a TDG analysis of (11) is shown in Fig. 3.20. The corresponding analysis with tree-local MC-TAGS has been shown in Fig. 2.10 on page 35 in Chapter 2. In the TDG, the two trees β_1 and β_2 of the auxiliary set are described in a single elementary description ψ where the two node names denoting the

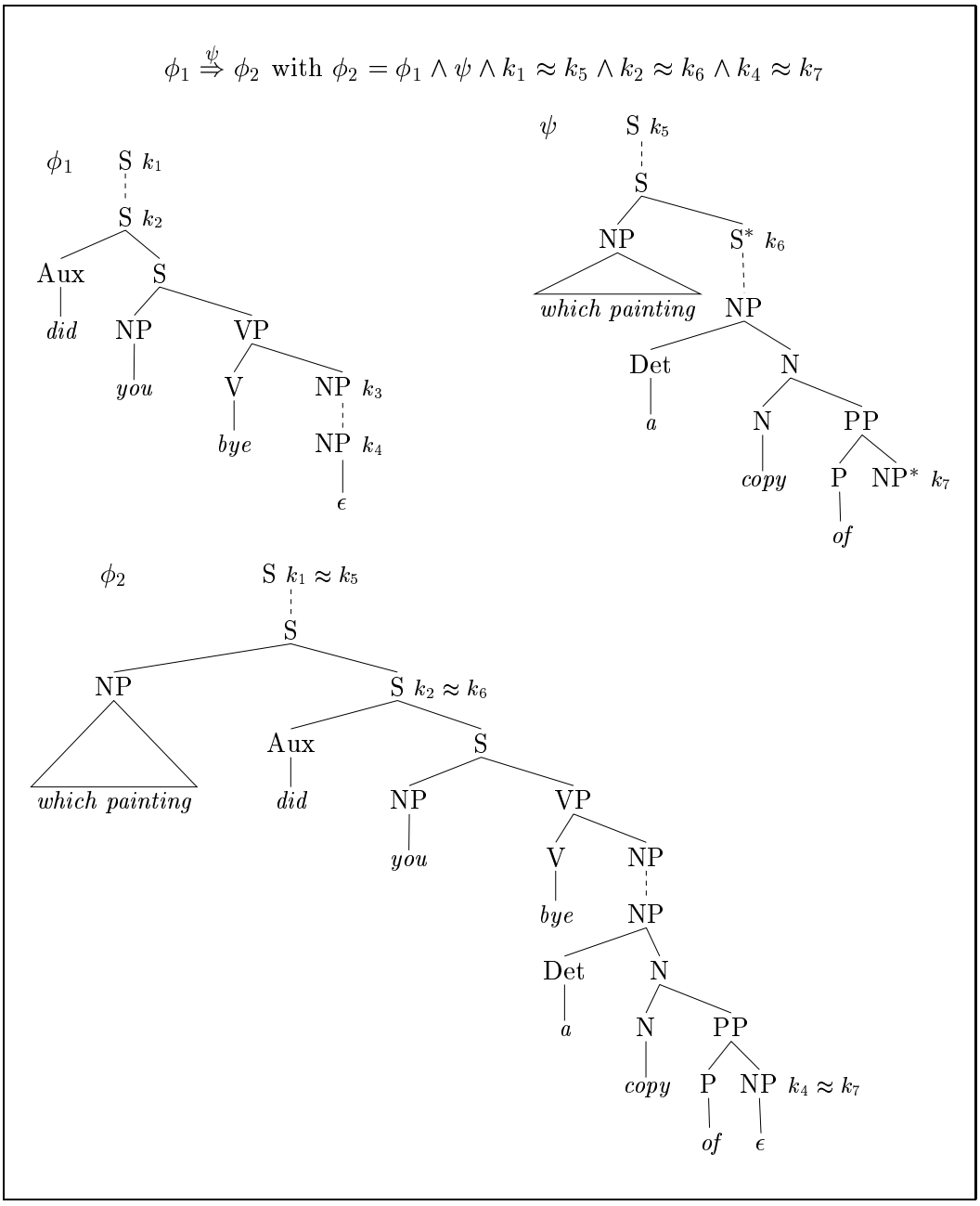


FIGURE 3.20: A MULTICOMPONENT DERIVATION IN A TDG

foot nodes of β_1 and β_2 are marked. However, as already mentioned for TAGs and TDGs, this way of constructing a corresponding TDG for a tree-local MC-TAG does not work in general, since TDGs are not local.

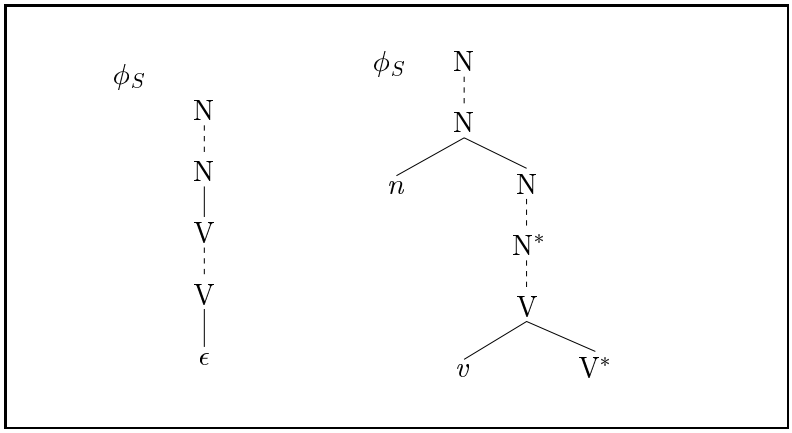


FIGURE 3.21: A TDG GENERATING SCR^{ind}

A TDG generating the indexed string language SCR^{ind} is shown in Fig. 3.21. For each $m \leq 0$, there is only one description in the description language of this grammar, and this description is an underspecified representation of minimal trees for all $\{n^{[\sigma(1)]} \dots n^{[\sigma(m)]} v^{[1]} \dots v^{[m]} \mid \sigma \in S_m\}$. As shown by Becker et al. (1992), the indexed language cannot be generated by set-local MC-TAGs. This shows that the derivational generative capacity of set-local MC-TAGs is not greater than the derivational generative capacity of TDGs.

A variant of MC-TAGs are Unordered Vector Grammar with Dominance Links (UVG-DL), proposed by Rambow (1994a,b). I have introduced these grammars in the last chapter. In his dissertation, Rambow argues that UVG-DLs are adequate to handle scrambling in German.

When comparing TDGs to UVG-DLs, it is very easy to construct a TDG G for a given UVG-DL G_V such that the two grammars are equivalent with respect to their weak, strong and even derivational generative capacity. The construction is very simple: for each set (vector) of productions in G_V, one elementary description is constructed with no marked names, one minimal name labelled with the start symbol of G_V, one subtree description for each production, dominance links from the minimal name to the minimal names of the subtree descriptions of the single productions and the dominance links associated in G_V with this production vector. The start description contains only one node name labelled by the start symbol of G_V. Therefore, clearly the following holds:

Proposition 3.11 *Each language generated by an Unordered Vector Grammar with Dominance Links is a TDL.*

For the UVG-DL in Fig. 2.19 on page 47, an equivalent TDG is shown in Fig. 3.22. The elementary description $\psi(v_i)$ is constructed for the production vector v_i. In the TDG, the

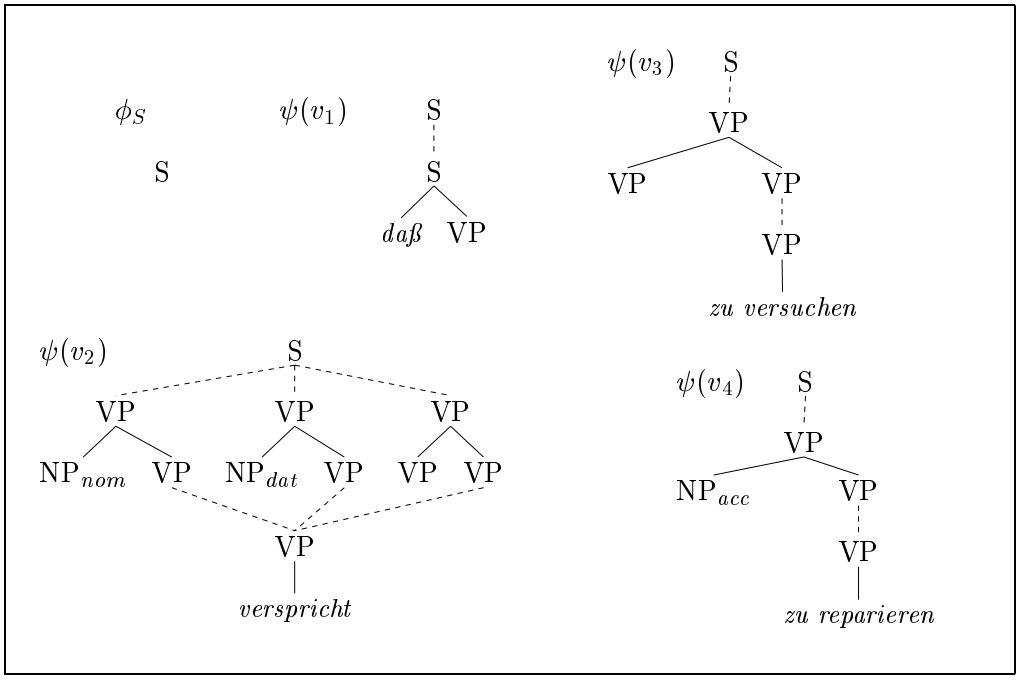


FIGURE 3.22: A TDG FOR THE UVG-DL IN FIG. 2.19, P. 47

application of the productions in the vectors does not take place during the derivation of a description but only when a minimal tree is found. Then node name equivalences must be added and this corresponds to the applications of the productions in G_V.

3.4.2 Complement clauses

As pointed out in the previous chapter, one of the deficiencies of TAGs is that the TAG operations of substitution and adjunction do not map cleanly onto the relations of complementation and modification. Complement clauses must be treated in a different way than other arguments (see Abeillé 1988b, 1993 and Rambow et al. 1995). One of the main ideas in TAG is that arguments should be added by substitution to the argument structure whereas adjuncts are added by adjunction. The only exception are complement clauses. The reason for this difference is the desire to describe argument structures locally. In TAGs all argument nodes for one lexical head are in one elementary tree. This is captured by the predicate argument cooccurrence principle. Even for unbounded dependencies the argument that has been moved has to be in the same elementary tree as the head of the argument structure. If one wants to preserve this in a TAG, it is not possible to add complement clauses by substitution but the argument structure must be adjoined to the complement clause (see Fig. 2.9).

With TDGs it is possible to have this local analysis of unbounded dependencies without treating complement clauses in a way different from other arguments. Even for unbounded dependencies inside of complement clauses the derivation can start from the argument structure and the complement clause is added in one derivation step. The crucial point here is that there can be arbitrarily many marked names in elementary descriptions. A sample

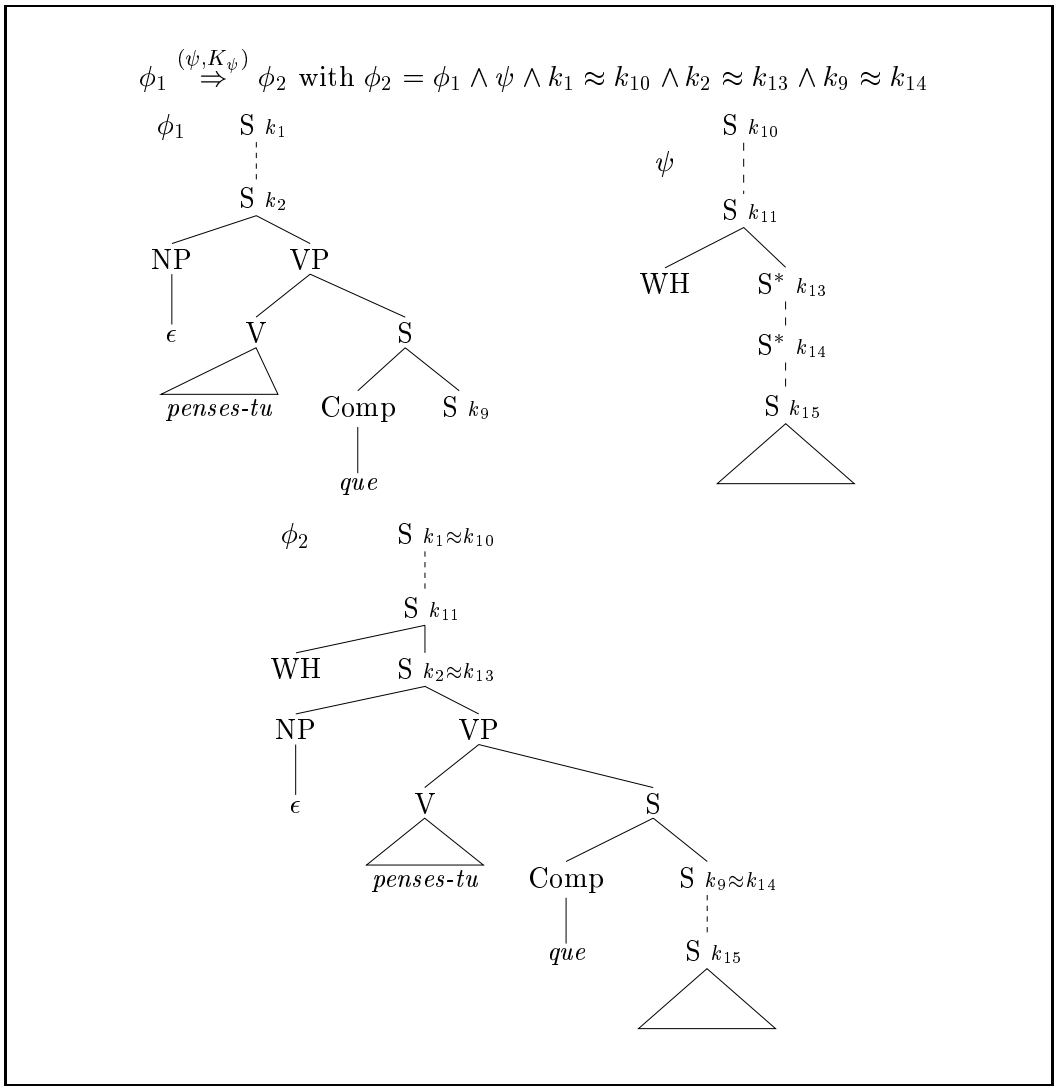


FIGURE 3.23: COMPLEMENT CLAUSES WITH UNBOUNDED DEPENDENCIES *qui penses-tu que Marie aime* ϵ (SEE ABEILLÉ 1988B)

analysis for (12) is given in Fig. 3.23.

(12) qui penses-tu que Marie aime ϵ
whom do you think Marie loves ϵ

This treatment of complement clauses corresponds to the *subsertion* operation proposed in Rambow et al. 1995 as derivation operation for Description-Tree-Grammars (DTG, see Section 2.5.2). Subsertion is a non-local operation in the sense that in one subsertion derivation step the d-edges chosen for the insertion of new subtrees need not be part of one single elementary d-tree. Therefore it seems to be no problem to show that each language generated by a DTG can also be generated by a TDG, i.e. that TDGs are more powerful than DTGs.

In TDGs the elementary description for arguments and those for modifiers differ in the following way: arguments are described by descriptions ψ such that no leaf name is marked in ψ. In contrast to this the elementary description of a modifier contains a marked leaf name, that can be compared to a foot node of an auxiliary tree in a TAG.

3.4.3 Underspecification

As already mentioned in the introduction and the previous chapter, my main motivation for defining TDGs was the desire to generate underspecified representations of trees. In this section I will consider some examples of structural ambiguities and I will show that TDGs enable us to describe several readings in one tree description.

First we will go back to the example of a syntactic ambiguity mentioned in Chapter 2, the nominal modifier scope ambiguity *a former professor in Tübingen*. As already explained, corresponding to the two syntactic structures (see Fig. 2.22) this expression has two different interpretations, either with wide scope of *former* or with wide scope of *in Tübingen*. The elementary descriptions for *former, in* and *professor* can be chosen in such a way that one single underspecified description is derived for the ambiguous expression. Fig. 3.24 shows the elementary descriptions ϕ_1, ψ_1, ψ_2 and ψ_3 for *a*, *professor*, *former* and *in* and the derived descriptions that are obtained when, starting from the description for *a*, the other three elementary descriptions are added. In the figure, the description of *former* is added before adding the description for the preposition. However, the other order, i.e. first adding ψ_3 and then ψ_2 leads to the same result. The resulting description ϕ_4 is not unique since for k_7 one of the two names k_3, k_5 may be chosen as equivalent name, and for k_8 either k_4 or k_6 may be chosen. But the four possible results of putting ϕ_3 and ψ_3 together are equivalent to each other. ϕ_4 is (up to equivalence) the only possible result because of the derivation axiom *(A5)*.

The choice of the elementary descriptions in Fig. 3.24 corresponds to the way elementary trees in TAGs look like. The linguistic principles underlying TAGs, in particular the predicate-argument cooccurrence principle are respected.

In a very similar way underspecified representations for PP-attachment ambiguities can be obtained. Here not only the dominance relation but also the labeling function is not fully specified. Fig. 3.25 shows a part of the analysis of the sentence (13).

(13) Jean regarde un homme avec un télescope
Jean is looking at a man with a telescope

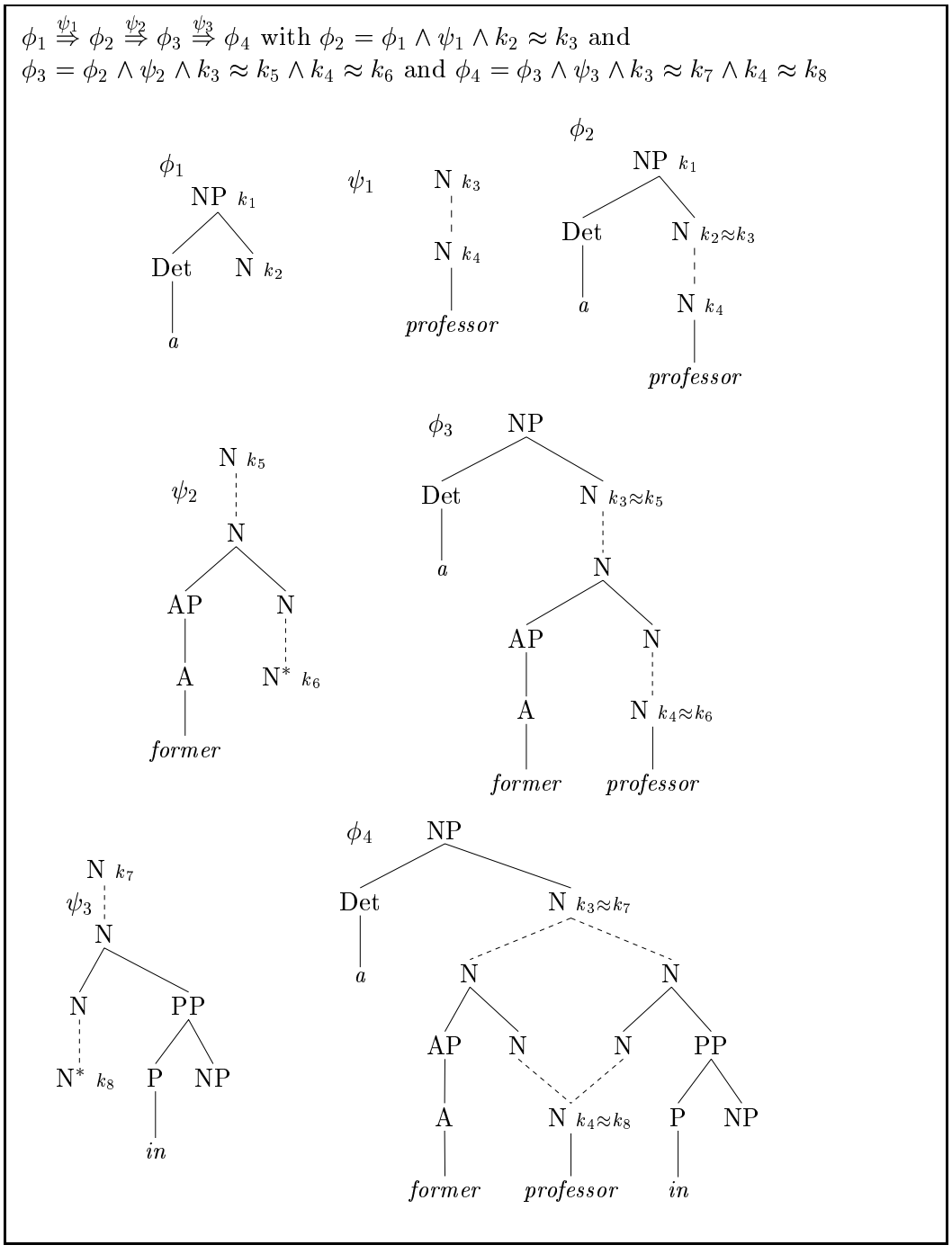


FIGURE 3.24: NOMINAL MODIFIERS: *a former professor in*

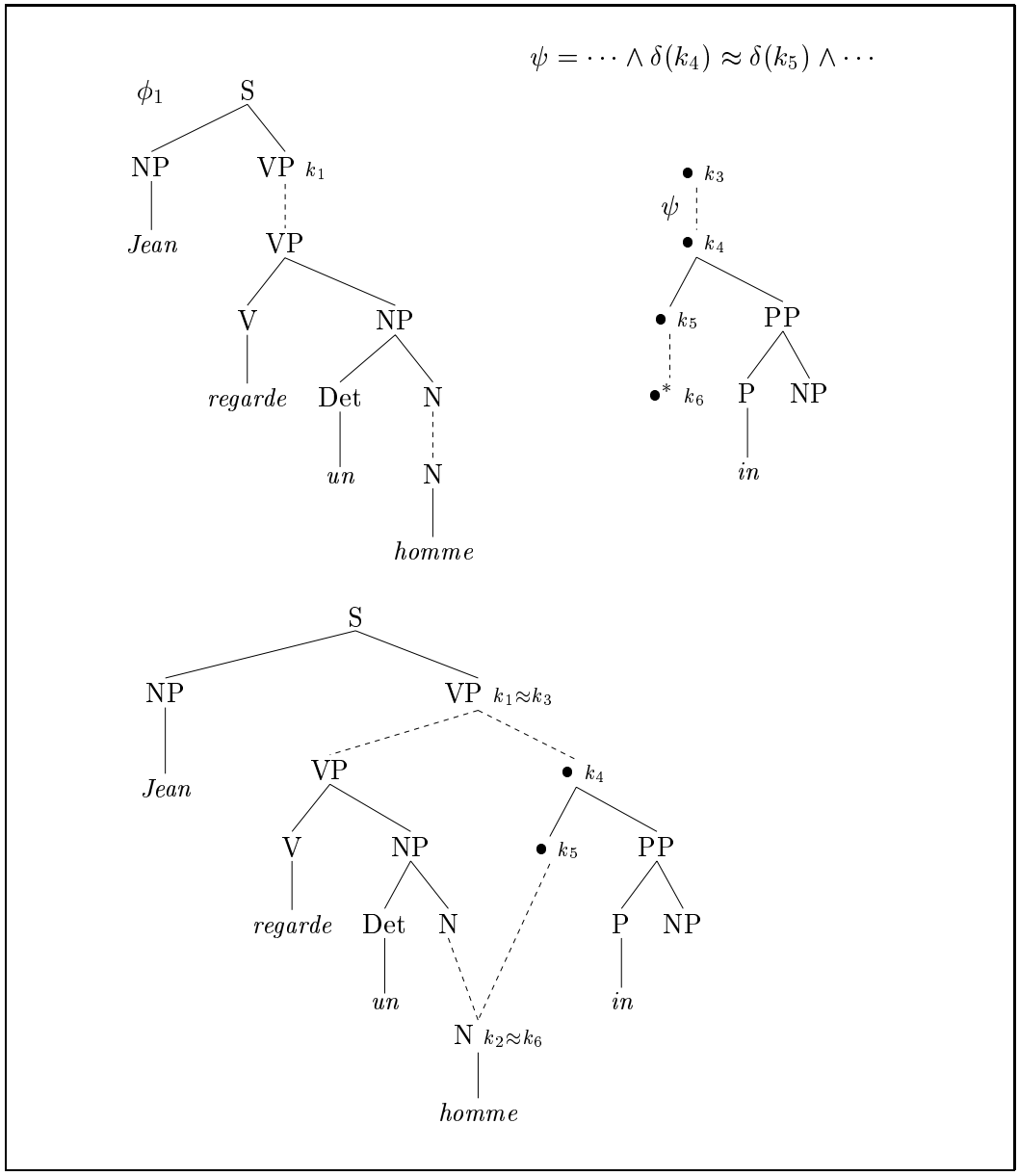


FIGURE 3.25: PP-ATTACHMENT: *Jean regarde un homme avec un télescope*

Other phenomena that might be handled with TDGs are ambiguities that arise with respect to temporal relations in narrative texts. Schilder (1997a,b) proposes a TDG analysis for the description of temporal relations. With this approach, underspecified representations can be obtained for narrative texts that are ambiguous with respect to their temporal structure.

In Chapter 5, when a syntax-semantics interface with synchronous local TDGs is proposed, I will also consider quantifier scope ambiguities, and we will see that for these ambiguities, it is possible to derive underspecified representations similar to those in Muskens 1995.

3.4.4 Generative capacity

The examples given in this section and the comparisons between TDGs on the one hand and MC-TAGs, UVG-DLs and DTGs on the other hand have shown that TDGs not only provide a description-based definition of TAG-like grammars. They also offer the advantages of multicomponent derivations and of a uniform distinction between complementation and modification operations. Furthermore they allow underspecification in a very natural way.

As pointed out in the beginning of Chapter 2, with respect to the generative capacity of a formalism developed for natural languages, there are two aspects that must be considered. The first concerns the question whether the formalism is powerful enough to handle natural languages in an adequate way. Since TDGs are at least as powerful as UVG-DLs and as DTGs, they probably satisfy this criterion. A second point is the question whether the formalism is not unnecessarily powerful, and this might be a problem for TDGs, since I have not shown any restriction of the generative power of TDGs, i.e. any class of languages that is a superset of the languages generated by TDGs.

One of the properties natural languages are supposed to have, is the so-called constant growth property. This property is one of the conditions for mildly context-sensitive grammars. Roughly said, a language satisfies the constant growth property if there is some constant c for this language such that for each word w there is either no word longer than w or there is a word with a length between $|w|$ and $|w| + c$.

Provided that natural languages are of constant growth, a formalism that is supposed to capture the complexity of natural languages must generate only languages satisfying the constant growth property. For TDGs as defined in this chapter, it might be at least difficult, probably even impossible to show constant growth of the generated string languages.

This indicates that TDGs in their general form might be unnecessarily powerful to deal with natural languages. To avoid this problem, a local version of TDGs will be defined in the next chapter, the *local* TDGs. We will see that local TDGs still offer the advantages of TDGs in the general form presented in this chapter. They are more powerful than set-local MC-TAGs, but they are restricted in such a way that their string languages are of constant growth.

Chapter 4

Local Tree Description Grammars

In the previous chapter I have introduced Tree Description Grammars, a description-based variant of TAGs. We have seen that TDGs allow subsertion-like dervation steps, "multi-component" derivation and underspecification. With respect to the languages generated by TDGs, we have seen that $\{a_1^k a_2^k ... a_n^k \mid 0 \leq k\}$ for $n \geq 5$ and $\{w^n \mid w \in T^*\}$ for $n \geq 3$ are TDLs, and that even all languages generated by Unordered Vector Grammars with Dominance Links (UVG-DL, Rambow 1994a,b) or by D-Tree-Grammars (DTG, Rambow et al. 1995) are TDLs. For these reasons TDGs are a very interesting variant of TAGs for natural languages.

As already pointed out in the end of the last chapter, the problem with TDGs is that they might be unnecessarily powerful. I have not found any lowest upper bound for the class of all TDLs. Since natural languages are supposed to be of constant growth, and constant growth is a consequence of semilinearity, it is desirable to define a grammar formalism that still has all the advantages of TDGs for natural languages as shown in the examples in Chapter 3 but can be shown to be semilinear. For TDGs in the general form this might be difficult. It is at least not obvious whether they are semilinear or not. The problem is the non-locality of the derivations in TDGs. In a TDG derivation step, for a minimal or marked name in the new elementary description, any name in the old description can be chosen as equivalent name. The names in the old description occurring in the equivalences added in this derivation step must not belong to one single elementary description. This is sketched in Fig. 4.1 where a possible non-local derivation step in a TDG is illustrated. The old description in this step has been derived from the four start or elementary descriptions $\psi_1, \ldots, \psi_4$. In the derivation step three equivalent names for the minimal name and the two marked names in the new elementary description depicted on the right must be found. As shown in the figure, it is possible to choose names from ψ_1, ψ_2 and ψ_4 for example, i.e. names form three different start or elementary descriptions added before. In other words, the subtree descriptions occurring in the new elementary description can be inserted into any of the strong dominances in the old description. Therefore the modification of the old description caused by this derivation step is non-local.

This non-locality not only is a problem because of the probable non-semilinearity of the formalism. Furthermore, also from a more linguistic point of view one does not want to allow non-locality as illustrated in Fig. 4.1. The parts of one local argument structure might be separated from each other by intervening material, but even in the case of non-local

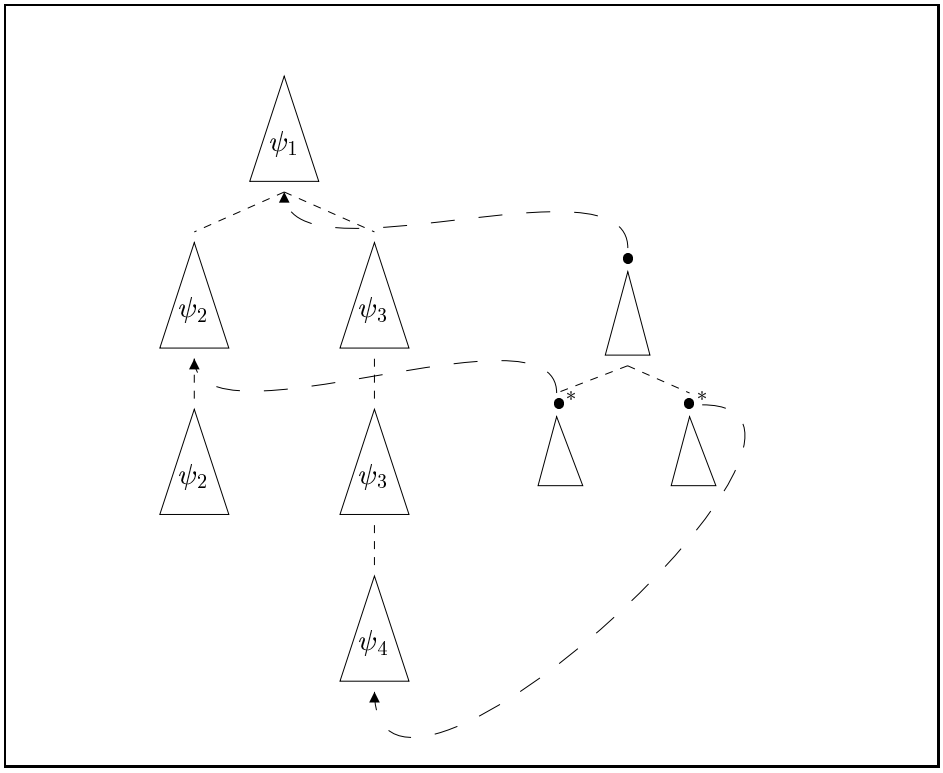


FIGURE 4.1: NON-LOCALITY OF DERIVATIONS IN (GENERAL) TDGS

dependencies only one single argument structure should be modified in a derivation step. In this sense derivation steps should be local in a grammar respecting the linguistic principles underlying TAGs.

To solve this problem, I will present a restricted version of TDGs called *local TDGs*. Local TDGs still allow multicomponent and subsertion-like derivations and even underspecified representations. The main idea of defining local TDGs is a restriction of the derivation: each derivation step must be local in the sense that the part of the old description that is modified by this step must belong to one single elementary description. With this locality restriction, the derivations can be described by a context-free grammar.

For local TDGs the semilinearity of the string languages can be shown by proving letter equivalence to context-free languages. Local TDGs and a sketch of the semilinearity proof for local TDLs (the string languages generated by local TDGs) are also presented in Kallmeyer 1997a.

The locality restriction concerns only the derivation in TDGs but not the definition of minimal trees, and therefore local TDGs are still more powerful than set-local MC-TAGs.

Sample local TDGs for the languages $L_2 := \{a_1^k \cdots a_4^k \mid k \geq 0\}$ and the copy languages suggest that the maximal number of marked names in one elementary description is somehow crucial for the generative capacity of a local TDG. Therefore I will introduce a hierarchy of local TDGs depending on the set of marked names. For this hierarchy, I will show that the

classes of rank 0 and 1 are both the context-free languages and that Tree Adjoining Languages form a subset of the class of rank 2. (Whether TALs are equal to local TDLs of rank 2 is an open question.) Furthermore, I will prove a pumping lemma for local TDLs of rank n and, using this lemma, I will show that the class of rank n for $n \leq 1$ is a proper subset of the class of rank $n+1$. In particular, the class of rank n contains the languages $L_i := \{a_1^k \cdots a_i^k \mid k \geq 0\}$ for $i \leq 2n$, whereas L_{2n+1} is not in this class. This hierarchy together with the pumping lemma is also presented in Kallmeyer 1998.

At the end of this chapter, following the idea of synchronous TAGs (see Chapter 2 and Shieber and Schabes 1990), I will introduce synchronous local TDGs. Roughly, the idea is that two local TDGs are connected by a synchronization relation. Depending on this relation, derivation is done in parallel in the two TDGs. Synchronous grammars are useful to describe dependencies between two languages that do not have the same structure but that somehow depend on each other in a compositional way.[1] With respect to the generative capacity of synchronous local TDGs I will show that, in contrast to TAGs, the synchronization of a local TDG with a second local TDG does not increase the set of languages generated by these grammars.

In this chapter first the notion of semilinearity will be introduced. Then a definition of local TDGs will be given and the semilinearity of the string languages of local TDGs will be proven. Further, a hierarchy of local TDGs is introduced based on the number of marked node names, and the classes of rank 0, 1 and 2 of this hierarchy are considered. Then a pumping lemma for local TDLs of rank n is proven. Finally, synchronous local TDGs are defined and it is shown that the synchronization does not increase the generative capacity of the grammars.

4.1 Definition of local TDGs

In this section I will define a restricted version of TDGs called local TDGs. This work was partly presented in Kallmeyer 1997a.

The main idea of local TDGs is a restriction of the derivation such that it is possible to describe the derivation process by a context-free grammar. Local TDGs still have the advantages of TDGs for natural languages and their string languages are semilinear.

Before coming to the introduction of local TDGs, I will present the definition of semilinearity together with the Parikh-Theorem. This theorem tells us that, if context-freeness of the derivation process can be shown for a grammar formalism, then semilinearity of the string languages also can be proven.

4.1.1 Definition of semilinearity

Semilinearity, introduced in Parikh 1966, is a language property closely related to the so-called Constant Growth Property. In this section, first the Constant Growth Property will be defined and then semilinearity will be introduced. We will see that the Constant Growth Property is a consequence of semilinearity, i.e. if a language L is semilinear, then the Constant Growth Property holds for L.

[1] In Chapter 5 synchronous local TDGs will be used to deal with the relation between syntax and semantics, and in particular to handle quantifier scope ambiguities.

Joshi (1985) introduces the notion of mild context-sensitive grammars (MCSG) by defining three necessary properties of MCSGs. He claims that grammars that are adequate for natural language structures are in the class of MCSGs. One of the properties of MCSGs is the Constant Growth Property. The idea of the Constant Growth Property is that the length of the words in a language grows in a linear way, e.g. $\{a^{2^n} | 1 \leq n\}$ does not belong to this class of languages.

The following definition of the Constant Growth Property can be found in Weir 1988, page 3.

Definition 4.1 (Constant Growth Property) *Let X be an alphabet and $L \subseteq X^*$ be a language. The Constant Growth Property holds for L, iff there is a constant $c_0 > 0$ and a finite set of constants $C \subset \mathbb{N} \setminus \{0\}$ such that for all $w \in L$ with $|w| > c_0$, there is a $w' \in L$ with $|w| = |w'| + c$ for some $c \in C$.*

As already mentioned, semilinearity is a more restricted property entailing the Constant Growth Property.

Let $\mathbb{N}$ be the set of the non-negative integers. $(\mathbb{N}, +)$ is a commutative monoid. For $(a_1, \ldots, a_n), (b_1, \ldots, b_n) \in \mathbb{N}^n$ and $m \in \mathbb{N}$ the following operations are defined: $(a_1, \ldots, a_n) + (b_1, \ldots, b_n) := (a_1 + b_1, \ldots, a_n + b_n)$ and $m(a_1, \ldots, a_n) := (ma_1, \ldots, ma_n)$.

A Parikh mapping is a function counting for each letter of an alphabet the occurrences of this letter in a word w.

Definition 4.2 (Parikh mapping) *Let $X = \{a_1, \ldots, a_n\}$ be an alphabet with some (arbitrary) fixed order of the elements. The Parikh mapping $p : X^* \to \mathbb{N}^n$ (wrt this order) is defined as follows:*
For all $w \in X^*$: $p(w) := (|w|_{a_1}, |w|_{a_2}, \ldots, |w|_{a_n})$, where $|w|_{a_i}$ is the a_i-length of w (number of occurrences of a_i).
For all $L \subseteq X^*$: $p(L) := \{p(w) | w \in L\}$ is the Parikh-image of L.

To define semilinearity we need the definition of letter equivalence. Two strings are letter-equivalent if they contain equal number of occurrences of each terminal symbol, and two languages are letter equivalent if every string in one language is letter-equivalent to a string in the other language and vice-versa.

Definition 4.3 (Letter-equivalent)

1. *Two words $x_1, x_2 \in X^*$ are letter-equivalent, iff there is a Parikh mapping $p : X^* \to \mathbb{N}^{|X|}$ such that $p(x_1) = p(x_2)$.*
2. *Two languages $L_1, L_2 \subseteq X^*$ are letter-equivalent, iff there is a Parikh mapping $p : X^* \to \mathbb{N}^{|X|}$ such that $p(L_1) = p(L_2)$.*

Definition 4.4 (Semilinear)

1. *Let $x_0, x_1, \ldots, x_m, 0 \leq m$ be in $\mathbb{N}^n$ for some $n \in \mathbb{N}$.*
   A set $\{x_0 + n_1 x_1 + \cdots + n_m x_m \,|\, n_i \in \mathbb{N}$ for $1 \leq i \leq m\}$ is a linear subset of $\mathbb{N}^n$.
2. *The union of finitely many linear subsets of $\mathbb{N}^n$ is a semilinear subset of $\mathbb{N}^n$.*
3. *A language $L \subseteq X^*$ is semilinear, iff there is a Parikh mapping p such that $p(L)$ is a semilinear subset of $\mathbb{N}^n$.*

Obviously, the Constant Growth Property holds for semilinear languages. To show this, the following constants (for the Constant Growth Property) can be chosen: suppose that L is semilinear, and $p(L) \subset \mathbb{N}^n$ is a semilinear Parikh image of L where $p(L)$ is the union of the linear sets $M_1, \ldots, M_l$. Then the Constant Growth Property holds for L with

$c_0 := max\{\sum_{i=1}^{n} y_i \mid$ there are $x_1, \ldots, x_m$ such that
$\{(y_1, \ldots, y_n) + n_1 x_1 + \cdots + n_m x_m \mid n_i \in \mathbb{N}\}$
is one of the sets $M_1, \ldots, M_l\}$,

$C := \{\sum_{i=1}^{n} y_i \mid$ there are $x_1, \ldots, x_m$ such that
$\{x_1 + n_1(y_1, \ldots, y_n) + \cdots + n_m x_m | n_i \in \mathbb{N}\}$
is one of the sets $M_1, \ldots, M_l\}$

Clearly, each language that is letter-equivalent to a semilinear language is semilinear as well, since the Parikh-images of the two languages are equal.

As Parikh (1966) has proven, the following proposition holds:

Proposition 4.1 (Parikh-Theorem) *Each context-free language is semilinear.*

Therefore, in order to show the semilinearity of some language L, it is sufficient to show that L is letter-equivalent to a context-free language.

It seems to be plausible that the Constant Growth Property and even the more restricted semilinearity hold for natural languages (see Joshi 1985 and Vijay-Shanker et al. 1987). As far as I know, the only example for a possibly non-semilinear phenomenon in natural language is case stacking in Old Georgian (see Michaelis and Kracht 1996). Since there are no speakers of Old Georgian, it is difficult to test whether there is really a (theoretically) infinite progression of stacking possible. As long as these are the only examples, there is no reason to assume natural languages not to be semilinear.

Consequently, if a grammar formalism is intended to capture human language capacity, it should be expected to provide for semilinearity of the string languages it generates.

4.1.2 Local descriptions

In the following, local TDGs will be introduced. As already mentioned above, local TDGs are TDGs where the derivation process is restricted in such a way that it can be captured by a context-free grammar. The nonterminals of this grammar describe states of single elementary descriptions used in the course of the derivation in the TDG. (Here the term "state" is not used in a technical sense. A state consists of an elementary description and information about the node names occurring in this description, e.g. whether a node name is a leaf name in the derived description.)

To obtain context-freeness of the derivation, the restriction must be such that a single derivation step only depends on one single elementary description that was added before and, besides this elementary description, the state of no other elementary description added before is modified by this derivation step. This elementary description is called the *derivation description* of this derivation step. If the derivation process is context-free, then it can be shown that each local Tree Description Language (local TDL, string language of a local TDG) is letter-equivalent to some context-free language. As context-free languages are semilinear (see Prop. 4.1), one can conclude that local TDLs are semilinear.

The main difference between local TDGs and TDGs in general is that in local TDGs

1. in one derivation step all node names in the old description that are used for new node name equivalences added in this step have to be part of one single elementary description (the derivation description), and

2. if a node name is used in more than one derivation step to introduce new node name equivalences, then only in the first of these steps this may cause a modification of old states.

To understand the intuitions behind the definitions of local descriptions and of derivations in local TDGs, it is helpful to have an idea of the way the nonterminals of the context-free grammar G_{CF} describing the derivation look like. These nonterminals describe the states of elementary descriptions used in the course of a derivation. For a description ϕ derived in the TDG, in the corresponding derivation in G_{CF} there is one nonterminal (state) Z_{ψ_d} for each start or elementary description ψ_d added in the course of the derivation of ϕ. Z_{ψ_d} specifies in which way the names of ψ_d can be used in a new derivation step. Furthermore the states specify whether the derived description has a minimal tree. For each node name k in ψ_d that has not yet been used to introduce a new node name equivalence, Z_{ψ_d} gives information about

(a) whether k has a parent or daughter in ϕ,

(b) whether k is minimal or a leaf name in ϕ and

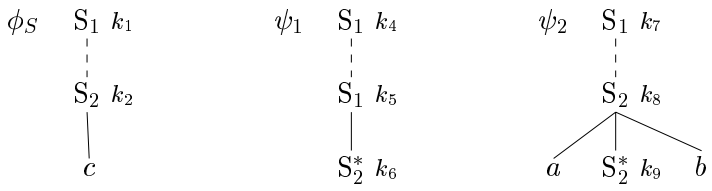
(c) whether k is strongly dominated by a name k' such that $\phi \vdash a(\delta(k')) \approx v$ for certain $a \in A$ and $v \in V \cup \{\bot\}$ in G_T.

Instead of the third information (c) for k, it is also possible to define a local variant of TDGs such that the relevant information (besides (a) and (b)) is the following:

(c') whether k strongly dominates a name k' such that $\phi \vdash a(\delta(k')) \approx v$ for certain $a \in A$ and $v \in V \cup \{\bot\}$ in G_T.

But it is not possible to have a context-free derivation grammar with (c) and (c') encoded in the nonterminals for each k. As we will see later, for each description ϕ that is elementary or derived in a local TDG and for each $k \in node(\phi)$: if k is strongly dominated by any name, then the attribute value pairs of all the names strongly dominating k are all the same. Therefore I choose to encode the property (c) and not (c') in the states of elementary descriptions.

As an example of a context-free grammar describing derivations in a (local) TDG, consider Fig. 4.2. For this example, labels S_1 and S_2 are used instead of feature structures. Each nonterminal (state) in the context-free grammar is a pair of a description from the TDG (either ϕ_S or ψ_1 or ψ_2) and a conjunction that encodes information about the node names occurring in this description. The first production signifies that a derivation in the TDG starts with the start description ϕ_S where the node name k_2 is strongly dominated by a node name with nonterminal S_1 (encoded by $dom_{\uparrow}(k_2, S_1)$) and the terminal c is added. The following two productions express that to ϕ_S either ψ_1 or ψ_2 can be added. In both cases, the state of ϕ_S is modified in such a way that k_1 and k_2 have been used for new equivalences ($equ(k_1) \wedge equ(k_2)$). The state of the new elementary description is such that the marked name has a child since it becomes equivalent to k_2. In the third production (i.e. when adding ψ_2 to ϕ_S in the TDG), the two terminals a, b are added, and the node name k_8 is strongly dominated by a node name with nonterminal S_1. From a state of ϕ_S with



Letter-equivalent context-free grammar (S is the start symbol and $\{a, b, c\}$ is the terminal alphabet):

$S \to \langle \phi_S, dom_\uparrow(k_2, S_1)\rangle c$
$\langle \phi_S, dom_\uparrow(k_2, S_1)\rangle \to \langle \phi_S, equ(k_1) \wedge equ(k_2)\rangle\langle \psi_1, child(k_6)\rangle$
$\langle \phi_S, dom_\uparrow(k_2, S_1)\rangle \to \langle \phi_S, equ(k_1) \wedge equ(k_2)\rangle\langle \psi_2, child(k_9) \wedge dom_\uparrow(k_8, S_1)\rangle ab$
$\langle \phi_S, equ(k_1) \wedge equ(k_2)\rangle \to \epsilon$
$\langle \psi_1, child(k_6)\rangle \to \epsilon$
$\langle \psi_2, child(k_9) \wedge dom_\uparrow(k_8, S_1)\rangle \to \langle \psi_2, child(k_9) \wedge equ(k_7) \wedge equ(k_8)\rangle\langle \psi_1, child(k_6)\rangle$
$\langle \psi_2, child(k_9) \wedge dom_\uparrow(k_8, S_1)\rangle \to$
$\quad \langle \psi_2, child(k_9) \wedge equ(k_7) \wedge equ(k_8)\rangle\langle \psi_2, child(k_9) \wedge dom_\uparrow(k_8, S_1)\rangle ab$
$\langle \psi_2, child(k_9) \wedge equ(k_7) \wedge equ(k_8)\rangle \to \epsilon$

FIGURE 4.2: SAMPLE TDG WITH A LETTER-EQUIVALENT CONTEXT-FREE GRAMMAR

$equ(k_1) \wedge equ(k_2)$, the empty word can be derived since no further description must be added with ϕ_S as derivation description in order to obtain a description that has a minimal tree. The same holds for ψ_1 if k_6 has a child (see the fifth production). To ψ_2, either ψ_1 or ψ_2 can be added (sixth and seventh production) similar to adding ψ_1 or ψ_2 to ϕ_S. Once ψ_2 was used as a derivation description with new equivalences for k_7 and k_8, no further derivation step must be performed with this description as derivation description (see last production). The states used in this context-free grammar are slightly simplified compared to those that will be used later, the example serves only as an illustration of the idea of a context-freee grammar describing the derivation.

As already mentioned, for a derivation step with an old description ϕ and a derivation description ψ_d, the following must be the case: no other state of some elementary description ψ that was added before can be influenced by this derivation step. This means that Z_{ψ_d} (in the derivation in G_{CF} corresponding to the derivation of ϕ in the TDG) not only is the state on which the derivation step depends but it is also the only state that is modified by this step. To guarantee that this is the case, the form of the descriptions and the derivation mode must be restricted. Roughly said, the idea of the local derivation mode is that all parts of the new elementary ψ that are between two marked or minimal names, must be "inserted" into one single strong dominance $\phi_1 \vdash_s k \lhd^* k'$ with $k' \in node(\psi_d)$.

The restricted descriptions allowed in local TDGs are called *local* descriptions. The description ψ_1 in Fig. 4.3 for example is local whereas ψ_2 and ψ_3 are not local descriptions. The reason why descriptions similar to ψ_2 should not be allowed, is the following: when adding this description, the subdescription with names k_{13}, k_{14} and k_{15} will be "inserted" between

two node names of a strong dominance $k_1 \lhd^* k_2$ where the marked k_{15} will become equivalent to k_2. But it is not clear where to put the subdescription with k_{10}, k_{11} and k_{12}. If one of the names k_{10}, k_{11} or k_{12} was marked, then ψ_2 would be local. Independet from marked names, ψ_3 should not be local because the minimal name k_{16} is parent of k_{17} and k_{18}. The problem with this description is that in a derivation step $\phi_1 \overset{\psi_3}{\Rightarrow} \phi_2$, k_{16} might be identified with some name k strongly dominating some k' in ϕ_1. Then the information about whether k' is strongly dominated by some name labelled by certain attribute value pairs might change. If k' does not belong to the derivation description, this means that the state of some elementary description has changed that should not be influenced by this derivation step. Therefore each marked or minimal name dominating any other minimal names must not have any daughters.

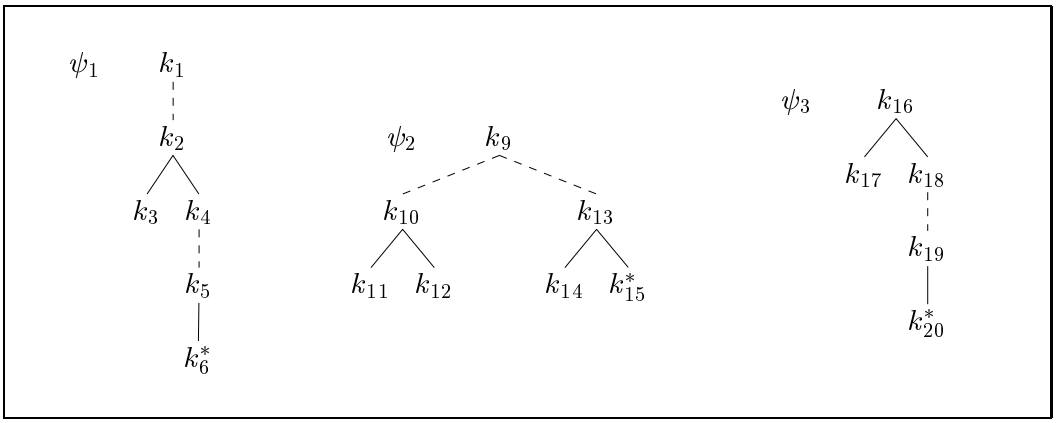


FIGURE 4.3: LOCAL AND NON-LOCAL ELEMENTARY DESCRIPTIONS

To avoid some special cases, local descriptions shall be K-minimal, i.e. for a local description ϕ and two names $k_1, k_2 \in node(\phi)$: $\phi \vdash k_1 \approx k_2$ holds iff $k_1 = k_2$ (see Def. 3.18 on page 87).

Definition 4.5 (Local description) *Let G be a TDG. An elementary description ψ in G is local iff ψ is K-minimal and for all $k_1, k_2, k_3 \in node(\psi)$:*

1. *If $\psi \vdash_s k_2 \lhd^* k_1$, and $\psi \vdash_s k_3 \lhd^* k_1$, then $k_2 = k_3$.*
2. *If $\psi \vdash_s k_1 \lhd^* k_2$, $\psi \vdash_s k_1 \lhd^* k_3$ and $k_2 \neq k_3$,*
   then: $k_1 \in M_\psi$, and there are $k_4, k_5 \in K_\psi$ with $\psi \vdash k_2 \lhd^* k_4 \wedge k_3 \lhd^* k_5$.
3. *If $k_1, k_2 \in M_\psi$ with $k_1 \neq k_2$ and $\psi \vdash k_1 \lhd^* k_2$, such that there is no further marked name between k_1 and k_2,*
   then there is a k such that $\psi \vdash_s k_1 \lhd^* k$ and $\psi \vdash k \lhd^* k_2$, and for all $k_\psi \in K_\psi$: if $\psi \vdash k \lhd^* k_\psi$, then $\psi \vdash k_2 \lhd^* k_\psi$.
4. *If $k_1, k_2 \in M_\psi$ are as in 3., and if there are k_4, k_5 with $\psi \vdash k_4 \lhd^* k_5$, $\psi \vdash_s k_1 \lhd^* k_4$ and $\psi \vdash_s k_5 \lhd^* k_2$,*
   then there is no $k \neq k_4$ with $\psi \vdash_s k_1 \lhd^* k$, and for all $a \in A$ there is a $v \in V \cup \{\bot\}$ with: $\psi \vdash a(\delta(k_1)) \approx v \wedge a(\delta(k_2)) \approx v \wedge a(\delta(k_4)) \approx v \wedge a(\delta(k_5)) \approx v$.

According to this definition, in a local description no name may be strongly dominated by more than one other name (see 1. axiom), and only marked or minimal names k are allowed

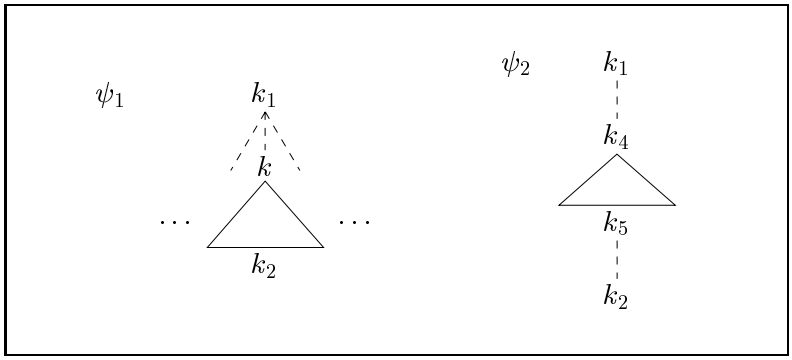


FIGURE 4.4: POSSIBLE FORMS OF LOCAL DESCRIPTIONS

to strongly dominate more than one other name. In the last case, each of the names strongly dominated by k must again dominate some further marked name (2. axiom). Furthermore each marked or minimal name dominating another marked name must strongly dominate some name (3. axiom). For marked names k_2 where k_1 is the next marked or minimal name dominating k_2 and $k_1 \lhd^* k_2$ is not a strong dominance, there are two possibilities (3. and 4. axiom):

1. Either (see ψ_1 in Fig. 4.4) k_2 has a parent and k is the name with $\psi \vdash_s k_1 \lhd^* k$ and $\psi \vdash k \lhd^* k_2$, and there are no further marked names allowed that are dominated by k without being dominated by k_2.
2. Or (see ψ_2 in Fig. 4.4) k_2 and k_1 are such that underspecification can occur. Then there are k_4 and k_5 with $\psi \vdash_s k_1 \lhd^* k_4$ and $\psi \vdash_s k_5 \lhd^* k_2$. In this case k_4, k_5, k_1 and k_2 are labelled by the same attribute value pairs, and there are no other names strongly dominated by k_1. As in the first case, there are no other marked names dominated by k_4 without being dominated by k_2.

In the second case, the pair (k_1, k_2) is called an *underspecification pair*:

Definition 4.6 (Underspecification pair) *Let ψ be a local elementary description in a TDG G.*

(a) (k_1, k_2) is called an underspecification pair in ψ, iff

1. *$k_1, k_2 \in M_\psi$,*
2. *there are $k_3 \neq k_2$ and $k_4 \neq k_1$ with $\psi \vdash_s k_1 \lhd^* k_3$, $\psi \vdash_s k_4 \lhd^* k_2$ and $\psi \vdash k_1 \lhd^* k_4 \wedge k_3 \lhd^* k_2$,*
3. *and there is no further marked name between k_1 and k_2.*

(b) $k \in node(\psi)$ is called top-underspecified iff there is a k' such that (k, k') is an underspecification pair in ψ.

(c) $k \in node(\psi)$ is called bottom-underspecified iff there is a k' such that (k', k) is an underspecification pair in ψ.

The elementary descriptions in Fig. 3.5 (p. 78), 3.7 (p. 80), 3.20 (p. 100), 3.21 (p. 101), 3.23 (p. 103) and also 3.24 (p. 105) in Chapter 3 are all local descriptions. The description

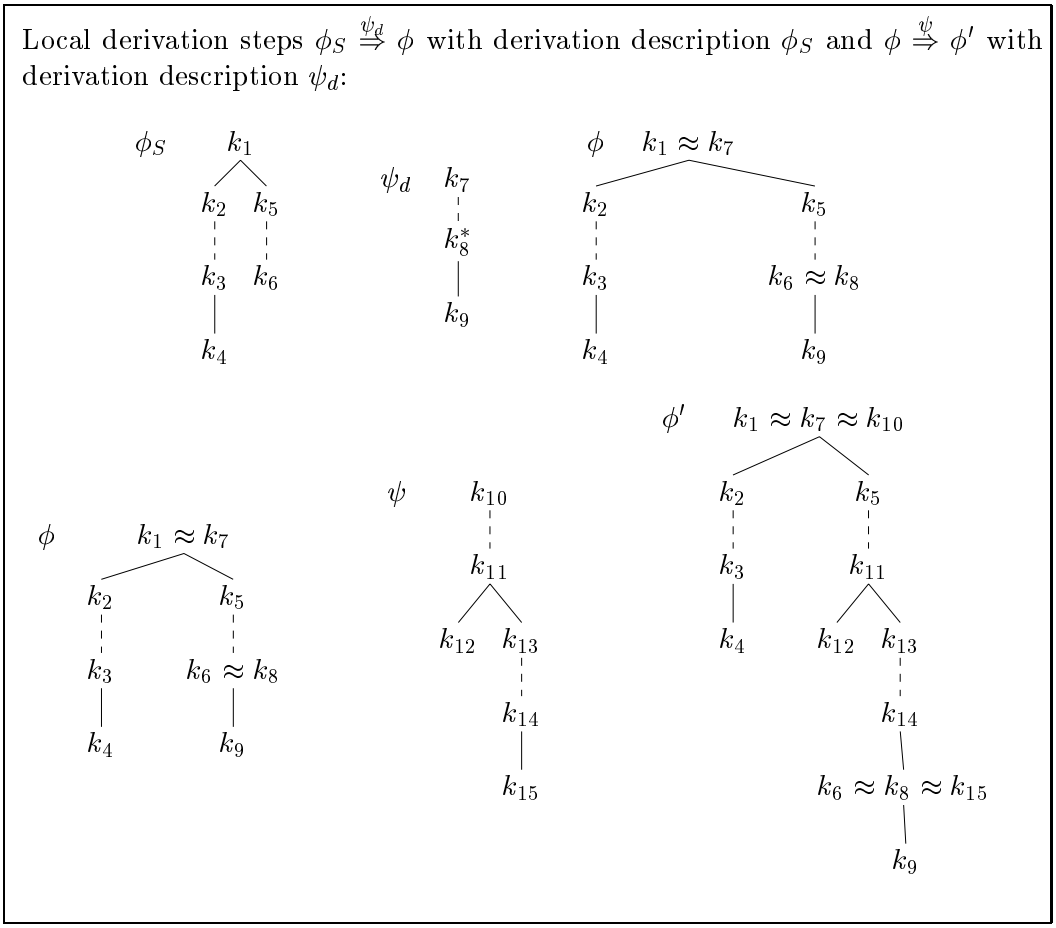


FIGURE 4.5: EXAMPLES OF LOCAL DERIVATION STEPS

$\psi(v_2)$ in Fig. 3.22 on page 102 is not local because the 1. condition in Def. 4.5 does not hold for it. Other examples of non-local descriptions are ψ_2 and ψ_3 in Fig. 4.3. ψ_2 is not local because the 2. axiom in Def. 4.5 is not fulfilled, and ψ_3 is not local because the 3. axiom does not hold.

A *local TDG* is a TDG containing only local descriptions (the start description is viewed as a local description with no marked names):

Definition 4.7 (Local TDG) *A TDG G is a local TDG, iff the start description (with an empty set of marked names) and all elementary descriptions in G are local.*

4.1.3 Local derivations

The notion of *locally derivable* with respect to a local TDG G, $\phi_1 \Rightarrow \phi_2$, for descriptions ϕ_1 and ϕ_2 can be inductively defined. As already explained, the main ideas of the definition of a local derivation step $\phi_1 \overset{\psi}{\Rightarrow} \phi_2$ are

1. to use only names from one elementary description ψ_d in the old description ϕ_1,
2. to use each name at most once for a derivation step by which the properties (a), (b), (c) mentioned on page 114 encoded in the states may be changed for this name, and
3. to insert "subtree descriptions" (e.g. the part with k_2, k_3, k_4 in the description ψ_1 of Fig. 4.3) into strong dominances $\phi_1 \vdash_s k \lhd^* k'$ with $k' \in node(\psi_d)$.

The derivation steps shown in Fig. 4.5 are for example local. In the first step $\phi_S \overset{\psi_d}{\Rightarrow} \phi$ the state of ϕ_S is modified. In the second step $\phi \overset{\psi}{\Rightarrow} \phi'$ only the state of ψ_d in the old ϕ is changed: k_7 and k_8 are used to introduce new node name equivalences, and the subdescription of ψ between k_{11} and k_{15} is inserted into the strong dominance $\phi \vdash_s k_5 \lhd^* k_8$ with $k_8 \in node(\psi_d)$. The only modification of properties of a name not occuring in ψ_d concerns k_6. In ϕ, k_6 is strongly dominated whereas in ϕ', k_6 has a parent, i.e. the properties (a) and (c) are changed for k_6. However, the properties of k_6 are the same as those of k_8. Since k_6 cannot be used any more for "state changing" new equivalences (it has already been used for $k_6 \approx k_8$), it is sufficient to observe k_8 instead of k_6. Similarly, after the derivation of ϕ', k_8 is no longer directly observed but the properties of k_{15} are then important. When deciding whether ϕ' has a minimal tree, the properties of k_{15} must be such that strong dominances can be eliminated. If this is the case, then the same holds for k_6 and k_8 as well.

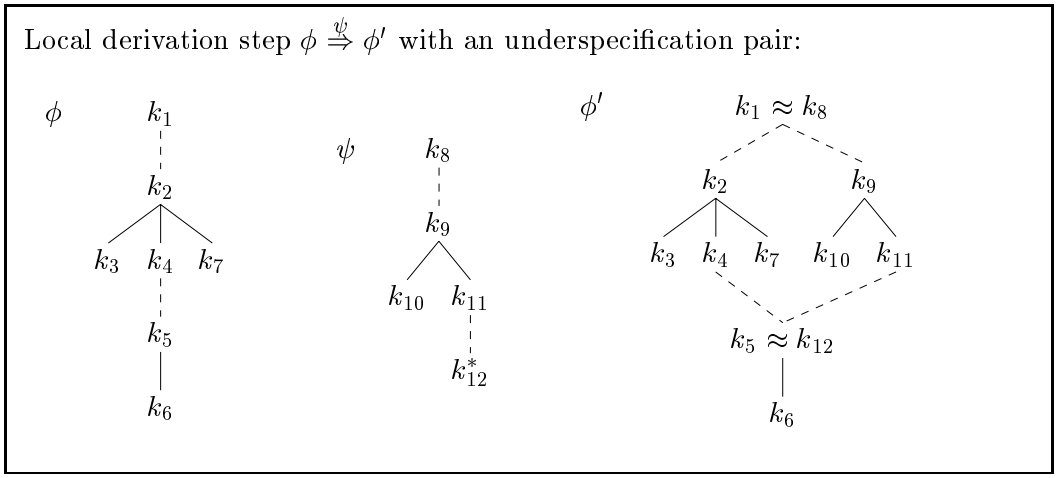


FIGURE 4.6: LOCAL DERIVATION STEP WITH AN UNDERSPECIFICATION PAIR

Exceptions from the idea to insert the part between two marked or minimal names into one strong dominance are the parts of a description that are between two names of an underspecification pair, e.g. between k_8 and k_{12} in ψ in Fig. 4.6. In derivation steps involving underspecification pairs the derivation must be such that the decision whether a derived description ϕ has a minimal tree or not depends only on the single states of the elementary descriptions used to derive ϕ. To guarantee this, it will not be allowed to use a bottom-underspecified name in the old description for the introduction of a new equivalence with a name that is not bottom-underspecified. Fig. 4.6 shows an example of a local derivation step resulting in an underspecified representation.

In the following definition of local derivation, axiom *(L1)* is the same as *(A1)* in the derivation definition 3.13, p. 76 for TDGs, and *(A2)* is a consequence of *(L2)*. *(A3)* is a consequence of *(L3)(ii)* and the form of the local descriptions. *(A4)* follows from *(L3)* and *(L4)*. Therefore, a result of a local derivation step satisfies the axioms *(A1)* to *(A4)* for derivations in TDGs.

Definition 4.8 (Local derivation in a local TDG) *Let G be a local TDG, ϕ_1 and ϕ_2 be descriptions with $\phi_S \stackrel{*}{\Rightarrow} \phi_1$, and let ψ be K-equivalent to a (local) elementary description in G.*
$\phi_1 \stackrel{\psi}{\Rightarrow} \phi_2$ holds iff there is a ψ_d with $\psi_d = \phi_S$ or $\phi_S \stackrel{*}{\Rightarrow} \phi \stackrel{\psi_d}{\Rightarrow} \phi' \stackrel{*}{\Rightarrow} \phi_1$, such that:

(L1) $\phi_2 \vdash \phi_1 \wedge \psi$.

(L2) *For all $k_\psi \in node(\psi)$: $k_\psi \in M_\psi$*
iff there is a $k_\phi \in node(\phi_1)$ with $\phi_2 \vdash k_\phi \approx k_\psi$
iff there is exactly one $k_d \in node(\psi_d)$ with $\phi_2 \vdash k_\psi \approx k_d$ and $\phi_1 \vdash k_\phi \approx k_d$ for all $k_\phi \in node(\phi_1)$ with $\phi_2 \vdash k_\phi \approx k_\psi$.

(L3) *For all $k_\psi \in M_\psi, k_d \in node(\psi_d)$ with $\phi_2 \vdash k_\psi \approx k_d$:*

- *(i)* *If there is a $k \neq k_d$ with $\phi_1 \vdash k \approx k_d$ such that either $\psi_d = \phi_S$ or ($\phi_S \stackrel{*}{\Rightarrow} \phi \stackrel{\psi_d}{\Rightarrow} \phi' \stackrel{*}{\Rightarrow} \phi_1$) $\phi' \nvdash k \approx k_d$, then:*
  - *either k_ψ is a leaf name or there is a $k_m \in M_\psi$ with $\psi \vdash k_\psi \lhd^* k_m$, or k_ψ and k are both top-underspecified.*
  - *either k_ψ is minimal or there is a $k_m \in M_\psi$ with $\psi \vdash_s k_m \lhd^* k_\psi$ or k_ψ and k are both bottom-underspecified, and*
  - *for all $a \in A, v \in V \cup \{\bot\}$: if $\psi \vdash a(\delta(k_\psi)) \approx v$, then $\phi_1 \vdash a(\delta(k_d)) \approx v$.*
- *(ii)* *If there is no $k_m \in M_\psi \setminus \{k_\psi\}$ with $\psi \vdash k_\psi \lhd^* k_m$, then either k_d is leaf name in ϕ_1 or k_ψ leaf name in ψ.*
- *(iii)* *If there is a $k_m \in M_\psi \setminus \{k_\psi\}$ with $\psi \vdash k_m \lhd^* k_\psi$ and no more marked names between k_m and k_ψ such that (k_m, k_ψ) is no underspecification pair, and if there is a $k \neq k_\psi$ with $\psi \vdash_s k_m \lhd^* k$ and $\psi \vdash k \lhd^* k_\psi$, then:*
  k_d is not bottom-underspecified in ψ_d, and there is a k' with $\phi_1 \vdash_s k' \lhd^* k_d$ such that for all k'': if $\psi \vdash k \lhd^* k''$, then $\phi_2 \vdash k' \lhd^* k''$.
- *(iv)* *If k_ψ is top-underspecified, then there is a k with $\phi_1 \vdash_s k_d \lhd^* k$.*
  If k_ψ is bottom-underspecified, then there is a k with $\phi_1 \vdash_s k \lhd^* k_d$.

(L4) *For all ϕ_3 fulfilling (L1) to (L3), the following holds:*
Let $f_1, f_2 \in K^K$ be mappings with $f_1(k) = f_2(k) = k$ for all $k \notin M_\psi$, and for all $k_m \in M_\psi$, there are $k_1, k_2 \in node(\phi_1)$ with $\phi_2 \vdash k_1 \approx k_m$ and $\phi_3 \vdash k_2 \approx k_m$ such that $f_1(k_m) = k_1$ and $f_2(k_m) = k_2$.
Then the following holds: if $\hat{f}_1(\phi_2) \vdash \hat{f}_2(\phi_3)$, then $\hat{f}_2(\phi_3) \vdash \hat{f}_1(\phi_2)$.

ψ_d is called the derivation description of the derivation step $\phi_1 \stackrel{\psi}{\Rightarrow} \phi_2$.

Note that the result of a local derivation step need not satisfy the axioms of local descriptions (e.g. ϕ' in Fig. 4.6).

With *(L2)*, all names in ϕ_1 used in one derivation step to add new node name equivalences are part of one single elementary description, the derivation description. Furthermore, as in

the case of general TDGs, for a $k \in node(\psi)$ there is a new equivalence iff k is marked or minimal and for each marked or minimal name there is exactly one equivalent name in the derivation description.

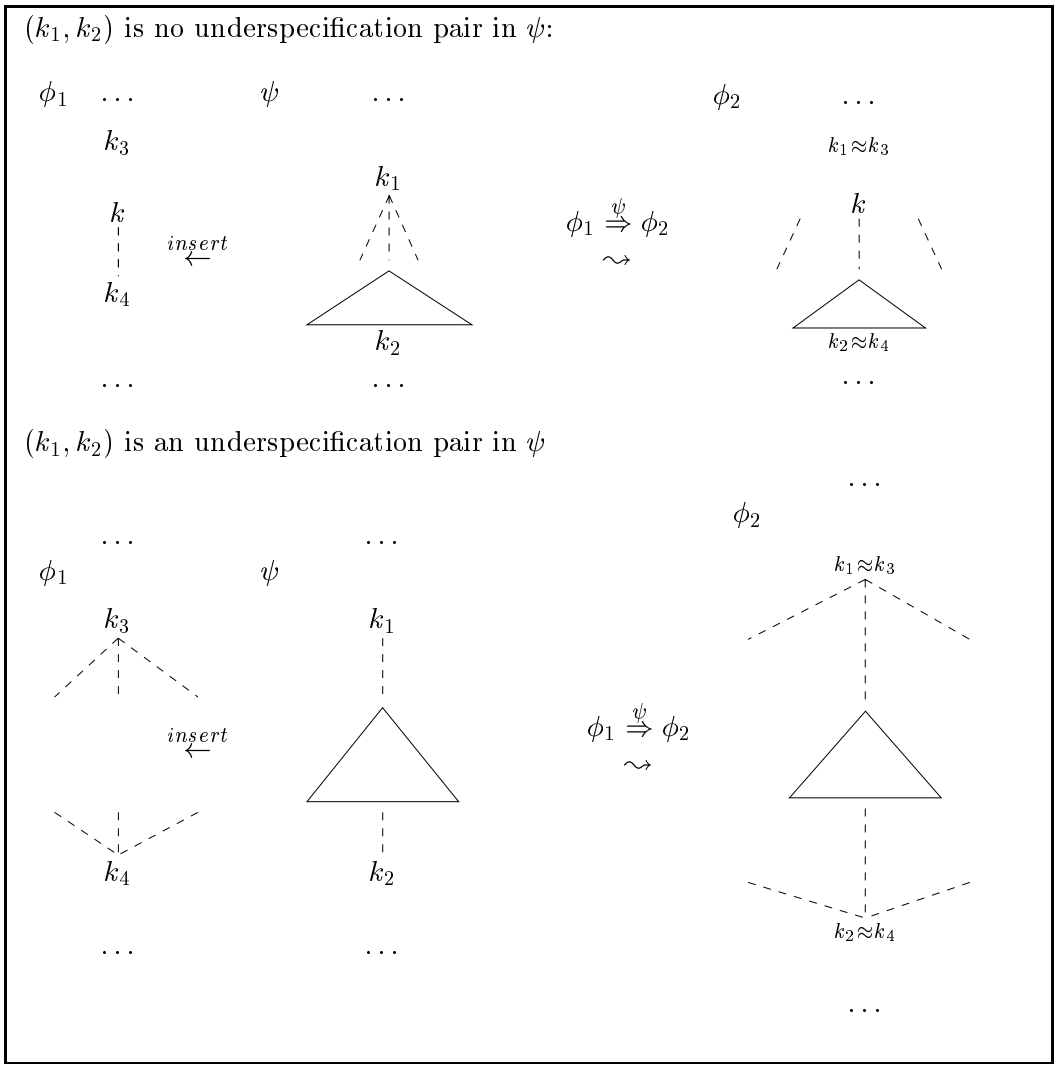


FIGURE 4.7: GENERAL FORM OF LOCAL DERIVATIONS

Because of *(L3)*, the following holds:

- With (i) for a marked k_ψ, the equivalent name k_d in ϕ_1 may be a name that was already used to introduce an equivalence $k_d \approx k$ in a preceding derivation step. But this is only allowed if k_ψ (a) is a leaf name or dominates another marked name, (b) is minimal or strongly dominated by some other name, and (c) does not add any information concerning the labels of the equivalent node names. Furthermore, if k_ψ is top- (or bottom-)underspecified, then k must also be top- (or bottom-)underspecified.

In other words, a name k_d in the old ϕ_1 may be used in more than one derivation step to introduce new node name equivalences, but this is only allowed if, apart from the first, all further new equivalences with this name do not modify the information with respect to k_d encoded in the state of the derivation description.

- With (*ii*) a marked or minimal name in ψ not dominating any other marked name either is a leaf name or it is identified with a leaf name in ϕ_1, i.e. an operation similar to substitution in TAGs takes place.
- With (*iii*) everything between two marked or minimal names must be "inserted" into one single strong dominance. Furthermore, only minimal names or bottom-underspecified names k_ψ can be identified with bottom-underspecified names k_d in the old description.
- With (*iv*), for two names k_1, k_2 in ψ that are are minimal or marked and that form an underspecification pair (k_1, k_2), k_1 must be identified with a name that is on the left side of a strong dominance in ϕ_1, and k_2 must be identified with a name that is on the right side of a strong dominance.

As in the case of general TDGs, the result of a derivation step must be maximally underspecified. This is expressed by *(L4)*.

With the local derivation mode, for two names k_1, k_2 in ψ with $\psi \vdash k_1 \lhd^* k_2$ that are either minimal or marked with no more marked names in between such that $\psi \vdash_s k_1 \lhd^* k_2$ does not hold: let k_3, k_4 be in ϕ_1 with $\phi_2 \vdash k_3 \approx k_1 \wedge k_4 \approx k_2$. Then the derivation has one of the two forms shown in Fig. 4.7.

The languages generated by a local TDG are defined similar to those generated by TDGs, but with a local derivation mode.

Definition 4.9 (Language of a local TDG) *Let $G = (A, V, T, D, \phi_S)$ be a local TDG.*

1. *The local description language of G is $L^l_D(G) := \{\phi \mid \phi_S \stackrel{*}{\Rightarrow} \phi\}$.*
2. *The local tree language of G is*
$$L^l_T(G) := \{B \mid \text{there is a } \phi \in L^l_D(G) \text{ such that } B \text{ is a minimal tree of } \phi\}.$$
3. *The local string language of G is*
$$L^l_S(G) := \{w \mid \text{there is a } B \in L^l_T(G) \text{ with } w = yield(B)\}.$$

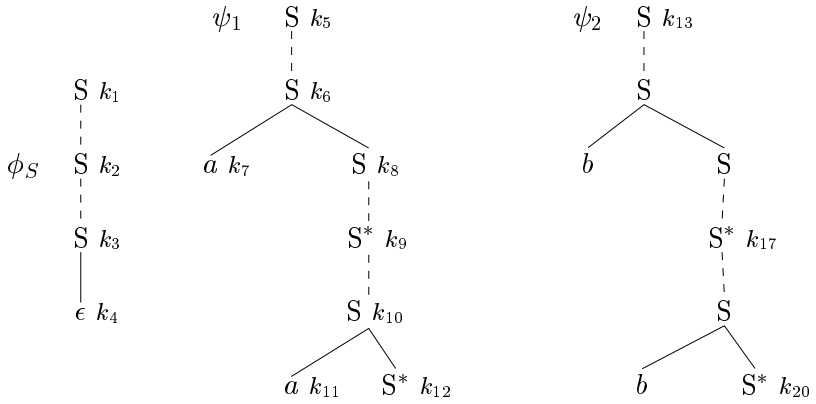
A language L is a local Tree Description Language (local TDL) iff there is a local TDG G with $L = L^l_S(G)$.

A sample local TDG is given in Fig. 4.8. This local TDG generates the string language $\{w_1w_2 \mid w_1w_2 \in \{a,b\}^* \text{ and } w_1 \text{ and } w_2 \text{ are letter-equivalent}\}$ in such a way that for each $w_2 \in \{a,b\}^*$ one single description is derived. The minimal trees of this description yield all w_1w_2 with w_1 letter-equivalent to w_2.

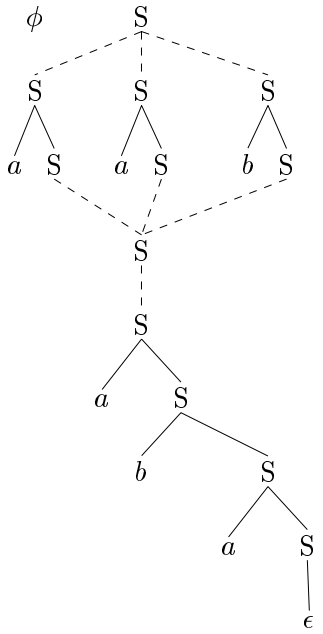
Clearly the languages $\{a_1^k a_2^k \ldots a_n^k \mid k \in \mathbb{N}\}$ for $n \geq 5$ and the copy languages $\{w^n \mid w \in T^*\}$ for $n \geq 3$ (see Fig. 3.10 on page 83 and 3.11 on page 84) are local Tree Description Languages. It is easy to construct an equivalent local TDG G_L for each TAG G_A. (This will be done in the proof of Prop. 4.8.) Consequently, local TDLs are a proper superset of TALs.

Derivation steps similar to substitution and adjunction in TAGs as in Fig. 3.5 (p. 78) and 3.7 (p. 80) are possible in local TDGs. In contrast to TDGs as presented in Chapter 3, even the following holds: if an elementary description ψ with $K_\psi = \emptyset$ is added in a derivation

Local TDG G for the string language $\{w_1w_2 \mid w_1w_2 \in \{a,b\}^*$ and w_1 and w_2 are letter equivalent$\}$:



For $w_2 = aba$, the following description ϕ is derived:



with $\{w_1w_2 \mid w_1w_2 = yield(B)$ for a minimal tree B of $\phi\}$
$= \{aababa, abaaba, baaaba\}$

FIGURE 4.8: SAMPLE LOCAL TDG

step, then the minimal name must be identified with a leaf name in the old description and therefore the derivation step must be a kind of TAG-substitution. Consequently a derivation step as shown in Fig. 3.6, p. 79 is not local. Similarly, for adjunction-like derivation steps: if an elementary description with exactly one marked name is added such that this marked name is a leaf name, then all node names except the minimal name must be inserted into one single strong dominance in the old description, and therefore the derivation step must be similar to adjunction.

Subsertion-like derivation steps as in Fig. 3.23 (p. 103) are also possible. In the case of Fig. 3.23, all node names used for new equivalences are taken from the description ϕ_1 that is an elementary description anchored by the verb *penses-tu*. Therefore this derivation step is local. However, subsertion in TDGs also can be non-local, and therefore such derivation steps are not generally possible in local TDGs.

Concerning underspecification, derivations as shown in Fig. 3.24 on page 105 are still possible because in the new elementary description the minimal name and the marked name have the same attribute value pairs, and they are identified with names from the same elementary description. In contrast to this, underspecified representations for PP-attachment as in Fig. 3.25 on page 106 are no longer allowed because the node names chosen for new equivalences belong to different elementary descriptions. A derivation of underspecified representations for PP-attachment ambiguities is not generally excluded, but if the linguistic principles underlying TAGs, in particular the uniqueness of the anchor for an elementary tree or description are adopted, then a single description must be derived for each of the two readings. Nevertheless local TDGs seem to be more suitable for natural languages than (general) TDGs because, as we will see later, as a consequence of the local derivation certain restrictions (e.g. island conditions for quantifier scope) result from the grammar formalism.

Obviously the properties **(LP)**, **(D)** and **(L)** (see Lemmata 3.3, 3.4 and 3.5 in Section 3.3) hold also for derivations in local TDGs. Therefore the derivation procedure given in Section 3.3 can be easily modified for local TDGs, and it can be shown that for lexicalized local TDGs the word problem is decidable.

4.2 Formal properties of local TDGs

In this section, first the semilinearity of local TDLs is shown, and then set-local MC-TAGs and local TDGs are compared. Although there are obvious similarities between these two formalisms, we will see that they are not derivationally equivalent.

4.2.1 Semilinearity of local TDLs

The semilinearity of local TDLs is proven by showing that each local TDL is letter-equivalent to a context-free language. Since context-free languages are semilinear (Parikh 1966), it can then be concluded that local TDLs are also semilinear.

To facilitate further considerations concerning formal properties of local TDGs, I will now introduce the notion of *label complete* local TDGs. Roughly said, a local TDG is label complete if for all node names the labels are fully specified.

Definition 4.10 (Label complete) *A local TDG G is label complete, iff for all elementary or start descriptions ψ in G and for all $k \in node(\psi)$: for each $a \in A$ there is a $v \in V \cup \{\bot\}$ with $\psi \vdash a(\delta(k)) \approx v$, and for all $t \in T \cup \{\epsilon\}$ either $\psi \vdash \delta(k) \approx t$ holds or $\psi \vdash \neg\delta(k) \approx t$.*

It can be easily shown that for each local TDG a weakly (and even derivationally) equivalent label complete local TDG can be constructed:

Lemma 4.1 *For each local TDG G there is a label complete local TDG G' such that G and G' are weakly and derivationally equivalent.*

Proof Assume $G = (A, V, T, D, \phi_S)$ to be a local TDG with $V = \{v_1, \ldots, v_n\}$ and $T = \{t_1, \ldots, t_m\}$.
Construction of a label complete equivalent TDG $G' = (A', V', T, D', \phi'_S)$:

1. $A' := A \cup \{start\}$ with $start \notin A$, $V' := V \cup \{yes, no\}$.
2. Elementary descriptions:
   $D' := D \cup \{\langle \phi_S, \emptyset \rangle\}$.
   For all $\langle \psi, K_\psi \rangle \in D'$ and all $k \in node(\psi)$: add the conjunct $start(\delta(k)) \approx no$ to ψ.
   $D_{temp} := D'$.

   (∗) For all $\langle \psi, K_\psi \rangle \in D_{temp}$: if there is a $k \in node(\psi)$ and an $a \in A$ such that there is no $v \in V \cup \{\bot\}$ with $\psi \vdash a(\delta(k)) \approx v$, then:
   (i) If $\psi_0 := \psi \wedge a(\delta(k)) \approx \bot$ is a description, then add $\langle \psi_0, K_\psi \rangle$ to D'.
   (ii) For all i, $1 \leq i \leq n$ $(n = |V|)$: if $\psi_i := \psi \wedge a(\delta(k)) \approx v_i$ is a description, then add $\langle \psi_i, K_\psi \rangle$ to D'.
   (iii) Remove $\langle \psi, K_\psi \rangle$ from D'.
   $D_{temp} := D'$.

   Repeat $(*)$ until for all $\langle \psi, K_\psi \rangle \in D'$, for all $k \in node(\psi)$ and for all $a \in A$ there is a $v \in V \cup \{\bot\}$ with $\psi \vdash a(\delta(k)) \approx v$.

   (∗∗) For all $\langle \psi, K_\psi \rangle \in D_{temp}$: if there is a $k \in node(\psi)$ such that for all $a \in A$ there is no $v \in V$ with $\psi \vdash a(\delta(k)) \approx v$, and such that there is no $t \in T \cup \{\epsilon\}$ with $\psi \vdash \delta(k) \approx t$, then:
   (i) If $\psi_0 := \psi \wedge \delta(k) \approx \epsilon$ is a description, then add $\langle \psi_0, K_\psi \rangle$ to D'.
   (ii) For all i, $1 \leq i \leq m$ $(m = |T|)$: if $\psi_i := \psi \wedge \delta(k) \approx t_i$ is a description, then add $\langle \psi_i, K_\psi \rangle$ to D'.
   (iii) Remove $\langle \psi, K_\psi \rangle$ from D'.
   $D_{temp} := D'$.

   Repeat $(**)$ until the condition for label completeness holds for G'.
   For all $\langle \psi, K_\psi \rangle \in D'$ where ψ was obtained from ϕ_S:

   if k_m minimal in ψ, choose a new name k and then:
   $\psi_{new} := \psi \wedge k \lhd k_m \wedge start(\delta(k)) \approx yes \bigwedge \{a(\delta(k)) \approx \bot \mid a \in A\}$.
   Replace $\langle \psi, K_\psi \rangle$ by $\langle \psi_{new}, K_\psi \rangle$.

3. Start description: $\phi'_S := start(\delta(k_s)) \approx yes \bigwedge\{a(\delta(k_s)) \approx \bot \mid a \in A\}$ for some new node name k_s.

The result is a local TDG $G' = (A', V', D', \phi'_S)$ that is label complete.
Clearly, $L^l_S(G) = L^l_S(G')$ holds and G and G' are even derivationally equivalent.

□

Note that the only reason why G and G' in this lemma are not strongly equivalent, is the introduction of an additional name k with $start(\delta(k)) \approx yes$ for all descriptions obtained form the start description in G. This is necessary in order to have one single start description. For each $B \in L^l_T(G)$, there is exactly one tree in $L^l_T(G')$ consisting of one root node with the tree B as single daughter.

Up to now I have only considered the strings yielded by trees. For the proof of the semilinearity of local TDGs I need the notion of the set of strings yielded by a description ϕ. These are the strings that can be obtained by reading the terminals in the leaf names from left to right. Since linear precedence is not necessarily completely specified, there might be more than one string in the yield of a description. (E.g., the yield of the description ϕ in Fig. 4.8 is the set $\{aababa, abaaba, baaaba\}$.) Even for descriptions with leaf names labelled by nonterminals, the yield is defined. In this case, only the leaf names with terminal labels are taken into account.

Definition 4.11 (Yield of a description) *Let ϕ be a description with terminals T such that for all leaf names k_1, k_2 in ϕ, either $\phi \vdash k_1 \approx k_2$ or $\phi \vdash k_1 \not\approx k_2$.*
$K^T_\phi := \{k \mid k$ is leaf name in ϕ and there is a $t \in T \cup \{\epsilon\}$ with $\phi \vdash \delta(k) \approx t\}$.
Let n be the number of equivalence classes of the relation $R \subseteq K^T_\phi \times K^T_\phi$ with $\langle k_1, k_2\rangle \in R$ iff $\phi \vdash k_1 \approx k_2$. Then the yield of ϕ is

$$\begin{aligned} Yield(\phi) \quad := \quad \{w = t_1 \ldots t_n \quad \mid \quad & \text{there are } k_1, \ldots, k_n \in K^T_\phi \text{ with } \phi \vdash \neg k_i \approx k_j \\ & \text{and } \phi \nvdash \neg k_i \prec k_j \text{ for all } 1 \leq i < j \leq n, \\ & \text{and } \phi \vdash \delta(k_i) \approx t_i \text{ for } 1 \leq i \leq n\} \end{aligned}$$

The elements in the yield of a description are all letter-equivalent. Clearly, if a description has a minimal tree, then the yield of this tree is in the yield of the description.

Now semilinearity of local TDLs will be shown by proving letter equivalence of local TDLs and context-free languages. As already explained above, the main idea of the proof is that the productions of the context-free grammar describe possible derivation steps. The nonterminal symbols of G_{CF} describe states of elementary descriptions used in the course of the derivation of some description ϕ. They should contain enough information to determine in which way the corresponding description can be used for another derivation step. These nonterminals are pairs consisting of an elementary or start description ψ and a conjunction containing for each $k \in node(\psi)$ the following conjuncts:

- $par(k)$ or $\neg par(k)$ which signifies either that k has a parent in the derived ϕ or that there is no parent of k in ϕ,
- $child(k)$ (signifying that k is parent of some k' in ϕ) or $\neg child(k)$
- $leaf(k)$, which means that k is a leaf name in ϕ, or $\neg leaf(k)$,
- $min(k)$ (i.e. k is minimal in ϕ) or $\neg min(k)$,

- $equ(k)$, signifying that k was already used for the introduction of a new node name equivalence, or $\neg equ(k)$,
- $top(k)$, meaning that there is a top-underspecified k' with $\phi \vdash k \approx k'$, or $\neg top(k)$,
- $bot(k)$, meaning that there is a bottom-underspecified k' with $\phi \vdash k \approx k'$, or $\neg bot(k)$,
- for each $X \subseteq A \times (V \cup \{\bot\})$, either $dom_{\uparrow}(k, X)$ (this means that k is strongly dominated by some name with feature structure X) or $\neg dom_{\uparrow}(k, X)$ is one of the conjuncts.

As an example, I will give the letter-equivalent context-free grammar for the local TDG G in Fig. 4.8 with $L^l_S(G) = \{w_1 w_2 \mid w_1 w_2 \in \{a, b\}^*$ and w_1 and w_2 are letter equivalent$\}$. In the productions of the context-free grammar only the relevant parts of the conjunctions are given, i.e. those parts that are not already implied by ψ itself.

The context-free grammar contains the following productions:

p1: $S \to \langle \phi_S, \ldots \rangle$

p2: $\langle \phi_S, \ldots \rangle \to \epsilon$

p3: $\langle \phi_S, \ldots \rangle \to$
$a \quad a$
$\langle \phi_S, equ(k_1) \wedge equ(k_2) \wedge equ(k_3) \wedge par(k_3) \wedge top(k_1) \wedge bot(k_2) \ldots \rangle$
$\langle \psi_1, child(k_{12}) \ldots \rangle$

p4: $\langle \phi_S, \ldots \rangle \to$
$b \quad b$
$\langle \phi_S, equ(k_1) \wedge equ(k_2) \wedge equ(k_3) \wedge par(k_3) \wedge top(k_1) \wedge bot(k_2) \ldots \rangle$
$\langle \psi_2, child(k_{20}) \ldots \rangle$

p5: $\langle \phi_S, equ(k_1) \wedge equ(k_2) \wedge equ(k_3) \wedge par(k_3) \wedge top(k_1) \wedge bot(k_2) \ldots \rangle \to \epsilon$

p6: $\langle \psi_1, child(k_{12}) \ldots \rangle \to \epsilon$

p7: $\langle \psi_1, child(k_{12}) \ldots \rangle \to$
$a \quad a$
$\langle \psi_1, child(k_{12}) \wedge equ(k_5) \wedge equ(k_9) \wedge equ(k_{10}) \wedge par(k_{10}) \ldots \rangle$
$\langle \psi_1, child(k_{12}) \ldots \rangle$

p8: $\langle \psi_1, child(k_{12}) \ldots \rangle \to$
$b \quad b$
$\langle \psi_1, child(k_{12}) \wedge equ(k_5) \wedge equ(k_9) \wedge equ(k_{10}) \wedge par(k_{10}) \ldots \rangle$
$\langle \psi_2, child(k_{20}) \ldots \rangle$

p9: $\langle \psi_1, child(k_{12}) \wedge equ(k_5) \wedge equ(k_9) \wedge equ(k_{10}) \wedge par(k_{10}) \ldots \rangle \to \epsilon$

p10: $\langle \psi_2, child(k_{20}) \ldots \rangle \to \epsilon$

p11: $\langle \psi_2, child(k_{20}) \ldots \rangle \to$
$a \quad a$
$\langle \psi_2, child(k_{20}) \wedge equ(k_{13}) \wedge equ(k_{17}) \wedge equ(k_{18}) \wedge par(k_{18}) \ldots \rangle$
$\langle \psi_1, child(k_{12}) \ldots \rangle$

p12: $\langle \psi_2, child(k_{20}) \ldots \rangle \to$
$b \quad b$
$\langle \psi_2, child(k_{20}) \wedge equ(k_{13}) \wedge equ(k_{17}) \wedge equ(k_{18}) \wedge par(k_{18}) \ldots \rangle$
$\langle \psi_2, child(k_{20}) \ldots \rangle$

p13: $\langle \psi_2, child(k_{20}) \wedge equ(k_{13}) \wedge equ(k_{17}) \wedge equ(k_{18}) \wedge par(k_{18}) \ldots \rangle \rightarrow \epsilon$

The production p1 signifies that the derivation (in the TDG) has to start with ϕ_S. According to p2, the derivation can already stop with ϕ_S, in other words ϕ_S has a minimal tree. With p3 and p4, ψ_1 or ψ_2 can be added to ϕ_S using k_1, k_2 and k_3 in ϕ_S for new equivalences. The production p5 is the only production with $\langle \phi_S, equ(k_1) \wedge equ(k_2) \wedge equ(k_3) \wedge par(k_3) \wedge top(k_1) \wedge bot(k_2) \ldots \rangle$ on the left side. This corresponds to the fact that ϕ_S cannot be used more than once as derivation description. With p6 to p9, the description ψ_1 can be used at most once as derivation description. If it is used as derivation description, then either ψ_1 or ψ_2 is added and in this derivation step the names k_5, k_9 and k_{10} in ψ_1 are used to introduce new equivalences. Similarly, with p10 to p13, ψ_2 can be used at most once as derivation description, and in this case new equivalences for the names k_{13}, k_{17} and k_{18} are added.

The derivation leading to the description ϕ in Fig. 4.8 in the local TDG is then described by the following context-free derivation:

$$
\begin{aligned}
S &\Rightarrow \langle \phi_S, \ldots \rangle \\
&\Rightarrow aa\langle \phi_S, equ(k_1) \wedge equ(k_2) \wedge equ(k_3) \wedge par(k_3) \wedge top(k_1) \wedge bot(k_2) \ldots \rangle \\
&\qquad \langle \psi_1, child(k_{12}) \ldots \rangle \\
&\Rightarrow aa\langle \psi_1, child(k_{12}) \ldots \rangle \\
&\Rightarrow bbaa\langle \psi_1, child(k_{12}) \wedge equ(k_5) \wedge equ(k_9) \wedge equ(k_{10}) \wedge par(k_{10}) \ldots \rangle \\
&\qquad \langle \psi_2, child(k_{20}) \ldots \rangle \\
&\Rightarrow bbaa\langle \psi_2, child(k_{20}) \ldots \rangle \\
&\Rightarrow aabbaa\langle \psi_2, child(k_{20}) \wedge equ(k_{13}) \wedge equ(k_{17}) \wedge equ(k_{18}) \wedge par(k_{18}) \ldots \rangle \\
&\qquad \langle \psi_1, child(k_{12}) \ldots \rangle \\
&\Rightarrow aabbaa\langle \psi_1, child(k_{12}) \ldots \rangle \\
&\Rightarrow aabbaa
\end{aligned}
$$

The string derived here is letter-equivalent to all words in the yield of ϕ.

Proposition 4.2 (Letter-equivalence of local TDLs and CFLs) *Let G_T be a local TDG. Then there is a context-free grammar G_{CF} such that $L^l_S(G_T)$ is letter-equivalent to $L(G_{CF})$.*

Proof Let $G_T = (A, V, T, D, \phi_S)$ be a local TDG. Without loss of generality let G_T be label complete.
Construction of a letter-equivalent context-free grammar $G_{CF} := (N, T, P, S)$:

1. G_{CF} has the same terminals T as G_T.
2. For each node name k define
$$
\begin{aligned}
F_k \quad := \quad & \{par(k), child(k), leaf(k), min(k), top(k), bot(k), equ(k)\} \\
& \cup \{dom_{\uparrow}(k, X) \mid X \subseteq A \times (V \cup \{\bot\}) \text{ such that for each } a \in A \\
& \qquad \text{there is exactly one } v \in V \cup \{\bot\} \text{ with } \langle a, v \rangle \in X\}
\end{aligned}
$$
The nonterminals are states Z of the form $Z = \phi_Z \wedge \xi_Z$ with:
   - $\phi_Z = \phi_S$ or there is a $K_\phi \subseteq node(\phi_Z)$ such that $\langle \phi_Z, K_\phi \rangle \in D$.
   - ξ_Z is a conjunction containing for all $k \in node(\phi_Z)$ and all formulas $\psi \in F_k$ either ψ or $\neg\psi$ as conjunct. These are all conjuncts in ξ_Z.

Additionally N contains a start symbol S different from all other nonterminals.

An entailment relation between descriptions derived from ϕ_s and states is defined:

Let $Z^\sim = \phi_Z^\sim \wedge \xi_Z^\sim$ be equivalent to a $Z \in N$ ("equivalent" is used in the sense of K-equivalence, i.e. it means that Z and $Z^\sim$ only differ in a bijection of K).

Definition: entailment of $Z^\sim$ by a description ϕ with $\phi_S \stackrel{*}{\Rightarrow} \phi$ (written $\phi \models Z^\sim$) is inductively defined:

(a) $\phi \models equ(k)$ iff there are ϕ_1, ϕ_2 such that $\phi_S \stackrel{*}{\Rightarrow} \phi_1 \Rightarrow \phi_2 \stackrel{*}{\Rightarrow} \phi$, $k \in node(\phi_1)$ and there is a $k' \notin node(\phi_1)$ such that $\phi_2 \models k \approx k'$.

(b) $\phi \models par(k)$ iff either $\phi \models equ(k)$ or there is a k' such that $\phi \models k' \lhd k$.

(c) $\phi \models child(k)$ iff either $\phi \models equ(k)$ or there is a k' such that $\phi \models k \lhd k'$.

(d) $\phi \models leaf(k)$ iff either $\phi \models equ(k)$ or k is a leaf name in ϕ.

(e) $\phi \models min(k)$ iff k is minimal in ϕ.

(f) $\phi \models top(k)$ iff there is a ψ and a top underspecified k' in ψ with $\phi_S \stackrel{*}{\Rightarrow} \phi_1 \stackrel{\psi}{\Rightarrow} \phi_2 \stackrel{*}{\Rightarrow} \phi$ and $\phi \vdash k \approx k'$.

(g) $\phi \models bot(k)$ iff there is a ψ and a bottom underspecified k' in ψ with $\phi_S \stackrel{*}{\Rightarrow} \phi_1 \stackrel{\psi}{\Rightarrow} \phi_2 \stackrel{*}{\Rightarrow} \phi$ and $\phi \vdash k \approx k'$.

(h) $\phi \models dom_\uparrow(k, X)$ iff either $\phi \models equ(k)$ or there is a k' with $\phi \vdash_s k' \lhd^* k$ and $\phi \models a(\delta(k')) \approx v$ for all $\langle a, v\rangle \in X$.

(i) Apart from this, entailment is defined as usual (see Def. 3.3 on page 57).

3. S is the start symbol in G_{CF}.

4. Productions P:

(i) If $Z_S = \phi_S \wedge \xi_S$ is a state with $\phi_S \models \xi_S$, then for one (arbitrarily chosen) $w_s \in Yield(\phi_S)$:

$$S \to w_s Z_S \qquad \in P$$

(ii) Let Z and Z' be states either for the same elementary description or both for the start description, let Z_{new} be a state for some elementary ψ, and let $w_\psi \in Yield(\psi)$. Then

$$Z \to w_\psi Z' Z_{new} \qquad \in P$$

iff for all ϕ with $\phi_S \stackrel{*}{\Rightarrow} \phi$ such that there is a $Z^\sim = \phi^\sim \wedge \xi^\sim$ equivalent to Z with $\phi \models Z^\sim$:

There are states $Z'^\sim = \phi'^\sim \wedge \xi'^\sim$ and $Z_{new}^\sim = \psi_{new}^\sim \wedge \xi_{new}^\sim$ equivalent to Z' and Z_{new} respectively such that: there is a ϕ' with

- $\phi \stackrel{\psi_{new}^\sim}{\Rightarrow} \phi'$,
- $\phi' \models Z'^\sim \wedge Z_{new}^\sim$,
- $\phi^\sim = \phi'^\sim$ and
- $\phi^\sim$ is the derivation description of the derivation step $\phi \stackrel{\psi}{\Rightarrow} \phi'$.

(iii) For all $Z \in N \setminus \{S\}$, $Z = \phi_Z \wedge \xi_Z$:

$$Z \to \epsilon \qquad \in P$$

iff for all $k \in node(\phi_Z)$: if there is no $t \in T \cup \{\epsilon\}$ with $\phi_Z \vdash \delta(k) \approx t$ and if X is the set of all pairs $\langle a, v\rangle$ such that $\phi_Z \vdash a(\delta(k)) \approx v$, then either $par(k)$ or $dom_{\uparrow}(k, X)$ or $equ(k)$ or $min(k)$ is in ξ_Z.

(iv) These are all productions in P.

G_{CF} is a context-free grammar.
A homomorphism $\varphi : (N \cup T)^* \to T^*$ is defined: $\varphi(X) := \epsilon$ for all $X \in N$, $\varphi(t) := t$ for all $t \in T$ and $\varphi(w_1 w_2) := \varphi(w_1)\varphi(w_2)$.

By induction on the length n of the derivation the following can be shown: For all $n \geq 0$:

$S \overset{n+1}{\Rightarrow} w_n$ wrt G_{CF} without applying ϵ-productions, and $Z_1, \ldots Z_n$ are all occurrences of nonterminals in w_n
iff there is a derivation $\phi_S \overset{n}{\Rightarrow} \phi_n$ wrt G_T such that there are pairwise different $Z_1^{\sim}, \ldots, Z_n^{\sim}$ with $Z_i^{\sim} = \phi_i^{\sim} \wedge \xi_i^{\sim}$ equivalent to Z_i $(1 \leq i \leq n)$ and:

- the descriptions that are elementary or start description and that have been used in the course of the derivation of ϕ_n, are exactly $\phi_1^{\sim}, \ldots, \phi_n^{\sim}$.
- $\phi_n \models Z_i^{\sim}$ for all $1 \leq i \leq n$,
- and each $w \in Yield(\phi_n)$ is letter-equivalent to $\varphi(w_n)$.

1. Induction start: $n = 0$: Induction claim holds because of the production with S on the left side in G_{CF}.
2. Induction step: Let the induction claim hold for $n-1$ with $S \overset{n}{\Rightarrow} w_{n-1}$ in G_{CF} and $\phi_S \overset{n-1}{\Rightarrow} \phi_{n-1}$ in G_T.
   Let ψ_d be one of those elementary or start descriptions added in the course of the derivation of ϕ_{n-1}, and let ψ be a new elementary description.
   In order to decide whether there is a ϕ_n such that $\phi_{n-1} \overset{\psi}{\Rightarrow} \phi_n$ with derivation description ψ_d, the following information must be available for each $k \in node(\psi_d)$ (additionally to the information given by ψ_d itself):
   (a) (in order to check *(L3) (i)*) one needs to know, whether $\phi_{n-1} \models equ(k)$ or $\phi_{n-1} \models \neg equ(k)$, whether $\phi_{n-1} \models top(k)$ or $\phi_{n-1} \models \neg top(k)$ and whether $\phi_{n-1} \models bot(k)$ or $\phi_{n-1} \models \neg bot(k)$. Furthermore, all $\langle a, v\rangle$, $a \in A, v \in V \cup \{\bot\}$ with $\phi_{n-1} \vdash a(\delta(k)) \approx v$ must be known,
   (b) (in order to check *(L3) (ii)*) one needs to know, whether $\phi_{n-1} \models leaf(k)$ or $\phi_{n-1} \models \neg leaf(k)$,
   (c) (in order to check *(L3) (iii)*) one needs to know, whether k is bottom-underspecified in ψ_d and whether $\phi_{n-1} \models dom_{\uparrow}(k, X)$ for some $X \in A \times (V \cup \{\bot\})$.
   (d) (in order to check *(L3) (iv)*) one needs to know, whether $\phi_{n-1} \models pred(k)$ or $\phi_{n-1} \models \neg pred(k)$ for $pred \in \{par, child, min, leaf\}$.
   There is one state Z_d of ψ_d occuring in w_{n-1} containing all these information such that there is a $Z_d^{\sim}$ equivalent to Z_d with $Z_d^{\sim} = \psi_d \wedge \xi_d$ and $\phi_{n-1} \models Z_d^{\sim}$.
   Clearly, for none of the names $k \in node(\phi_{n-1}) \setminus node(\psi_d)$, the properties described in the states are influenced in a derivation step with derivation description ψ_d.

Then, because of the construction of the productions in 4.(ii):
There is a production $Z_d \rightarrow w_\psi Z'_d Z_\psi$ ($w \in T^*$) that introduces a new state for ψ and, using this production, a word w_n can be derived from w_{n-1}
iff
there are $Z'^{\sim}_d = \psi_d \wedge \xi'_d$ and $Z^{\sim}_\psi = \psi \wedge \xi$ equivalent to Z'_d and Z_ψ respectively such that there is a ϕ_n with $\phi_{n-1} \overset{\psi}{\Rightarrow} \phi_n$ with derivation description ψ_d and $\phi_n \models Z'^{\sim}_d$ and $\phi_n \models Z^{\sim}_\psi$. Furthermore, each $w \in Yield(\phi_n)$ is letter-equivalent to $\varphi(w_\psi)\varphi(w_{n-1})$ and therefore letter-equivalent to $\varphi(w_n)$.
Consequently, the induction claim holds for w_n and ϕ_n.

With the construction of productions $Z \rightarrow \epsilon$ and with *(L3)*: for w_n, ϕ_n (as above): There is a $w'_n \in T^*$ that can be derived from w_n by applying only ϵ-productions iff ϕ_n has a minimal tree.
As productions in a context-free grammar can be applied in any order, in general the following holds:
$\phi_S \overset{*}{\Rightarrow} \phi$ wrt G_T, ϕ has a minimal tree B with $w = yield(B)$ iff there is a w' letter-equivalent to w such that $S \overset{*}{\Rightarrow} w'$ wrt G_{CF}.

□

As a consequence the following holds:

Proposition 4.3 (Semilinearity of local TDLs) *Each local Tree Description Language is semilinear.*

As a corollary of the semilinearity of local TDLs the Constant Growth Property holds for local TDLs and each local TDL is letter-equivalent to a regular language.

Since there are indexed languages that are not semilinear, another consequence of Prop. 4.3 is that the set of indexed languages is not a subset of the set of local TDLs. An example of an indexed language that is not semilinear is the language $L := \{(ab)^{n^2} d (ab)^{2^m} d (ab)^{n^2} \mid m, n \geq 0\}$ (see Hümpel-Schökel 1993 for a restricted indexed grammar generating L). L is not semilinear for the following reason: if L was semilinear, then the Constant Growth Property would hold for L. This means that there would be a constant c and a finite set C of constants such that for each $w \in L, |w| > c$ there would be a $w' \in L$ with $|w| = |w'| + c_0$ for some $c_0 \in C$. However, for $c_{max} := \max C$, $n := \max\{2, c_{max}, c\}$, $m \geq log_2 c_{max} + 2$ and $w = (ab)^{n^2} d (ab)^{2^m} d (ab)^{n^2}$ there is no such w'.

4.2.2 Set-local MC-TAGs and local TDGs

I will show in the following that, although local TDGs are very similar to set-local MC-TAGs (defined in Chapter 2, 2.4.1), these two formalisms are not equivalent. The crucial difference is that the locality of TDGs concerns only the derivation process but not the construction of a minimal tree for a derived description.

First, I will prove that for each set-local MC-TAG a local TDG can be constructed that is derivationally equivalent to the MC-TAG. Then, by giving a local TDG for an indexed string

language that cannot be generated by a set-local MC-TAG, I will show that local TDGs are more powerful than set-local MC-TAGs with respect to their derivational generative capacity.

The construction of a derivationally equivalent local TDG for a given set-local MC-TAG is as follows: I suppose that the MC-TAG is such that adjunction is not allowed at foot nodes of auxiliary trees. (This can be assumed without loss of generality, since the proof of Lemma 2.2, p. 29 for TAGs can be easily extended to MC-TAGs.) Then all elementary trees in the MC-TAG are replaced by corresponding descriptions. In these descriptions, for each node in a tree there is one top name k_t and one bottom name k_b with a dominance $k_t \lhd^* k_b$. An attribute *adj* controls adjunction in the following way: for the bottom name k_{foot} of a foot node, $adj(\delta(k_{foot})) \approx foot$ holds and top and bottom names of foot nodes are equivalent. For all other nodes with top name k_t and bottom name k_b, $adj(\delta(k_t)) \approx root$ holds, and if adjunction is not allowed at that node, this is expressed by $adj(\delta(k_b)) \approx root$. If there is an auxiliary tree β that must not be adjoined at this node, this constraint is expressed by $\beta(\delta(k_b)) \approx no$ (for the bottom name k_{foot} of the foot node of an auxiliary tree β, $\beta(\delta(k_{foot})) \approx yes$ holds). Obligatory adjunction for this node is expressed by $adj(\delta(k_b)) \approx foot$.

The elementary descriptions in the local TDG consist of a minimal node name and a "bottom" part. The minimal name has an attribute *cat* with a new nonterminal S_{new} as value. It dominates the descriptions in the bottom part. As an example, in Fig. 4.9 an MC-TAG for $\{a^n b^n c^n d^n \mid n \geq 0\}$ is given together with the derivationally equivalent local TDG constructed according to the proof of Prop. 4.4. The bottom part contains either the description of an initial tree where the minimal name is marked (see ψ_1 in Fig. 4.9) or descriptions of all elementary trees in one auxiliary set where the minimal names are removed and the names corresponding to the foot nodes are marked (see ψ_2 in Fig. 4.9). The start description is $\phi_S = k_1 \lhd k_2 \wedge cat(\delta(k_1)) \approx S_{new}$.

Proposition 4.4 *For each MC-TAG G_M there is a local TDG G_D such that the two grammars are derivationally (and therefore also weakly) equivalent.*

Proof: Let $G_M = \langle I, A, N, T, \mathcal{A} \rangle$ be a set-local MC-TAG. Without loss of generality suppose that for all $\beta \in A$, $C_\beta(u_f^\beta) = \emptyset$ holds (u_f^β is the foot node of β), that there are no substitution nodes in G_M, and that $|\mathcal{U}_\gamma| > 1$ for all $\gamma \in I \cup A$.
First, a description $d(\gamma, k_t)$ for a γ derived in G_M and a $k_t \in K$ is defined:
$d(\gamma, k_t)$ (unique up to K-equivalence), is a description that can be inductively constructed in the following way:

(a) Let u_r be the root of γ and $\{\beta_1, \ldots \beta_k\} := A \setminus C_\gamma(u_r)$.
If u_r is a leaf in γ, then $k_b := k_t$, else choose a new name k_b ($k_b \neq k_t$).
$d(\gamma, k_t) := k_t \lhd^* k_b$.
k_t is called the *top name*, and k_b the *bottom name* of u_r.

(b) If u_r is no leaf in γ, then $d(\gamma, k_t) := d(\gamma, k_t) \wedge adj(\delta(k_t)) \approx root$.

(c) If $\mu_\gamma(u_r) = X$ for some $X \in N$,
then $d(\gamma, k_t) := d(\gamma, k_t) \wedge cat(\delta(k_t)) \approx X \wedge cat(\delta(k_b)) \approx X$.

(d) If $\mu_\gamma(u_r) = t$ for some $t \in T \cup \{\epsilon\}$, then $d(\gamma, k_t) := d(\gamma, k_t) \wedge \delta(k_t) \approx t$.

(e) If $C_\gamma(u_r) = \emptyset$ and u_r is no leaf, then $d(\gamma, k_t) := d(\gamma, k_t) \wedge adj(\delta(k_b)) \approx root$.

(f) If $C_\gamma(u_r) \neq \emptyset$, then $d(\gamma, k_t) := d(\gamma, k_t) \wedge \beta_1(\delta(k_b)) \approx no \wedge \ldots \wedge \beta_k(\delta(k_b)) \approx no$

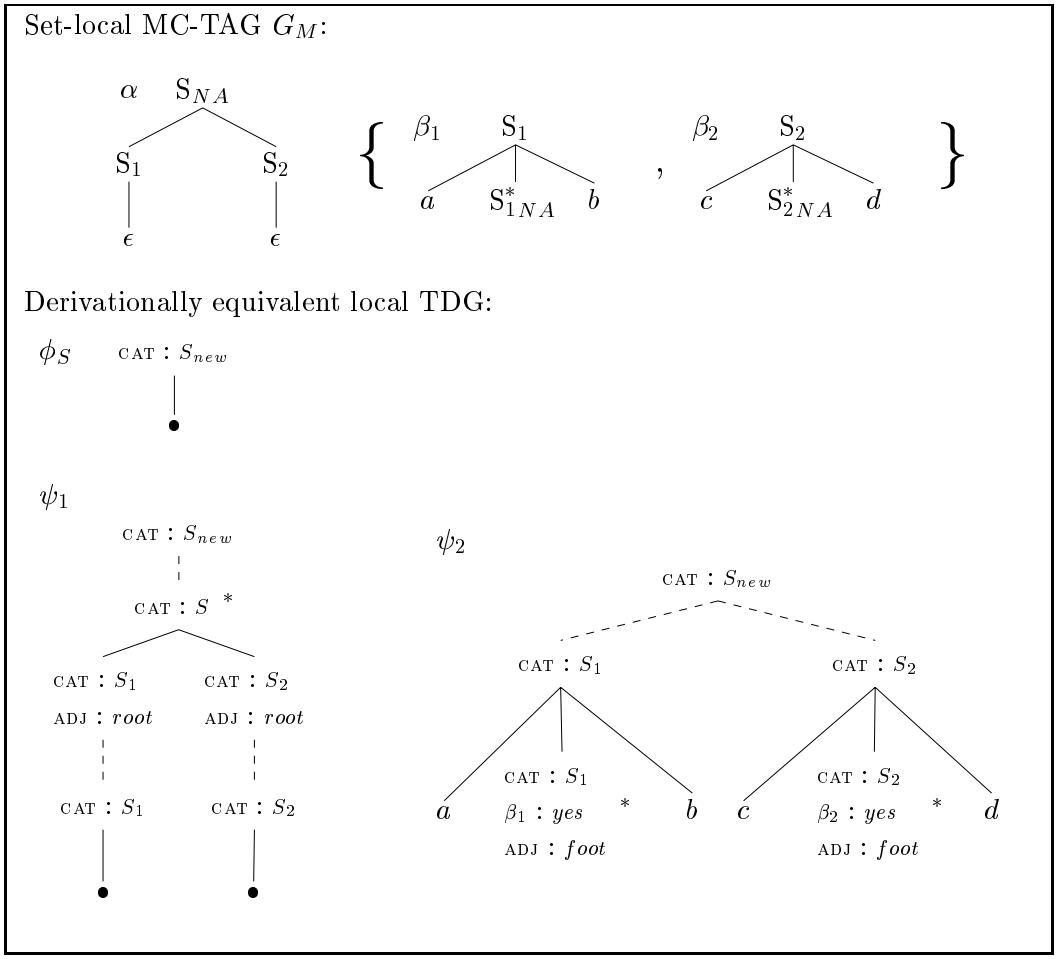


FIGURE 4.9: A SAMPLE SET-LOCAL MC-TAG AND THE CORRESPONDING LOCAL TDG

(g) If $O_\gamma(u_r) = 1$ (obligatory adjunction), then $d(\gamma, k_t) := d(\gamma, k_t) \wedge adj(\delta(k_b)) \approx foot$.

(h) If $\gamma_1, \ldots \gamma_n$ are from left to right the trees that are immediately dominated by u_r in γ, then choose new names $k_1, \ldots k_n$ and let $d(\gamma_i, k_i)$ (for $1 \leq i \leq n$) be a description of γ_i with minimal name k_i such that the sets $\{k_t, k_b\}, node(d(\gamma_1, k_1)), \ldots, node(d(\gamma_n, k_n))$ are pairwise disjoint.
$d(\gamma, k_t) := d(\gamma, k_t) \wedge k_b \lhd k_1 \wedge \ldots k_b \lhd k_n$
$\wedge k_1 \prec k_2 \wedge \ldots k_{n-1} \prec k_n$
$\wedge d(\gamma_1, k_1) \wedge \ldots \wedge d(\gamma_n, k_n)$

(i) If $\gamma \in A$ and k_{foot} is the bottom name of the foot node, then $d(\gamma, k_t) := d(\gamma, k_t) \wedge \gamma(\delta(k_{foot})) \approx yes \wedge adj(\delta(k_{foot})) \approx foot$.

Construction of a TDG $G_D = \langle A', V, T, D, \phi_S \rangle$:

1. $A' := \{cat, adj\} \cup A$, $V := N \cup \{yes, no, root, foot\} \cup \{S_{new}\}$ with $S_{new} \notin N$.
2. $\phi_S := k_1 \lhd k_2 \wedge cat(\delta(k_1)) \approx S_{new}$

3. for all $\alpha \in I$ there is a $k \in K$, a $\psi := d(\alpha, k)$ and a $k_0 \notin node(\psi)$ such that

$$\langle \psi \wedge k_0 \lhd^* k \wedge cat(\delta(k_0)) \approx S_{new}, \{k\} \rangle \in D$$

4. for all $\{\beta_1, \ldots, \beta_n\} \in \mathcal{A}$:
   there are $k_1, \ldots, k_n \in K$ and $\psi_1 := d(\beta_1, k_1), \ldots, \psi_n := d(\beta_n, k_n)$ where $\psi'_1, \ldots, \psi'_n$ are such that for all $1 \leq i \leq n$, ψ'_i can be obtained from ψ_i by removing all conjuncts with an occurrence of k_i, and $node(\psi'_1), \ldots, node(\psi'_n)$ are pairwise disjoint.
   Let k'_i be minimal in ψ'_i for $1 \leq i \leq n$, and let $\{k^f_1, \ldots, k^f_n\}$ be the bottom names of the foot nodes in $\psi_1, \ldots, \psi_n$.
   Then there is a $k_0 \notin \bigcup_{i=1}^n node(\psi'_i)$ with

$$\langle k_0 \lhd^* k'_1 \wedge \ldots \wedge k_0 \lhd^* k'_n \wedge cat(\delta(k_0)) \approx S_{new} \wedge \psi'_1 \wedge \ldots \wedge \psi'_n, \{k^f_1, \ldots, k^f_n\} \rangle \in D$$

These are all elementary descriptions in D. The result is a local TDG G_D.

To show: G_M and G_D are derivationally equivalent.
By induction on n, the following holds for all $n \geq 0$:

there is an $\alpha \in I$ and a tree γ with $\alpha \stackrel{n}{\Rightarrow} \gamma$ in G_M
iff
there is a description ϕ'_γ with $\phi_S \stackrel{n+1}{\Rightarrow} \phi'_\gamma$ in G_D such that

- there is a description $d(\gamma, k_{min})$ of γ and a $k \notin node(d(\gamma, k_{min}))$ such that: let ϕ_γ be the description that can be obtained by
  - taking $\phi := d(\gamma, k_{min}) \wedge k \lhd k_{min} \wedge cat(\delta(k)) \approx S_{new}$,
  - and then for all k_1 and k_2 where k_1 is top and k_2 bottom name of a foot node added in the course of the derivation of γ: replacing all occurrences of k_1 in ϕ by k_2, and afterwards replacing the conjunct $adj(\delta(k_2)) \approx root$ by $adj(\delta(k_2)) \approx foot$

  Then there is a function $f : node(\phi'_\gamma) \to node(\phi_\gamma)$ such that: $\hat{f}(\phi'_\gamma) \vdash \phi_\gamma$ and $\phi_\gamma \vdash \hat{f}(\phi'_\gamma)$ and for all $k \in node(\phi'_\gamma)$: $\phi'_\gamma \vdash f(k) \approx k$ and $f(k)$ has not been used to add a new equivalence in the course of the derivation of ϕ'_γ.
- for all $u_1, u_2 \in \mathcal{U}_\gamma$ and the corresponding bottom names k^f_1, k^f_2 in ϕ_γ:
  there is an elementary ψ_d in G_D with $k^f_1, k^f_2 \in node(\psi_d)$ and $\phi_S \stackrel{*}{\Rightarrow} \phi \stackrel{\psi_d}{\Rightarrow} \phi' \stackrel{*}{\Rightarrow} \phi'_\gamma$ for some ϕ and ϕ' iff either $u_1, u_2 \in \mathcal{U}_\alpha$ or $u_1, u_2 \in \bigcup_{i=1}^m \mathcal{U}_{\beta_i}$ for some $\{\beta_1, \ldots, \beta_m\} \in \mathcal{A}$.

Induction on n:

(i) Induction start: $n = 0$. Since exactly the descriptions constructed for initial trees can be added to ϕ_S, the induction claim clearly holds for $n = 0$.

(ii) Induction step $n \to n + 1$:
   suppose that $\alpha \stackrel{n}{\Rightarrow} \gamma$ in G_M and $\phi_S \stackrel{n+1}{\Rightarrow} \phi'_\gamma$ such that induction claim holds for γ and ϕ'_γ with ϕ_γ and with $f : node(\phi'_\gamma) \to node(\phi_\gamma)$.
   For all marked names k in a description ψ constructed for an auxiliary set, there is a k_p with $\psi \vdash k_p \lhd k$ (this is the case because of step (c) in the construction and because in these descriptions only bottom names of foot nodes are marked). Therefore (with *(L3)(i)*) in a derivation step where ψ is added, k becomes equivalent to a k_d in the

old description that has not been used yet to introduce a new equivalence. In other words for all $k_d \in node(\phi'_\gamma)$ that might be used for a new equivalence: $f(k_d) = k_d$ and therefore $k_d \in node(\phi_\gamma)$.

In G_M, in a further derivation step, a new auxiliary set must be added.

In G_D, in a further derivation step, a description constructed for some auxiliary set must be added because:

suppose that $\langle \psi, K_\psi \rangle$ with k_m minimal in ψ was constructed for an $\alpha \in I$
then $K_\psi = \{k\}$ for some k with $\psi \vdash_s k_m \lhd^* k$ and k is no leaf name. Consequently (because of *(L3)(ii)*): if $\phi'_\gamma \overset{\psi}{\Rightarrow} \phi_{new}$, then there must be a leaf name k_l in ϕ'_γ with $\phi_{new} \vdash k_l \approx k$. This is a contradiction since for all leaf names k' in ϕ'_γ, there is a $t \in T \cup \{\epsilon\}$ with $\phi'_\gamma \vdash \delta(k') \approx t$.

In ϕ_γ, all names k_b with $\phi_\gamma \vdash_s k_t \lhd^* k_b$ for some $k_t \in node(\phi_\gamma)$ are bottom names of some $u \in \mathcal{U}_\gamma$.

Because of the construction of the description of a tree and since when constructing ϕ_γ from ϕ'_γ only for nodes withl NA-constraints (foot nodes) equivalences between top and bottom names were aded, the following holds for γ and ϕ_γ:

(*) for all $u \in \mathcal{U}_\gamma$ and $k_t, k_b \in node(\phi_\gamma)$ such that k_t is top and k_b bottom name of u:
- $C_\gamma(u) \neq \emptyset$
  iff
  $\phi_\gamma \vdash_s k_t \lhd^* k_b$, $\phi_\gamma \vdash adj(\delta(k_t)) \approx root$, $\phi_\gamma \nvdash adj(\delta(k_b)) \approx root$ and for all $\beta \in A$: $\phi_\gamma \vdash \beta(\delta(k_b)) \approx no$ iff $\beta \notin C_\gamma(u)$.
- $O_\gamma(u) = 1$ iff $\phi_\gamma \vdash adj(\delta(k_b)) \approx foot$.
- for all $X \in N$: $\mu_\gamma(u) = X$ iff $\phi_\gamma \vdash cat(\delta(k_t)) \approx X \wedge cat(\delta(k_b)) \approx X$.

Let $\{\beta_1, \ldots, \beta_m\} \in \mathcal{A}$ be an auxiliary set and $\langle \psi', \{k_1^f, \ldots, k_m^f\} \rangle$ a corresponding elementary description in G_D with $node(\psi') \cap node(\phi'_\gamma) = \emptyset$ and $k_{\psi'}$ minimal name in ψ'.

Let $u_1, \ldots, u_m \in \mathcal{U}_\gamma$ be nodes in γ, $k_1^t, \ldots, k_m^t$ the top and $k_1^b, \ldots, k_m^b$ the bottom names in ϕ_γ of $u_1, \ldots, u_m$ respectively, and let k_γ be the minimal name in ϕ_γ.

When adding ψ' to a description derived in G_D, $k_{\psi'}$ must become equivalent to the minimal name of this description. Let $k_1, \ldots, k_m$ be the names with $\psi' \vdash_s k_{\psi'} \lhd^* k_i$ and $\psi' \vdash k_i \lhd^* k_i^f$ for $1 \leq i \leq m$.

With the induction claim for n:

There is an elementary ψ_d in G_D with $k_1^b, \ldots, k_m^b \in node(\psi_d)$ and $\phi_S \overset{*}{\Rightarrow} \phi \overset{\psi_d}{\Rightarrow} \phi' \overset{*}{\Rightarrow} \phi'_\gamma$ for some ϕ and ϕ' iff either $u_1, \ldots, u_m \in \mathcal{U}_\alpha$ or $u_1, \ldots, u_m \in \bigcup_{i=1}^{l} \mathcal{U}_{\beta_i}$ for some $\{\beta_1, \ldots, \beta_l\} \in \mathcal{A}$.

Therefore (with (*)):

$\gamma \overset{\{\beta_1,\ldots,\beta_m\}}{\Rightarrow} \gamma_{n+1}$ with $\gamma_{n+1} = \gamma[u_1, \beta_1] \ldots [u_m, \beta_2]$
iff
there is a $\phi'_{\gamma_{n+1}}$ with $\phi'_\gamma \overset{\psi'}{\Rightarrow} \phi'_{\gamma_{n+1}}$, $\phi'_{\gamma_{n+1}} \vdash k_\gamma \approx k_{\psi'} \wedge k_1^b \approx k_1^f \wedge \ldots \wedge k_m^b \approx k_m^f$, $\phi'_{\gamma_{n+1}} \vdash_s k_i^t \lhd^* k_i$ and $\phi'_{\gamma_{n+1}} \vdash cat(\delta(k_i^t)) \approx cat(\delta(k_i))$ for all $1 \leq i \leq m$.

In such a case, γ_{n+1} and $\phi'_{\gamma_{n+1}}$ clearly satisfy the conditions specified in the induction

claim.

Therefore, the induction claim holds also for $n+1$.

For G_M and G_D, the following holds:

for all $\phi \in L_D(G_D)$ and all $k_1, k_2 \in node(\phi)$: either $\phi \vdash k_1 \prec k_2$ or $\phi \vdash \neg k_1 \prec k_2$,
and for each $\gamma \in I \cup A$ and a description $d(\gamma, k)$ of γ: for all $u_1, u_2 \in \mathcal{U}_\gamma$ where k_1^b and k_2^b are the bottom names of u_1 and u_2 respectively: $d(\gamma, k) \vdash k_1^b \prec k_2^b$ iff $\langle u_1, u_2 \rangle \in \mathcal{L}_\gamma$.

Consequently (with the construction of the descriptions) for each pair γ, ϕ'_γ as in the induction claim: $O_\gamma(u) = 0$ for all $u \in \mathcal{U}_\gamma$ iff ϕ'_γ has exactly one minimal tree γ'. In this case $yield(\gamma) = yield(\gamma')$, and therefore G_M and G_D are weakly and even derivationally equivalent.

□

The next propostition states that the set of indexed string languages generated by local TDGs is a proper superset of the set of indexed string languages generated by set-local MC-TAGs:

Proposition 4.5 *There are languages L^{ind} such that there is a local TDG G with $L^{ind} = L_{ind}(G)$, and there is no MC-TAG generating L^{ind} as indexed string language.*

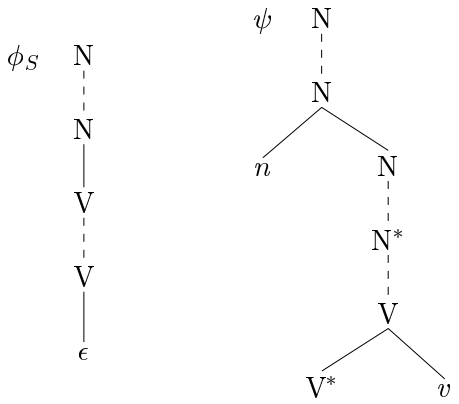
Proof $L^{ind} := SCR^{ind} = \{n^{[\sigma(1)]} \dots n^{[\sigma(m)]} v^{[1]} \dots v^{[m]} \mid m \geq 0 \text{ and } \sigma \in S_m\}$.

1. The following local TDG $G = \langle A, V, T, D, \phi_S \rangle$ (see also Fig. 3.21) generates L^{ind} as indexed string language:
   (a) $A = \{cat\}$, $V = \{N, V\}$, $T = \{n, v\}$.
   (b) Start description:

$$\begin{aligned} \phi_S \quad = \quad & k_1 \lhd^* k_2 \wedge k_2 \lhd k_3 \wedge k_3 \lhd^* k_4 \wedge k_4 \lhd k_5 \\ & \wedge cat(\delta(k_1)) \approx N \wedge cat(\delta(k_2)) \approx N \wedge cat(\delta(k_3)) \approx V \\ & \wedge cat(\delta(k_4)) \approx V \wedge \delta(k_5) \approx \epsilon \end{aligned}$$

   (c) $D = \{(\psi, K_\psi)\}$ with

$$\begin{aligned} \psi \quad = \quad & k_1 \lhd^* k_2 \wedge k_2 \lhd k_3 \wedge k_2 \lhd k_4 \wedge k_3 \prec k_4 \wedge k_4 \lhd^* k_5 \wedge k_5 \lhd^* k_6 \\ & \wedge k_6 \lhd k_7 \wedge k_6 \lhd k_8 \wedge k_7 \prec k_8 \wedge cat(\delta(k_1)) \approx N \wedge cat(\delta(k_2)) \approx N \\ & \wedge \delta(k_3) \approx n \wedge cat(\delta(k_4)) \approx N \wedge \delta(k_5)) \approx N \wedge cat(\delta(k_6)) \approx V \\ & \wedge cat(\delta(k_7)) \approx V \wedge \delta(k_8) \approx v \\ K_\psi \quad = \quad & \{k_5, k_7\} \end{aligned}$$



2. As shown by Rambow (1994a), there is no MC-TAG generating L^{ind} as indexed string language (see also Prop. 2.3 on page 38).

□

This proposition shows that, although the derivation process is still local, local TDGs are more powerful than set-local MC-TAGs. The additional power of local TDGs results from the possibility to generate underspecified representations. The locality condition restricts only the derivation of descriptions but not the way minimal trees are obtained from underspecified descriptions. In this last step, arbitrary node names can become equivalent to each other as long as the resulting description is still satisfiable.

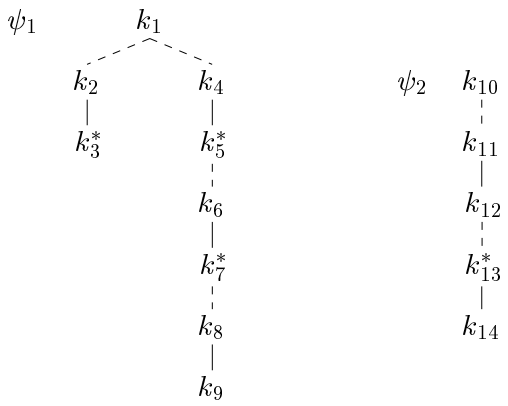
4.3 A hierarchy of local TDLs

The construction of an equivalent local TDG for a given set-local MC-TAG shows that TALs can be generated by local TDGs with at most one marked node name in each elementary description. In fact, as we will see in the proof of Prop. 4.8, a projection equivalent local TDG for a given TAG is even easier to construct than the local TDG in the proof of Prop. 4.4. The extra power of local TDGs in contrast to TAGs arises from the possibility of marking more than one node name in an elementary description. This suggests the introduction of a hierarchy of local TDGs defined with respect to the maximal number of marked node names in an elementary description. In this section such a hierarchy will be defined and we will see that context-free languages are the first class and TALs are a subset of the second class of this hierarchy. Using a pumping lemma that can be shown for the set of local TDLs of rank n, I will prove that the i-th class is as proper subset of the $(i+1)$-th class for $i > 0$.

This hierarchy of local TDGs together with the pumping lemma is also presented in Kallmeyer 1998.

4.3.1 Local TDGs of rank i

The hierarchy for local TDLs is such that the rank of the grammar depends on the marked node names. I will distinguish between marked names where in a derivation step the part of the description dominating this name can be inserted somewhere "in between" on the one hand, and on the other hand marked node names that must be identified with a leaf name.



In ψ_1, k_3 and k_5 are a-marked and k_7 is s-marked.
$m_a(\psi_1) = 2, m_s(\psi_1) = 1$.
In ψ_2, k_{12} is both, s-marked and a-marked.
$m_a(\psi_2) = 1, m_s(\psi_2) = 1$.

FIGURE 4.10: A-MARKED AND S-MARKED NODE NAMES

The first are marked names that either dominate some other marked name or that are leaf names or bottom-underspecified. The latter are marked names dominating other names but not dominating any other marked names. These two characterizations are not exclusive, i.e. there are marked names that have both properties. Since there is a similarity between foot nodes of auxiliary trees in TAGs and the first kind of marked node names, these are called *adjunction-marked (a-marked)*. For similar reasons, the second kind of marked names are called *substitution-marked (s-marked)*. We will see that a-marked names contribute more to the generative capacity of a grammar than s-marked names.

Definition 4.12 (A-marked, s-marked) *Let G be a local TDG and ψ an elementary description in G with $k \in K_\psi$. k is called*

1. *adjunction-marked (a-marked), iff it is either bottom-underspecified or a leaf name or there is a $k' \in K_\psi, k' \neq k$ with $\psi \vdash k \lhd^* k'$.*
   $m_a(\psi) := |\{k \mid k \text{ is a-marked in } \psi\}|$.
2. *substitution-marked (s-marked), iff it is no leaf name and if there is no $k' \in K_\psi, k' \neq k$ with $\psi \vdash k \lhd^* k'$.*
   $m_s(\psi) := |\{k \mid k \text{ is s-marked in } \psi\}|$.

Examples of elementary local descriptions containig both, a-marked node names and s-marked node names, are the descriptions ψ_1 and ψ_2 in Fig. 4.10. k_3 is a-marked because it is a leaf name, k_5 dominates another marked names and is therefore a-marked, and k_7 is not a leaf name and does not dominate other marked names and is consequently s-marked. k_{13} is even both, a-marked and s-marked: it is bottom-underspecified and therefore a-marked, and it is s-marked, since it is not a leaf name and does not dominate other marked names.

Roughly said, in a derivation step, for each s-marked name in the new elementary description, there is one substring added to the yield of the description, and for each a-marked name, two substrings are added. Fig. 4.11 gives an example of a derivation step showing that for bottom-underspecified names that are s-marked such as k_{15} in Fig. 4.10, even three substrings are added. In this figure, $Yield(\phi) = \{bb\}$ holds and $Yield(\phi') = \{abccba, baccab\}$. The name k_4 in the elementary ψ is both, a- and s-marked. For this name, when adding ψ, the substrings a, cc and a are added.

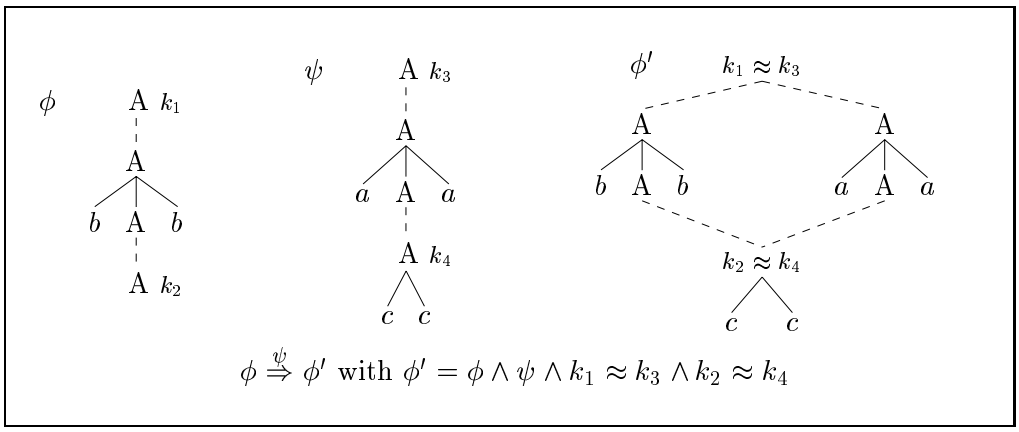


FIGURE 4.11: SAMPLE DERIVATION STEP WITH A NAME THAT IS BOTH, S-MARKED AND A-MARKED

Because of the number of new substrings, a-marked names count twice as much as s-marked names for the rank of a local TDG:

Definition 4.13 (Rank of a local TDG) *Let $i \in \mathbb{N}$.*

1. *A local TDG G is of rank i*
   iff $max\{n \mid n = 2m_a(\psi) + m_s(\psi)$ for some elementary ψ in $G\} = i$.
   TDG_i^L is the set of all local TDGs of rank i.
2. *A local TDL L is of rank i*
   iff there is a $G \in TDG_i^L$ with $L_S^l(G) = L$.
   TDL_i^L is the set of all local TDLs of rank i.

Another alternative would be a definition of the rank as the maximal number of marked node names in an elementary description. Then TAGs would exactly generate the local TDLs of rank 1 whereas with the definition of rank given here, I can only show that TALs are a subset of the local TDLs of rank 2. Equality of TALs and local TDLs is not excluded but it is at least difficult to shown. On the other hand, a definition of the rank by way of the maximal number of marked names would not show the difference between s-marked and a-marked names. Later, when proving a pumping lemma for local TDGs of rank n in 4.3.3, we will see more clearly that this distinction is important. Therefore I have chosen to define the rank of a local TDG as above.

For a given local TDG it is always possible to find a weakly equivalent local TDG with one more s-marked name per elementary description. Therefore $TDL_i^L \subseteq TDL_{i+1}^L$ holds for $i \geq 0$.

4.3.2 Relation to CFL and TAL

Clearly, the class TDL_0^L is exactly the class of all context-free languages: suppose that each elementary description contains more than one node name (otherwise it would not contribute any new information when added to a derived description). Then, because of the local derivation axiom *(L3)(ii)* (see Def. 4.8, page 120), in a derivation step where an elementary description with minimal name k is added to a description ϕ, k must be identified with a leaf name in ϕ, i.e. we only have a kind of context-free substitution operation. Therefore it is easy to construct a projection equivalent context-free grammar for each $G \in TDG_0^L$.

The other direction, i.e. the construction of a strongly equivalent local TDG without marked names for a given context-free grammar is even more straightforward. An example is shown in Fig. 4.12.

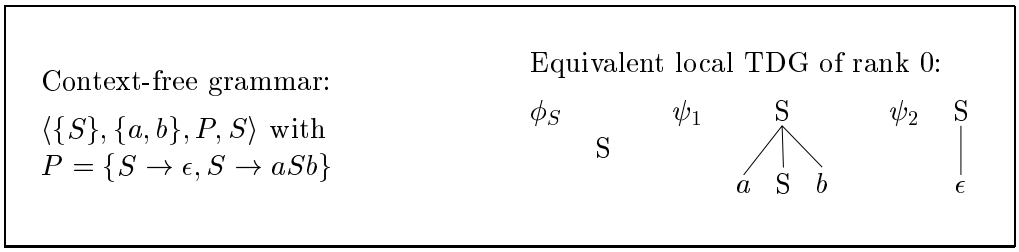


FIGURE 4.12: A CONTEXT-FREE GRAMMAR AND AN EQUIVALENT LOCAL TDG OF RANK 0 FOR $\{a^n b^n \,|\, 0 \leq n\}$

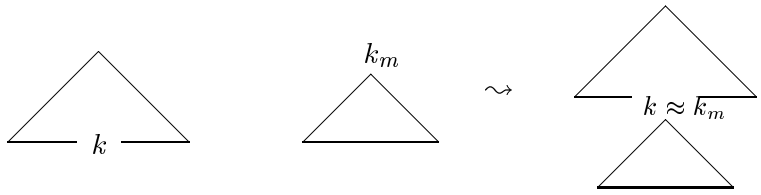
Proposition 4.6 ($CFL = TDL_0^L$) *Each language $L \in TDL_0^L$ is a context-free language and vice-versa.*

Instead of proving this, I will show a more general proposition, namely that even TDL_1^L is the class of context-free languages. This is the case because even there we only have some kind of substitution operation. Each elementary description ψ with one s-marked name consists of a minimal name strongly dominating a description ψ' such that, when adding ψ in a derivation step, ψ' must be substituted for a leaf name.

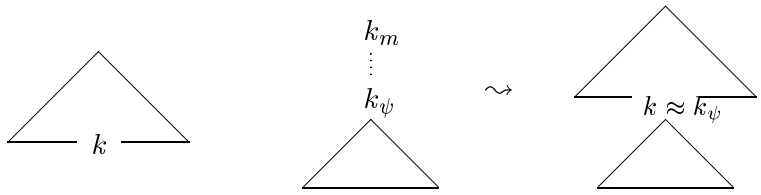
Proposition 4.7 ($CFL = TDL_1^L$) *Each language $L \in TDL_1^L$ is a context-free language and vice-versa.*

The proof idea is as follows: the construction of a TDG for a context-free grammar is as illustrated in Fig. 4.12. For the construction of an equivalent context-free grammar for a given local TDG G of rank 1, the following is done: suppose that G is label complete and that the start and elementary descriptions of G have pairwise disjoint sets of node names. A derivation step in G where an elementary description ψ with minimal name k_m is added, can have one of the following forms:

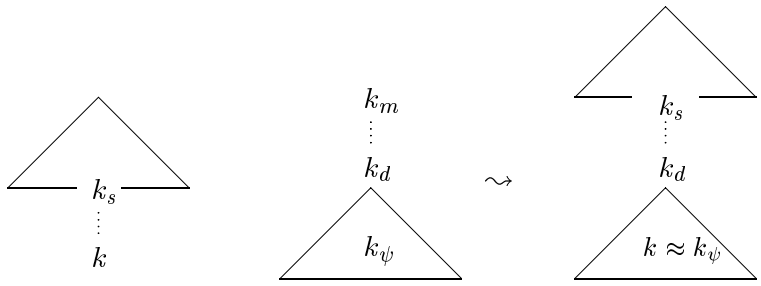
1. either $K_\psi = \emptyset$ and k_m becomes equivalent to a leaf name in the derivation description ψ_d.



   For all such leaf names k, there is a production $k \to k_m$ in the context-free grammar.
2. or $K_\psi = \{k_\psi\}$, $\psi \vdash_s k_m \lhd^* k_\psi$, and k_ψ becomes equivalent to a leaf name.



   For all such leaf names there is a production $k \to k_\psi$, if k either has a parent or is strongly dominated by a node name with the same label as k.
3. or $K_\psi = \{k_\psi\}$ and there is a name k_d with $\psi \vdash_s k_m \lhd^* k_d$ and $\psi \vdash k_d \lhd^* k_\psi$. In this case, k_ψ becomes equivalent to a leaf name k such that there is a k_s with $\psi_d \vdash k_s \lhd^* k$ and $\phi' \vdash_s k_s \lhd^* k_d$.



   For all such k and k_d where k_s and k_d have the same labels there is a production $k \to k_d$.

For the start description and elementary descriptions without marked names, the *substitution root* is defined as the minimal names, and for elementary descriptions with one marked name, the substitution root is the name strongly dominated by the minimal name. Besides the productions above, the following is needed:

- for each start or elementary description ψ with substitution root k_r and leaf names $k_1 \ldots k_n$ ordered from left to right (linear precedence is fully specified in TDGs of rank 1) there is a production $k_r \to k_1 \ldots k_n$ iff there is no strong dominance $\psi \vdash_s k_1 \lhd^* k_2$ where k_2 is neither a leaf name nor the substitution root in ψ and k_1 and k_2 have different labels.
- for each leaf name k with a terminal label t that is not strongly dominated by a node name with a different label there is a production $k \to t$.

Proof of Prop. 4.7 To show: $TDL_1^L = CFL$.

1. To show: $CFL \subseteq TDL_1^L$ (see also example in Fig. 4.12).

Let $G_c = (N, T, P, S)$ be a context-free grammar. Construction of an equivalent local TDG $G_D = (A, V, T, D, \phi_S)$ without marked names:
$A = \{cat\}$, $V = N$, $\phi_S = cat(\delta(k)) \approx S$.
For each $X \to X_1 \dots X_n \in P$, there is a $\langle \psi, \emptyset \rangle \in D$ with
$\psi = k_0 \lhd k_1 \wedge \dots \wedge k_0 \lhd k_n \wedge k_1 \prec k_2 \wedge \dots \wedge k_{n-1} \prec k_n \wedge \psi_1 \wedge \dots \wedge \psi_n$ such that for $1 \leq i \leq n$: if $X_i \in N$, then $\psi_i = cat(\delta(k_i)) \approx X_i$, else $\psi_i = \delta(k_i) \approx X_i$.
These are all elements of D.
Since there is a direct correspondence between the derivation steps in G_c and those in G_D, it is obvious that $L(G_c) = L^l_S(G_D)$ holds.

2. To show: $TDL^L_1 \subseteq CFL$.
Let $G = (A, V, T, D, \phi_S)$ be a local TDG of rank 1. Without loss of generality assume that G is label complete (the label complete TDG G' constructed for G in the proof of Lemma 4.1 is of the same rank as G). Furthermore, assume that there is no elementary description ψ with $|node(\psi)| = 1$, and that for all $\psi_1, \psi_2 \in D \cup \{\phi_S\}$: $node(\psi_1) \cap node(\psi_2) = \emptyset$.
Clearly, for each elementary ψ in G with $K_\psi \neq \emptyset$, the following holds: if k_m minimal in ψ, then (because of $|K_\psi| \leq 1$ and the 2. axiom in Def. 4.5) there is exactly one k with $\psi \vdash_s k_m \lhd^* k$ and k is no leaf name (otherwise $k \in K_\psi$ would hold and then k would be a-marked).
Construction of an equivalent context-free grammar $G_c = (N, T, P, S)$:
$D' := D \cup \{\langle \phi_S, \emptyset \rangle\}$.
First the notion of *substitution root* $r_s(\psi)$ for an elementary ψ with minimal name k_m is defined: If $K_\psi = \emptyset$, then $r_s(\psi) := k_m$. Else, $r_s(\psi) := k$ where k is the name with $\psi \vdash_s k_m \lhd^* k$.

   (a) $N := \{k \mid k \in node(\psi)$ for some ψ occuring in $D'\}$.
   (b) The start symbol S is the name k_s that is minimal in ϕ_S.
   (c) Productions: for all $(\psi, K_\psi) \in D'$:
      - if for all $k_1, k_2 \in node(\psi)$ with $\psi \vdash_s k_1 \lhd^* k_2$, k_2 is no leaf name in ψ and either k_1 not minimal or $K_\psi = \emptyset$: $\psi \not\vdash \neg k_1 \approx k_2$,
        then there is a production $r_s(\psi) \to k_1 \dots k_n$ in P where
        $k_1, \dots, k_n$ are all leaf names in ψ, and $\psi \vdash k_i \prec k_{i+1}$ for $1 \leq i < n$.
      - for all leaf names k in ψ: if there is a $t \in T \cup \{\epsilon\}$ with $\psi \vdash \delta(k) \approx t$, and if there is no k_d with $\psi \vdash_s k_d \lhd^* k$ and $\psi \vdash \neg k_d \approx k$,
        then there is a production $k \to t$ in P.
      - for all leaf names k in ψ such that there is no $t \in T \cup \{\epsilon\}$ with $\psi \vdash \delta(k) \approx t$ and for all $(\psi', K_{\psi'}) \in D$, $\psi' \neq \phi_S$ with $K_{\psi'} = \emptyset$:
        if
        1. either there is a k_p with $\psi \vdash k_p \lhd k$ or there is a k_d with $\psi \vdash_s k_d \lhd^* k$ and for all $a \in A, v \in V \cup \{\bot\}$: $\psi \vdash a(\delta(k_d)) \approx v$ iff $\psi \vdash a(\delta(k)) \approx v$, and
        2. for all $a \in A$ and $v \in V \cup \{\bot\}$: $\psi \vdash a(\delta(k)) \approx v$ iff $\psi' \vdash a(\delta(r_s(\psi'))) \approx v$,

        then there is a production $k \to r_s(\psi')$ in P.
      - for all leaf names k in ψ such that there is no $t \in T \cup \{\epsilon\}$ with $\psi \vdash \delta(k) \approx t$ and for all $(\psi', K_{\psi'}) \in D'$, with $K_{\psi'} \neq \emptyset$ and k_m minimal in ψ' such that there is a

$k' \neq k$ with $\psi \vdash k' \lhd^* k$ and $\psi \vdash a(\delta(k')) \approx v$ iff $\psi' \vdash a(\delta(k_m)) \approx v$ for all $a \in A$ and $v \in V \cup \{\bot\}$:
if

1. either: $r_s(\psi')$ marked and then:
   either there is a k_p with $\psi \vdash k_p \lhd k$ or there is a k_d with $\psi \vdash_s k_d \lhd^* k$ and for all $a \in A, v \in V \cup \{\bot\}$: $\psi \vdash a(\delta(k_d)) \approx v$ iff $\psi \vdash a(\delta(k)) \approx v$,
   or $r_s(\psi')$ is not marked, and then there is a $k_d \in node(\psi)$ with $\psi \vdash_s k_d \lhd^* k$ and $\psi \vdash a(\delta(k_d)) \approx v$ iff $\psi' \vdash a(\delta(r_s(\psi'))) \approx v$ for all $a \in A$ and $v \in V \cup \{\bot\}$, and
2. for all $a \in A$ and $v \in V \cup \{\bot\}$ and for the marked $k_{\psi'} \in K_{\psi'}$: $\psi \vdash a(\delta(k)) \approx v$ iff $\psi' \vdash a(\delta(k_{\psi'})) \approx v$,

then there is a production $k \to r_s(\psi')$ in P.

These are all productions in P.

The result is a context-free grammar.

By induction on n, the following can be easily shown:
in G_{CF} there is a derivation $S \Rightarrow w_1 \overset{2n}{\Rightarrow} w_n = x_1 \dots x_m$ such that no production $X \to \epsilon$ for some $X \in N$ has been applied and each x_i $(1 \leq i \leq n)$ is a leaf name in one of the descriptions in D'
iff
there is a derivation in G $\phi_s \overset{n}{\Rightarrow} \phi_n$ such that

(a) for all $k, k' \in node(\phi)$ where $\phi \vdash_s k \lhd^* k'$, and k' is no leaf name: $\phi \vdash \delta(k) \approx \delta(k')$,
(b) if $k_1, \dots, k_m$ are all leaf names in ϕ_n such that $\phi \vdash k_i \prec k_j$ for $i < j$, then for all i, $1 \leq i \leq m$: if ψ is the description added in one of the derivation steps with $k_i \in node(\psi)$ and if ψ' with $\psi \approx_K \psi'$ is in D' such that $\psi' = \hat{f}(\psi)$, then $x_i = f(k_i)$.

Induction start $n = 0$:

The start symbol S is the minimal name of ϕ_S. There is a production $S \to x_1 \dots x_m$ in G_{CF} iff (construction of the productions)

(a) for all $k, k' \in node(\phi_S)$ with $\phi_S \vdash_s k \lhd^* k'$ and k' is no leaf name in ϕ_S: $\phi_S \vdash k \approx k'$, and
(b) $x_1, \dots, x_m$ are all leaf names in ϕ_S with $\phi_S \vdash x_i \prec x_j$ for $i < j$.

Induction step $n \to n + 1$:

Suppose that $S \Rightarrow w_1 \overset{2n}{\Rightarrow} w_n = x_1 \dots x_m$ in G_{CF} and $\phi_s \overset{n}{\Rightarrow} \phi_n$ in G such that the induction claim holds for these two derivations.
Let ψ be a description with $node(\psi) \cap node(\phi_n) = \emptyset$ such that there is a $\psi' \in D'$ with $\psi \approx_K \psi'$ and $\psi' = \hat{f}_\psi(\psi)$. Let k_m be the minimal name in ψ and let $k_1, \dots, k_r$ be all leaf names in ψ such that $\psi \vdash k_i \prec k_j$ for $i < j$.

1. if $K_\psi = \emptyset$, then:
   there is an i $(1 \leq i \leq n))$ such that k_i is part of the elementary description ψ_d and there are productions $x_i \to f_\psi(k_m)$ and $f_\psi(k_m) \to f_\psi(k_1) \dots f_\psi(k_r)$ iff
   (a) either there is a k_p with $\psi_d \vdash k_p \lhd k_i$ (and therefore $\phi_n \vdash k_p \lhd k_i$) or there is a k_d with $\psi_d \vdash_s k_d \lhd^* k_i$ (and, since there are no a-marked names and k_i is a leaf

name in ϕ_n, $\phi_n \vdash_s k_d \lhd^* k_i$) and $\phi_n \vdash \delta(k_d) \approx \delta(k_i)$,

(b) for all $a \in A, v \in V \cup \{\bot\}$: $\phi_n \vdash a(\delta(k_i)) \approx v$ iff $\psi \vdash a(\delta(k_m)) \approx v$, and

(c) for all k, k' with $\psi \vdash_s k \lhd^* k'$ and k' is no leaf name in ψ: $\psi \vdash \delta(k) \approx \delta(k')$.

iff $\phi_n \overset{\psi}{\Rightarrow} \phi_{n+1}$ in G with $\phi_{n+1} = \phi_n \wedge \psi \wedge k_i \approx k_m$ and ϕ_{n+1} satisfies condition (a) in the induction claim.

2. if $K_\psi = \{r_s(\psi)\}$, then:
there is an i $(1 \leq i \leq n)$ such that k_i is part of the elementary description ψ_d and there are productions $x_i \to f_\psi(r_s(\psi))$ and $f_\psi(r_s(\psi)) \to f_\psi(k_1) \ldots f_\psi(k_r)$ iff

(a) there is a $k \neq k_i$, $k \in node(\psi_d)$ with $\phi_n \vdash k \lhd^* k_i$ and for all $a \in A, v \in V \cup \{\bot\}$: $\phi_n \vdash a(\delta(k)) \approx v$ iff $\psi \vdash a(\delta(k_m)) \approx v$,

(b) either there is a k_p with $\phi_n \vdash k_p \lhd k_i$ or there is a k_d with $\phi_n \vdash_s k_d \lhd^* k_i$ and $\phi_n \vdash \delta(k_d) \approx \delta(k_i)$,

(c) for all $a \in A, v \in V \cup \{\bot\}$: $\phi_n \vdash a(\delta(k_i)) \approx v$ iff $\psi \vdash a(\delta(r_s(\psi))) \approx v$, and

(d) for all k, k' with $\psi \vdash_s k \lhd^* k'$ and k' is no leaf name in ψ and $k \neq k_m$: $\psi \vdash \delta(k) \approx \delta(k')$.

iff $\phi_n \overset{\psi}{\Rightarrow} \phi_{n+1}$ in G with $\phi_{n+1} = \phi_n \wedge \psi \wedge k \approx k_m \wedge k_i \approx r_s(\psi)$ and ϕ_{n+1} satisfies condition (a) in the induction claim.

3. if $K_\psi = \{k_\psi\}$ with $k_\psi \neq r_s(\psi)$, then:
there is an i $(1 \leq i \leq n)$ such that k_i is part of the elementary description ψ_d and there are productions $x_i \to f_\psi(r_s(\psi))$ and $f_\psi(r_s(\psi)) \to f_\psi(k_1) \ldots f_\psi(k_r)$ iff

(a) there is a $k \neq k_i$, $k \in node(\psi_d)$ with $\phi_n \vdash k \lhd^* k_i$ and for all $a \in A, v \in V \cup \{\bot\}$: $\phi_n \vdash a(\delta(k)) \approx v$ iff $\psi \vdash a(\delta(k_m)) \approx v$,

(b) there is a k_d with $\phi_n \vdash_s k_d \lhd^* k_i$ and for all $a \in A, v \in V \cup \{\bot\}$: $\phi_n \vdash a(\delta(k_d)) \approx v$ iff $\psi \vdash a(\delta(r_s(\psi))) \approx v$,

(c) for all $a \in A, v \in V \cup \{\bot\}$: $\phi_n \vdash a(\delta(k_i)) \approx v$ iff $\psi \vdash a(\delta(k_\psi)) \approx v$, and

(d) for all k, k' with $\psi \vdash_s k \lhd^* k'$ and k' is no leaf name in ψ and $k \neq k_m$: $\psi \vdash \delta(k) \approx \delta(k')$.

iff $\phi_n \overset{\psi}{\Rightarrow} \phi_{n+1}$ in G with $\phi_{n+1} = \phi_n \wedge \psi \wedge k \approx k_m \wedge k_i \approx k_\psi \wedge k_d \lhd^* r_s(\psi)$ and ϕ_{n+1} satisfies condition (a) in the induction claim.

For $w_n = x_1 \ldots x_m$ and ϕ_n as in the induction claim, the following holds because of the construction of the productions with terminals or ϵ on the right side:
there are productions $x_i \to t_i$ with $t_i \in T \cup \{\epsilon\}$ for all i, $1 \leq i \leq n$ iff ϕ_n has a minimal tree B with $t_1 \ldots t_m = yield(B)$.

□

Note that since $TDL_0^L \subseteq TDL_1^L$ holds and since the local TDG constructed in the first part for a context-free grammar is of rank 0, I have also proven Prop. 4.6 by proving Prop. 4.7.

For TDL_2^L it might be expected that these languages are exactly the TALs. At least $TAL \subseteq TDL_2^L$ is easy to see because it is quite obvious how to construct an equivalent TDG for a TAG where for each foot node in the TAG there is a corresponding a-marked name in the

TDG. The construction is similar to the one in the proof of Prop. 4.9 where a derivationally equivalent local TDG was constructed for a given set-local MC-TAG. In the case of simple TAGs, the top node name of the elementary descriptions is not necessary. Examples of a local TDGs of rank 2 for a Tree Adjoining Language are shown in Fig. 4.13 and Fig. 4.14 where local TDGs for $\{a_1^k a_2^k a_3^k a_4^k \mid k \geq 0\}$ are given, and in Fig. 4.15 and 4.16 where local TDGs of rank 2 are shown for $\{ww \mid w \in \{a,b\}^*\}$.

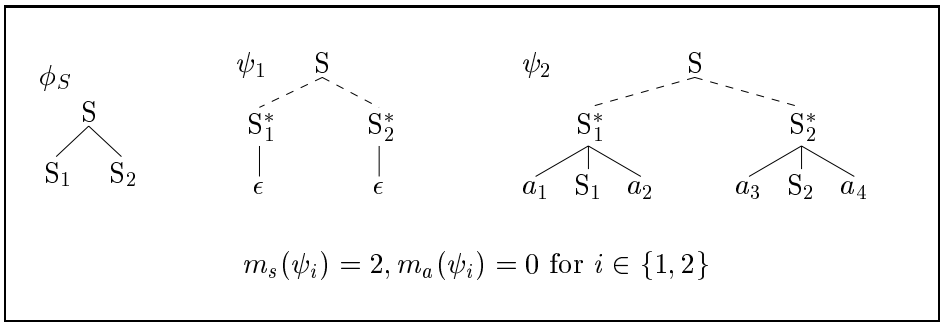


FIGURE 4.13: A LOCAL TDG FOR $\{a_1^n a_2^n a_3^n a_4^n \mid 0 \leq n\}$ WITH S-MARKED NAMES

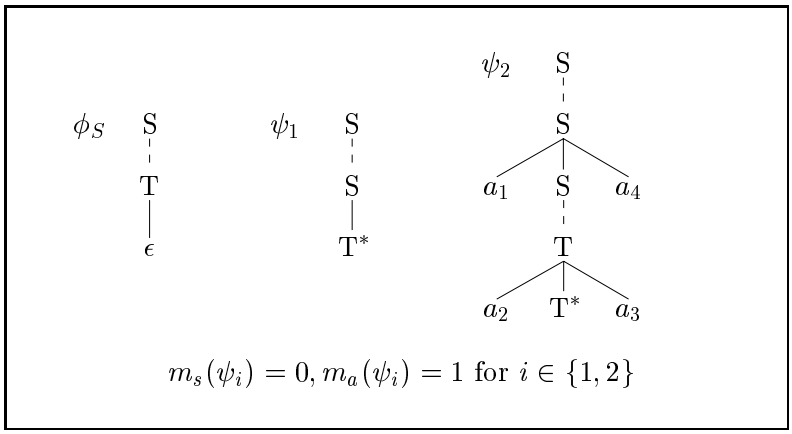


FIGURE 4.14: A LOCAL TDG FOR $\{a_1^n a_2^n a_3^n a_4^n \mid 0 \leq n\}$ WITH A-MARKED NAMES

Proposition 4.8 *($TAL \subseteq TDL_2^L$)*
For each TAG, there is a projection equivalent TDG of rank 2.

Proof Let $G_A = \langle N, T, I, A \rangle$ be a TAG. Without loss of generality suppose that adjunction is not allowed at foot nodes and that G_A is without substitution.
Construction of an equivalent local TDG $G_D = (A', V, T, D, \phi_S)$ of rank 2 (with $|m_a(G_D)| = 1, |m_s(G_D)| = 0$):
1. $A' := \{cat, adj\} \cup A$, $V := N \cup \{yes, no, root, foot\}$.

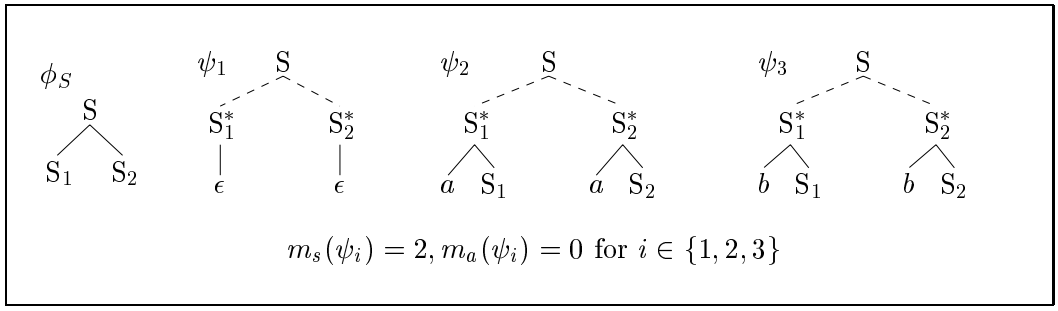


FIGURE 4.15: A LOCAL TDG FOR $\{ww \mid w \in \{a, b\}^*\}$ WITH S-MARKED NAMES

2. Elementary descriptions:
   For each elementary tree γ in G_A:
   Choose an arbitrary name k_r. Let $\psi := d(\gamma, k_r)$ be a description for γ with minimal name k_r as defined in the proof of Prop. 4.9.
   If $\gamma \in I$, then $\langle \psi, \emptyset \rangle \in D$,
   if $\gamma \in A$ and k_{foot} is the bottom name of the foot node in ψ, then $\langle \psi, \{k_{foot}\} \rangle \in D$.
   These are all elementary descriptions in G_D.
3. $\phi_S := k \approx k$.

Clearly, there is a direct correspondence between one derivation step (adjunction) in G_A and one step in G_D, and derivational and projection equivalence of G_A and G_D are easy to see.

□

Whether even $TAL = TDL_2^L$ holds, is an open question. The problem is that it is not clear how to find an equivalent TAG for a local TDG with elementary descriptions with two s-marked node names.

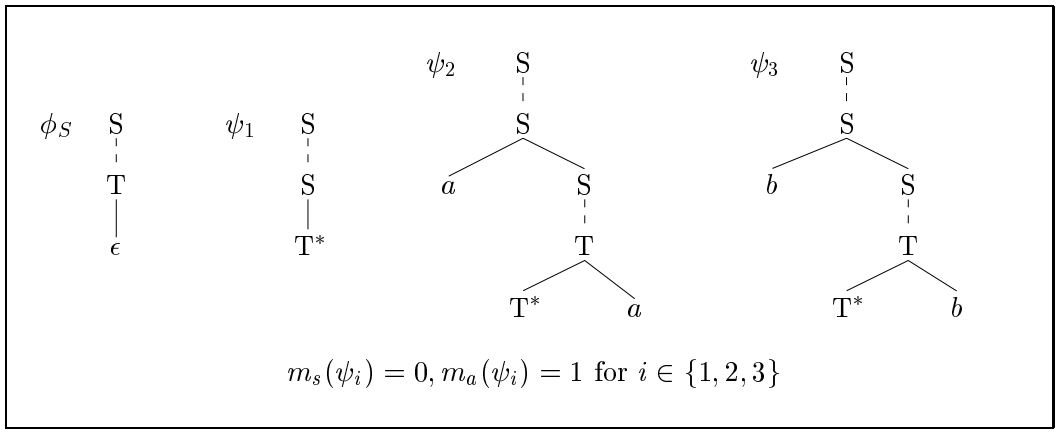


FIGURE 4.16: A LOCAL TDG FOR $\{ww \mid w \in \{a, b\}^*\}$ WITH A-MARKED NAMES

Generalizing from the examples in Fig. 4.13 to Fig. 4.16, it is clear that each language $\{a_1^k \dots a_{2n}^k \mid k \geq 0\}$ is in TDL_n^L, and that $\{w^n \mid w \in \{a,b\}^*\} \in TDL_n^L$ holds. In the next subsection, I will show that for $l > 2n$, $\{a_1^k \dots a_l^k \mid k \geq 0\} \notin TDL_n^L$ holds.

4.3.3 A pumping lemma for local TDLs of rank n

As already mentioned, it is clear that $TDL_i^L \subseteq TDL_{i+1}^L$ holds. In this subsection I will show that the set of local TDLs of rank i is even a proper subset of the set of local TDLs of rank $i+1$. First I will prove a pumping lemma for TDLs of rank i, and then, using the pumping lemma for $i = n$ I will show that $\{a_1^k \dots a_{2n+1}^k \mid 0 \leq k\} \notin TDL_n^L$ holds.

Before considering the pumping lemma itself, I will introduce the notion of *derivation grammar* for local TDGs. The derivation grammar is in principle the context-free grammar constructed in the proof of Prop. 4.2, but the terminals are removed from the productions. Only for label complete local TDGs, derivation grammars are defined.

Definition 4.14 (Derivation grammar of a local TDG) *Let G be a label complete local TDG, and let $G_c = (N, T, P, S)$ be the context-free grammar constructed as in the proof of Prop. 4.2 for G. Let a homomorphism $\varphi : (N \cup T)^* \to N^*$ be defined by $\varphi(X) := X$ for all $X \in N$ and $\varphi(t) := \epsilon$ for all $t \in T$.*
The context-free grammar $G_d(G) := (N, T, P', S)$ with

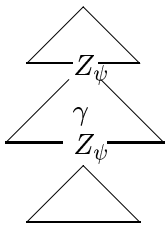
$$P' := \{A \to \alpha \mid \text{there is a } A \to \alpha' \in P \text{ such that } \varphi(\alpha') = \alpha\}$$

is called the derivation grammar of G.

There is an obvious similarity to the notion of derivation grammar for TAGs. But in the case of TAGs, one production in the derivation grammar describes all adjunctions or substitutions of elementary trees to nodes of one single elementary tree at the same time, i.e. one production may describe several derivation steps in the TAG. In the case of local TDGs, however, each production describes only one single derivation step in the TDG. Therefore there is a direct correspondence between a derivation step in the TDG and a derivation step in the derivation grammar.

The following considerations with respect to the derivation grammar are in principle the same as those leading to the pumping lemma for TALs in Vijay-Shanker 1987. Since the derivation grammar of a local TDG is context-free, the pumping lemma for context-free grammars must hold for it. This means that in a derivation tree (of the context-free derivation grammar) from a certain tree height on, there is a nonterminal Z_ψ occuring twice on a path in the tree and therefore there is a subtree γ that can be iterated. The question is what this means for the underlying local TDG and what this tells us about local TDLs.

Before adding ψ again, corresponding to the second occurrence of Z_ψ on one path, we have the following situation for a string w yielded by the old description: $w = x_{10}v_1 \dots x_{1m-1}v_m x_{1m}$ where $x_{1i} \in T^*$ for $0 \leq i \leq m$, $v_1 \dots v_m$ is the string yielded by the subdescription derived from ψ (ordered by linear precedence). As a next derivation step, ψ is added again. If the grammar is of rank n, then by adding ψ, the string w can be split by inserting at most n new strings. Before the next adding of ψ (corresponding to another iteration) takes place, these substrings will have expanded to substrings $w_1, \dots, w_n$ with $w_1 \dots w_n = v_1 \dots v_m$. These w_i may be split into several words with other words in



between but the order of the letters is as in $v_1 \dots v_m$. If this is repeated k times, $k \geq 1$, then we end up with a word contaning the letters of $x_1 := x_{10} \dots x_{1m}$ and k occurrences of all symbols of $w_1 \dots w_n$ that are for each of these occurrences (from left to right) ordered as in $w_1 \dots w_n$. In the last steps (after the iterations of the tree γ), the symbols of some string $x_2 \in T^*$ are added.

To formalize the pumping lemma, the following two definitions are useful. The *remove set* $\rho(x, w)$ for some $x, w \in T^*$ is the set of words that can be obtained by removing from w the letters of x:

Definition 4.15 (Remove set $\rho(x, w)$) *Let T be an alphabet, $x, w \in T^*$. The remove set $\rho(x, w) \subseteq T^*$ is recursively defined:*
If $x = \epsilon$, then $\rho(x, w) := \{w\}$.
Else $x = x_1 x'$ with $x_1 \in T, x' \in T^*$. Then:

1. *if $w = \epsilon$, then $\rho(x, w) := \emptyset$.*
2. *if $w = vw'$ for some $v \in T$ and $v \neq x_1$, then $\rho(x, w) := \{v\hat{w}' \mid \hat{w}' \in \rho(x, w')\}$.*
3. *if $w = vw'$ with $v = x_1$, then $\rho(x, w) := \{v\hat{w}' \mid \hat{w}' \in \rho(x, w')\} \cup \rho(x', w')$.*

This means that when reading w from left to right, first one occurrence of the first letter of x, then one occurrence of the second letter, etc. are removed. Of course, there may be more than one word in a remove set, e.g. $\rho(ab, adbeab) = \{adbe, dbea, deab\}$.

Related to this definition is the following definition of a *k-partition* of a word w for a $k \in \mathbb{N}$, $k \geq 0$. A k-partition is a vector of k nonempty words such that the original word w can be obtained by taking from left to right each time the left symbol of one of the words in the partition.

Definition 4.16 (k-partition of a word) *Let T be an alphabet, $k > 0$ and $w \in T^*$ with $|w| \geq k$.*
$P_k^w = \langle w_1, \dots, w_k \rangle$ with $w_i \in T^*$ for $1 \leq i \leq k$ is a k-partition of w iff the following holds:

- *If $k = 1$, then $P_k^w = \langle w \rangle$.*
- *If $k > 1$, then there is a $\hat{w} \in \rho(w_1, w)$ such that $\langle w_2, \dots, w_k \rangle$ is a $(k-1)$-partition of $\hat{w}$.*

$\langle abc, aad, cd \rangle$ for example is a 3-partition of *aaacbcdd*.

The notions of remove set and k-partition facilitate the formalization of the pumping lemma.

Proposition 4.9 (Pumping lemma for local TDLs of rank n) *For each $n \geq 1$ and each local TDG G of rank n there is a constant c_G such that for all $w \in L_S^l(G)$ with $|w| > c_G$:*

There are $x_1, x_2, w_1, \ldots, w_n \in T^*$ *with* $w_1 \ldots w_n \neq \epsilon$ *such that* $\langle x_1, x_2, w_1 \ldots w_n \rangle$ *is a 3-partition of* w, *and for each* $k \geq 1$ *there is a* $w^{(k)} \in L_S^l(G)$ *and a 3-partition* $\langle x_1, x_2, \hat{w}^{(k)} \rangle$ *of* $w^{(k)}$ *with*

(i) $\hat{w}^{(k)} = u_0 w_1 u_1 \ldots u_{n-1} w_n u_n$ *for some* $u_0, \ldots, u_n \in T^*$, *and*

(ii) there is a k-paritition $P_k = \langle \hat{w}_1, \ldots, \hat{w}_k \rangle$ *of* $\hat{w}^{(k)}$ *such that* $\hat{w}_i = w_1 \ldots w_n$ *for all* i, $1 \leq i \leq k$.

Note that $w_1, \ldots, w_n$ need not necessarily be substrings of w.

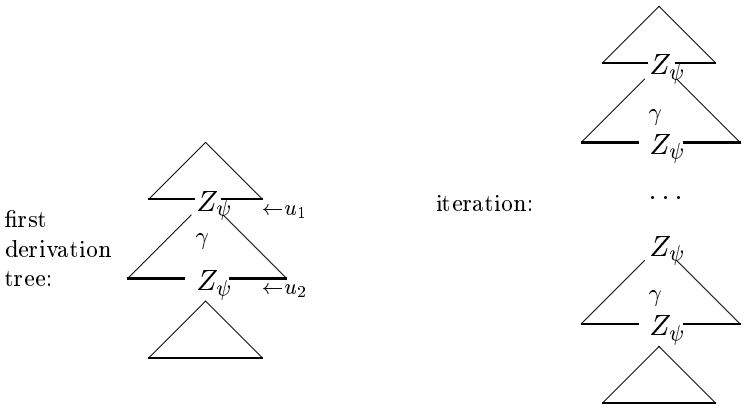
Roughly said, according to the pumping lemma, for each word w in the string language of a local TDG of rank n with a length greater than some constant c_G, the following holds: after removing the letters of some words x_1 and x_2 from w, the resulting word has the form $w_1 \ldots w_n$. Then for each k there is a word $w^{(k)}$ in the language containing also the letters of x_1 and x_2, such that: if these letters are removed from $w^{(k)}$, the result $\hat{w}^{(k)}$ is a word that can be obtained by taking k occurrences of $w_1 \ldots w_n$ and then, starting with ϵ, taking (in arbitrary order) always the left letter of one of these k words as the next letter in $\hat{w}^{(k)}$. Furthermore, $\hat{w}^{(k)}$ still contains as substrings one occurrence of each of the words $w_1, \ldots, w_n$ (in this order).

The lemma for rank n holds for example for $L_{2n} := \{a_1^k \ldots a_{2n}^k \mid 0 \leq k\}$ with:

- $c_G = 2n - 1$,
- $x_1 = x_2 = \epsilon$ and
- if $w = a_1^k \ldots a_{2n}^k$, then $w_i = a_{2i-1}^k a_{2i}^k$ for $1 \leq i\}leqn$.

Proof of the pumping lemma Let $G = \langle A, V, T, D, \phi_S \rangle$ be a local TDG of rank r_G. Without loss of generality let G be label complete. Furthermore, suppose that for all elementary ψ in G, $|Yield(\psi)| = 1$ holds (i.e. linear precedence is fully specified for the leaf names).
Let $G_c = (N, T, P, S) := G_d(G)$ be the derivation grammar of G.
$c_D := max\{|w_\psi| \mid \{w_\psi\} = Yield(\psi)$ for some ψ elementary in $G\}$

To show: With $c_G := c_D \sum_{i=0}^{|N|} 2^i$ the pumping lemma for r_G holds for G.



Let $w \in L_S(G)$ with $|w| > c_G$. In the derivation tree in G_c corresponding to the derivation of the description with a minimal tree yielding w in G, there must be at least one path from

the root to a leaf with two different nodes u_1 and u_2, u_1 dominating u_2, such that u_1 and u_2 have the same label $Z_\psi \in N$, because:

if this was not the case, each path in the tree would contain at most $|N|+1$ nodes. Since each internal node in the tree, apart from those that have a daughter labelled by ϵ, has two daughters, there would be then at most $\sum_{i=0}^{|N|-1} 2^i$ nodes representing a derivation step in G. This means that the derivation of a description ϕ with a minimal tree yielding w would be at most of length $\sum_{i=o}^{|N|-1} 2^i$, and therefore, including ϕ_S, we have in ϕ at most $1 + \sum_{i=0}^{|N|-1} 2^i \leq \sum_{i=0}^{|N|} 2^i$ descriptions that are elementary or start descriptions. Then the string w (yielded by a minimal tree of ϕ) would have a maximal length of $l_{max} := c_D + c_D \sum_{i=0}^{|N|-1} 2^i$ with $l_{max} \leq c_G$.
Contradiction to $l_{max} \geq |w| > c_G$.

Furthermore, u_1 and u_2 as above can be found such that even the following holds: if the label Z_ψ is a state of the elementary ψ, then for all $k \in node(\psi)$, $\neg equ(k)$ is a conjunct in Z_ψ (i.e. this corresponds to a derivation step where ψ is added). This is the case because in each production, the state Z_ψ on the left side and the Z'_ψ on the right side for the same ψ differ at least in so far as there is one $k \in node(\psi)$, such that $\neg equ(k)$ occurs in Z_ψ and $equ(k)$ in Z'_ψ. This means that none of the states for ψ is repeated unless there is some new Z_ψ with $Z_\psi \models \neg equ(k)$ for all $k \in node(\psi)$.
Then, in the derivation tree, the subtree γ of nodes dominated by u_1 without being dominated by u_2 can be iterated as shown in the figure.

For the corresponding descriptions derived in G, the following holds:
Suppose that ψ is elementary in G with $r_\psi := 2m_a(\psi) + m_s(\psi)$ and $Yield(\psi) = \{w_\psi\}$. Without loss of generality, $w_\psi \neq \epsilon$ holds.
Let $\phi_S \overset{*}{\Rightarrow} \phi_1 \overset{\psi}{\Rightarrow} \phi_2 \overset{*}{\Rightarrow} \phi_3 \overset{\psi}{\Rightarrow} \phi_4$ be a derivation such that γ is the derivation tree corresponding to $\phi_2 \overset{*}{\Rightarrow} \phi_3 \overset{\psi}{\Rightarrow} \phi_4$ (this means that none of the derivation descriptions used in the steps $\phi_2 \overset{*}{\Rightarrow} \phi_3$ has been added in the course of the derivation of ϕ_1).
To show:

(*) For all $w_{\phi_3} \in Yield(\phi_3)$: For each a-marked $k \in K_\psi$, there are $w_1(k), w_2(k) \in T^*$ and for each s-marked $k \in K_\psi$, there is one $w(k) \in T^*$ such that: there is one order $w_1, \ldots, w_{r_\psi}$ of these r_ψ words and there is a $x_1 \in Yield(\phi_1)$ such that:

1. $w_1 \ldots w_{r_\psi} \neq \epsilon$.
2. There is a $\hat{w} \in \rho(x_1, w_{\phi_3})$ with $\hat{w} = w_1 \ldots w_{r_\psi}$.
3. For all $n \geq 2$: if $\phi_S \overset{*}{\Rightarrow} \phi_1 \overset{\psi}{\Rightarrow} \phi_2 \overset{*}{\Rightarrow} \phi_3 \overset{\psi}{\Rightarrow} \phi_4 \overset{*}{\Rightarrow} \ldots \overset{*}{\Rightarrow} \phi_{2n-1} \overset{\psi}{\Rightarrow} \phi_{2n}$ where γ is the derivation tree for each part $\phi_{2i} \overset{*}{\Rightarrow} \phi_{2i+1} \overset{\psi}{\Rightarrow} \phi_{2i+2}$ for $1 < i < n$, then:
   For each $w_{\phi_{2i+1}} \in Yield(\phi_{2i+1})$ $(2 < i < n)$:
   (i) There are $u_0, \ldots, u_{r_\psi} \in T^*$ with $u_0 w_1 u_1 \ldots u_{r_\psi - 1} w_{r_\psi} u_{r_\psi} = w_{\phi_{2i+1}}$.
   (ii) There is a $\hat{w}_{\phi_{2i+1}} \in \rho(x_1, w_{\phi_{2i+1}})$ such that there is a i-partition $\langle \hat{w}_1, \ldots, \hat{w}_i \rangle$ of $\hat{w}_{\phi_{2i+1}}$ with $\hat{w}_j = w_1 \ldots w_{r_\psi}$ for all $1 \leq j \leq i$.

Proof of (*): Let $w_{\phi_3} \in Yield(\phi_3)$ and $k_1^{\phi_3}, \ldots, k_{m_3}^{\phi_3}$ be all leaf names in ϕ_3 with $\phi_3 \not\vdash k_j^{\phi_3} \prec k_i^{\phi_3}$ for $i < j$ and $\phi_3 \vdash \delta(k_i^{\phi_3}) \approx t_i^{\phi_3}$ for $1 \leq i \leq m_3$ and $w_{\phi_3} = t_1^{\phi_3} \ldots t_{m_3}^{\phi_3}$.

Definition of a homomorphism $\varphi_1 : \{k_1^{\phi_3}, \ldots, k_{m_3}^{\phi_3}\}^* \to T^*$: for all i, $1 \leq i \leq m_3$, if $k_i^{\phi_3} \in node(\phi_1)$, then $\varphi_1(k_i^{\phi_3}) = \epsilon$, else $\varphi_1(k_i^{\phi_3}) = t_i^{\phi_3}$.

and a homomorphism $\varphi_2 : \{k_1^{\phi_3}, \ldots, k_{m_3}^{\phi_3}\}^* \to T^*$: for all i, $1 \leq i \leq m_3$, if $k_i^{\phi_3} \in node(\phi_1)$, then $\varphi_2(k_i^{\phi_3}) = t_i^{\phi_3}$, else $\varphi_2(k_i^{\phi_3}) = \epsilon$.

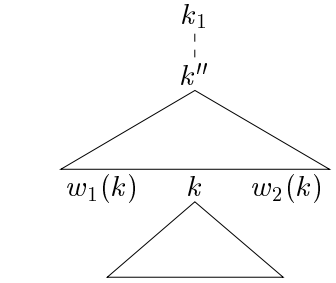
$x_1 := \varphi_2(k_1^{\phi_3} \ldots k_{m_3}^{\phi_3})$.

First, the $w_1(k)$ and $w_2(k)$ for a-marked and the $w(k)$ for s-marked names k are specified:

(i) For each a-marked $k \in node(\psi)$: suppose that $\phi_2 \vdash k_2 \approx k$ for $k_2 \in node(\phi_1)$ and that k' is the next minimal or marked name dominating k in ψ.

(a) If (k', k) is no underspecification pair, then there is a k_1 with $\phi_1 \vdash_s k_1 \lhd^* k_2$ and a k'' with $\psi \vdash_s k' \lhd^* k''$, such that $\phi_2 \vdash k_2 \approx k$, and $\phi_2 \vdash_s k_1 \lhd^* k''$.

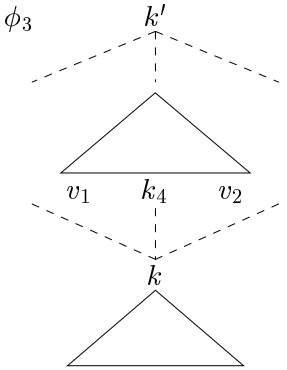
k a-marked:



Let i_1, j_1 be the smallest and i_2, j_2 the greatest numbers with $1 \leq i_1 \leq i_2 \leq j_1 \leq j_2$ and
for all i, $i_1 \leq i \leq i_2$: $\phi_3 \vdash k_1 \lhd^* k_i^{\phi_3} \wedge k_i^{\phi_3} \prec k$, and
for all j, $j_1 \leq j \leq j_2$: $\phi_3 \vdash k_1 \lhd^* k_j^{\phi_3} \wedge k \prec k_j^{\phi_3}$,
then $w_1(k) := t_{i_1}^{\phi_3} \ldots t_{i_2}^{\phi_3}$ and $w_2(k) := t_{j_1}^{\phi_3} \ldots t_{j_2}^{\phi_3}$.
Clearly, there are $x, y, z \in T^*$ with $w_{\phi_3} = xw_1(k)yw_2(k)z$.

(b) If (k', k) is an underspecification pair, then:

k a-marked:



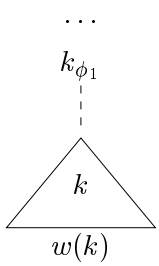
v_1 and v_2 are substrings of $w_1(k)$ and $w_2(k)$

Let i_1, j_1 be the smallest and i_2, j_2 the greatest numbers with $1 \leq i_1 \leq i_2 \leq j_1 \leq j_2$ and
for all i, $i_1 \leq i \leq i_2$: $\phi_3 \vdash k' \lhd^* k_i^{\phi_3} \wedge k_i^{\phi_3} \prec k$, and
for all j, $j_1 \leq j \leq j_2$: $\phi_3 \vdash k' \lhd^* k_j^{\phi_3} \wedge k \prec k_j^{\phi_3}$,

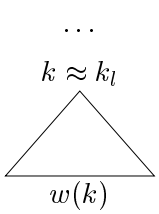
then $w_1(k) := \varphi(k_{i_1}^{\phi_3}) \dots \varphi(k_{i_2}^{\phi_3})$ and $w_2(k) := \varphi(k_{j_1}^{\phi_3}) \dots \varphi(k_{j_2}^{\phi_3})$.
Here, $w_1(k)$ and $w_2(k)$ are not necessarily substrings of w_{ϕ_3}, but they are substrings of $\varphi(k_1^{\phi_3} \dots k_{m_3}^{\phi_3}) \in \rho(x_1, w_{\phi_3})$.

(ii) For each s-marked $k \in node(\psi)$ with k' next marked or minimal name dominating k: there is a leaf name k_l in ϕ_1 with $\phi_2 \vdash k_l \approx k$. Then

k s-marked: either



or



(a) either (k', k) is no underspecification pair, and there is a k_{ϕ_1} with $\phi_1 \vdash_s k_{\phi_1} \lhd^* k_l$.
Then: if i_1 is the smallest and i_2 the greatest number with $1 \leq i_1 \leq i_2 \leq m_3$ and $\phi_3 \vdash k_{\phi_1} \lhd^* k_i^{\phi_3}$ for all i, $i_1 \leq i \leq i_2$, then $w(k) := t_{i_1}^{\phi_3} \dots t_{i_2}^{\phi_3}$.

(b) or: there is either a k_{ϕ_1} with $\phi_1 \vdash k_{\phi_1} \lhd k_l$ or (k', k) is an underspecification pair.
Then: if i_1 is the smallest and i_2 the greatest number with $1 \leq i_1 \leq i_2 \leq m_3$ and $\phi_3 \vdash k_l \lhd^* k_i^{\phi_3}$ for all i, $i_1 \leq i \leq i_2$, then $w(k) := t_{i_1}^{\phi_3} \dots t_{i_2}^{\phi_3}$.

Clearly, in both cases, there are $x, y \in T^*$ such that $w_{\phi_3} = xw(k)y$.

Now, the claims 1., 2. and 3. will be shown:

Suppose that $w_1, \dots, w_{r_\psi}$ are the ordered words obtained for the marked names. Then:

1. $w_1 \dots w_{r_\psi} \neq \epsilon$ because $w_\psi \neq \epsilon$.
2. $w_1 \dots w_{r_\psi} = \varphi_1(k_1^{\phi_3} \dots k_{m_3}^{\phi_3}) \in \rho(x_1, w_{\phi_3})$. This is the case because for each i ($1 \leq i \leq m_3$): $k_i^{\phi_3} \notin node(\phi_1)$ iff
   - either there are marked $k \in K_\psi$ and $k_1 \in node(\phi_1)$ as described in (i)(a) such that either $\phi_3 \vdash k_1 \lhd^* k_i^{\phi_3} \wedge k_i^{\phi_3} \prec k$ or $\phi_3 \vdash k_1 \lhd^* k_i^{\phi_3} \wedge k \prec k_i^{\phi_3}$
   - or there are marked $k', k \in K_\psi$ as described in (i)(b) such that either $\phi_3 \vdash k' \lhd^* k_i^{\phi_3} \wedge k_i^{\phi_3} \prec k$ or $\phi_3 \vdash k' \lhd^* k_i^{\phi_3} \wedge k \prec k_i^{\phi_3}$
   - or there is a k_{ϕ_1} as described in (ii)(a) with $\phi_3 \vdash k_{\phi_1} \lhd^* k_i^{\phi_3}$.
   - or there is a k_l as described in (ii) with $\phi_3 \vdash k_l \lhd^* k_i^{\phi_3}$.
3. When pumping $n-1$ times, each time again $w_1, \dots w_{r_\psi}$ are inserted in this order, and this time, in the case (i)(b) it is possible to choose such a dominance relation for the minimal tree that all $w_1, \dots, w_{r_\psi}$ are really substrings of the new word.
   Therefore 3.(i) and (ii) hold.

Now that (*) is shown, it is only necessary to look at the part $\phi_{2n} \stackrel{*}{\Rightarrow} \phi$ of the derivation with the tree with root u_2 as derivation tree. In these steps, only finitely many terminals are added. This means that there is a $x_2 \in T^*$ such that there is a $w^{(n)} \in Yield(\phi)$ and then a $v \in \rho(x_2, w^{(n)})$ and a $\hat{w}^{(n)} \in \rho(x_1, v)$ with:

(i) $\hat{w}^{(n)} = u_0 w_1 u_1 \dots u_{r_\psi - 1} w_{r_\psi} u_{r_\psi}$ for some $u_0, \dots, u_{r_\psi} \in T^*$, and

(ii) there is a n-paritition $P_n = \langle \hat{w}_1, \dots, \hat{w}_n \rangle$ of $\hat{w}^{(n)}$ such that for all i, $1 \leq i \leq n$: $\hat{w}_i = w_1 \dots w_{r_\psi}$.

Since $r_\psi \leq r_G$, the pumping lemma for r_G holds for G with the constant c_G.

□

The fact that a-marked names introduce two substrings whereas s-marked names introduce only one explains why a-marked names contribute twice as much to the generative capacity as s-marked names.

With the pumping lemma, the following can be shown:

Proposition 4.10 *For each i, $1 \leq i$, $TDL_i^L \overset{\subset}{\neq} TDL_{i+1}^L$ holds.*

Proof It is sufficient to show that $TDL_i^L \neq TDL_{i+1}^L$ holds.
I consider the languages $L_n := \{a_1^k \dots a_n^k \mid k \geq 0\}$.
As we have already seen, $L_i \in TDL_n^L$ holds for $i \leq 2n$.
To show: $L_{2n+1} \notin TDL_n^L$
Proof by contraposition: suppose that $L_{2n+1} \in TDL_n^L$.
Then the pumping lemma for n must be fulfilled by L_{2n+1}.
Let c_n be the corresponding constant, $w \in L_{2n+1}$ with $|w| \geq c_n$.
$w = a_1^l \dots a_{2n+1}^l$ for some $l \geq 1$. Since x_1 and x_2 in the pumping lemma are not iterated, $w_1 \dots w_n = a_1^m \dots a_{2n+1}^m$ for some $m \geq 1$ must hold.
There must be at least one i, $1 \leq i \leq n$ such that there are at least three different symbols occuring in w_i. Suppose $w_i = y_1 a_{j-1}^{m'} a_j^m a_{j+1}^{m''} y_2$ for some $y_1, y_2 \in T^*$ and $1 \leq m', m'', \leq m$.
For each $1 \leq k$ and each $w^{(k)}$, $\hat{w}^{(k)} := a_1^{mk} \dots a_{2n+1}^{mk}$ holds.
But, because of *(i)* in Prop. 4.9, $w_i = y_1 a_{j-1}^{m'} a_j^m a_{j+1}^{m''} y_2$ must be a substring of each $\hat{w}^{(k)}$.
Since this is only the case for $k = 1$, this is a contradiction.
Therefore, $L_{2n+1} \notin TDL_n^L$ holds.

□

With this proposition, it has been shown that the hierarchy of local TDGs defined in this section is a well defined hierarchy in the sense that there is a real progression of classes of languages.

With the pumping lemma, it is not possible to show $L_{n+1}^{copy} := \{w^{n+1} \mid w \in T^*\} \notin TDL_n^L$ because for each $n \in \mathbb{N}$, the pumping lemma for TDLs of rank n holds for all L_i^{copy}, $0 \leq i$. Here it might be interesting to look at the closure properties of local TDGs. Perhaps then $L_{n+1}^{copy} \notin TDL_n^L$ might be shown by applying the pumping lemma to the intersection of L_{n+1}^{copy} with some other language. As a consequence, one would know that local TDGs of rank n for a specific $n \in \mathbb{N}$ allow only limited cross-serial dependencies, which is one of the conditions for mild context-sensitivity. I leave this issue for further research.

4.4 Synchronous local TDGs

In this section I will deal with the definition and the formal properties of *synchronous local Tree Description Grammars*. Similar to synchronous TAGs (see Chapter 2, 2.4.2), a pair of synchronous local TDGs consists of two local TDGs that are related by a synchronization relation. Derivation in the two TDGs takes place simultaneously and is controlled by the synchronization relation. The use of synchronous grammar formalisms is motivated by the desire to describe two languages that are closely related to each other but that do not have the same structures. This is for example the case in Machine Translation or for the relation between syntax and semantics. Semantics clearly depends on the syntax but the way a semantic expression is built does not correspond exactly to the syntactic structure. Therefore Shieber and Schabes (1990) propose to use synchronous TAGs to describe the syntax-semantics interface. In Chapter 5, I will show that synchronous local TDGs might be used for a syntax-semantics interface and that they are more adequate than synchronous TAGs.

I will follow the idea of synchronous TAGs by defining synchronous local TDGs and I will show that, in contrast to TAGs, the synchronization of a local TDG with a second local TDG does not increase the generative capacity.

4.4.1 Synchronization

The idea of synchronous local TDGs is that two grammars are related by a synchronization relation and the derivation in the two grammars is done in parallel.

In this subsection, first the synchronization of two local TDGs is defined and then the derivation mode for synchronous local TDGs is introduced.

Synchronous local TDGs are ordered pairs of local TDGs with an additional synchronization relation between the elementary descriptions of the two grammars. This relation is such that for each elementary description ψ_1 in the first local TDG G_1 there is one elementary description ψ_2 in the second local TDG G_2 related to ψ_1. For each pair $\langle \psi_1, \psi_2 \rangle$ in the synchronization relation, there is an additional relation between $node(\psi_1)$ and $node(\psi_2)$ specifying which node names are treated in parallel in a synchronous derivation step.

The objects derived in synchronous TDGs are triples consisting of one description derived in G_1, one description derived in G_2, and a relation between the node names of these descriptions. Such triples are called *configurations*.

Definition 4.17 (Configuration) *A triple $\langle \phi_1, \phi_2, \sigma \rangle$ is a configuration iff ϕ_1 and ϕ_2 are descriptions and $\sigma \in node(\phi_1) \times node(\phi_2)$.*

Synchronization is not universally allowed for marked names but only for s-marked names. The underlying idea is that the synchronization mechanism is used to express dependencies between argument slots (leaf names) and between arguments (minimal names of descriptions ψ with $K_\psi = \emptyset$ or s-marked names). In Chapter 5 we will see that in the case of the syntax-semantics interface presented there, it is in fact sufficient to control only substitution-like new equivalences via synchronization. Adjunction-like new equivalences are controlled only by the properties of local derivations. Therefore the synchronization relation between two sets of node names is defined in such a way that marked names can occur in the relation

only if they are not a-marked and if they cannot (in some further derivation step) become equivalent to a bottom underspecified name.

This restriction is also motivated by the fact that it enables the proof given in the next subsection that synchronization does not increase the generative capacity of local TDGs.

Definition 4.18 (Synchronous local TDGs) *A quadruple $G_s = \langle G_1, G_2, \Sigma, \kappa_s \rangle$ is a pair of synchronous local TDGs iff*

1. *$G_1 = \langle A_1, V_1, T_1, D_1, \phi_1^s \rangle$ and $G_2 = \langle A_2, V_2, T_2, D_2, \phi_2^s \rangle$ are local TDGs.*
2. *$\Sigma \subseteq D_1 \times D_2 \times (K \times K)$ such that*
   - *(i) for all $\psi_1 \in D_1$, there is a ψ_2 and a σ with $\langle \psi_1, \psi_2, \sigma \rangle \in \Sigma$.*
   - *(ii) for all $\langle \psi_1, \psi_2, \sigma \rangle \in \Sigma$: $\sigma \in node(\psi_1) \times node(\psi_2)$ and for all $\langle k_1, k_2 \rangle \in \sigma$ and all $i \in \{1, 2\}$:*
     if $k_i \in M_{\psi_i}$, then
     - *k_i is no leaf name in ψ_i,*
     - *there is no marked $k_m \neq k_i$ in ψ with $\psi_i \vdash k_i \lhd^* k_m$,*
     - *and there are no k, k', k'' with $\psi_i \vdash_s k \lhd^* k'$, $\psi_i \vdash_s k'' \lhd^* k_i$, $\psi_i \vdash k \lhd^* k'' \wedge k' \lhd^* k_i$ and $\psi_i \nvdash \neg\delta(k) \approx \delta(k_i)$.*

   Each element in Σ is called an elementary configuration of G_s.
3. *$\kappa_s = \langle \phi_1^s, \phi_2^s, \sigma_s \rangle$ is a configuration, the start configuration.*

The idea of the derivation definition for synchronous local TDGs is similar to the one of synchronous TAGs. Derivation takes place simultanously in both TDGs. In one parallel derivation step the new elementary descriptions added in this step must be related by the synchronization relation Σ. Furthermore the derivation descriptions of this step must be such that they have been added previously in the same derivation step. This means that a derivation step depends not only on the history of one single local TDG derivation but on a global derivation history that specifies which descriptions were added in parallel. In this respect, synchronous local TDGs differ from synchronous TAGs. In a derivation step in synchronous TAGs, the two nodes that are replaced by new elementary trees (by adjunction or substitution) need not be part of two elementary trees that were added in the same derivation step.

With respect to new node name equivalences, the synchronization signifies the following: if there is a new equivalence in one of the TDGs and if there are related node names in the other TDG, then these node names also must become equivalent.

For the definition of the derivation for TDGs, an equivalence relation $\approx_K$ was needed in order to make sure that in a derivation step descriptions with disjoint sets of node names can be chosen. Similarly, in a derivation step in a pair of synchronous local TDGs, the two old configurations must be such that their first descriptions have disjoint sets of node names, and their second descriptions also have disjoint sets of node names. In order to choose an appropriate new configuration in a derivation step, the equivalence relation $\approx_K$ for descriptions (see Def. 3.12, p. 74) is extended to configurations:

Definition 4.19 (K-equivalence of configurations) *Let $\langle \phi_1, \phi_2, \sigma \rangle$ and $\langle \phi_1', \phi_2', \sigma' \rangle$ be two configurations. $\langle \phi_1, \phi_2, \sigma \rangle$ is K-eqivalent to $\langle \phi_1', \phi_2', \sigma' \rangle$ (written $\langle \phi_1, \phi_2, \sigma \rangle \approx_K \langle \phi_1', \phi_2', \sigma' \rangle$) iff there are bijections $f_1 : K \to K$ and $f_2 : K \to K$ such that*

(a) $\hat{f}_1(\phi_1) = \phi_1'$
(b) $\hat{f}_2(\phi_2) = \phi_2'$
(c) $\sigma' = \{\langle k_1', k_2' \rangle \mid$ *there is a* $\langle k_1, k_2 \rangle \in \sigma$ *with* $k_1' = f_1(k_1)$ *and* $k_2' = f_2(k_2)\}$

For the definition of $\hat{f}$ for a mapping $f : K \to K$, see also Def. 3.12.

Definition 4.20 (Derivation in synchronous local TDGs) *Let G_s be a pair of synchronous local TDGs, $G_s = \langle G_1, G_2, \Sigma, \langle \phi_S^1, \phi_S^2 \rangle \rangle$, and let $\langle \phi_1, \phi_2, \sigma \rangle$ and $\langle \phi_1', \phi_2', \sigma' \rangle$ be two configurations such that $\phi_S^1 \stackrel{n}{\Rightarrow} \phi_1$ with respect to G_1 and $\phi_S^2 \stackrel{n}{\Rightarrow} \phi_2$ with respect to G_2 for an $n \in \mathbb{N}$.*
$\langle \phi_1, \phi_2, \sigma \rangle \Rightarrow \langle \phi_1', \phi_2', \sigma' \rangle$ wrt G_s iff there is a configuration $\langle \psi_1, \psi_2, \sigma_\psi \rangle$ and a $\langle \psi_1', \psi_2', \sigma_\psi' \rangle \in \Sigma$ with $\langle \psi_1, \psi_2, \sigma_\psi \rangle \approx_K \langle \psi_1', \psi_2', \sigma_\psi' \rangle$, such that:

1. $\phi_1 \stackrel{\psi_1}{\Rightarrow} \phi_1'$
2. $\phi_2 \stackrel{\psi_2}{\Rightarrow} \phi_2'$
3. *If ψ_d^1 and ψ_d^2 are the derivation descriptions of these steps, then either $\psi_d^1 = \phi_S^1$ and $\psi_d^2 = \phi_S^2$ or there is an i, $1 \leq i \leq n$ such that ψ_d^1 and ψ_d^2 were both added in the i-th derivation step when deriving ϕ_1 and ϕ_2 respectively.*
4. *For all names k_1, k_2, k_3, k_4 with $\langle k_1, k_3 \rangle \in \sigma$ and $\langle k_2, k_4 \rangle \in \sigma_\psi$:*
   (a) *If $\phi_1' \vdash k_1 \approx k_2$, then there must be k, k' with $\langle k_1, k \rangle \in \sigma$ and $\langle k_2, k' \rangle \in \sigma_\psi$ and $\phi_2' \vdash k \approx k'$.*
   (b) *If $\phi_2' \vdash k_3 \approx k_4$, then there must be k, k' with $\langle k, k_3 \rangle \in \sigma$ and $\langle k', k_4 \rangle \in \sigma_\psi$ and $\phi_1' \vdash k \approx k'$.*
5. $\sigma' = \sigma \cup \sigma_\psi$.

Note that there may be new node name equivalences in one of the derivation steps ($\phi_1 \stackrel{\psi_1}{\Rightarrow} \phi_1'$ or $\phi_2 \stackrel{\psi_2}{\Rightarrow} \phi_2'$) with names that do not occur in the synchronization relation. Only if both names of a new equivalence occur in σ or σ_ψ, then there must be an equivalence with related names in the other grammar.

With synchronous local TDGs sets of pairs of descriptions are derived as description language. The tree language consists of pairs of trees and the string language of pairs of strings:

Definition 4.21 (Language of synchronous local TDGs) *Let G_s be a pair of synchronous local TDGs, $G_s = \langle G_1, G_2, \Sigma, \kappa_s \rangle$.*

1. *The description language (set of description pairs) of G_s is*
   $L_D(G_s) := \{\langle \phi_1, \phi_2 \rangle \mid$ *there is a* $\sigma \in K \times K$ *such that* $\kappa_s \stackrel{*}{\Rightarrow} \langle \phi_1, \phi_2, \sigma \rangle\}$
2. *The tree language of G_s is*
   $L_T(G_s) := \{\langle B_1, B_2 \rangle \mid$ *there is a* $\langle \phi_1, \phi_2 \rangle \in L_D(G_s)$, *such that B_1 is minimal tree of ϕ_1 and B_2 is minimal tree of ϕ_2*$\}$

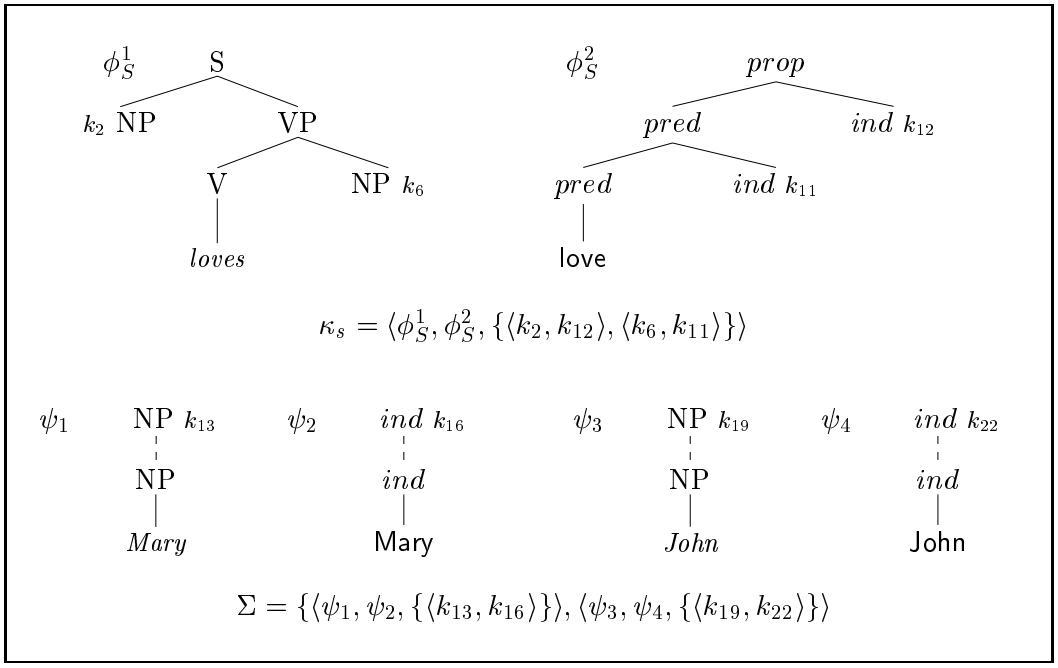


FIGURE 4.17: A SAMPLE PAIR OF SYNCHRONOUS LOCAL TDGS

3. The string language of G_s is

$$L_S(G_s) := \{\langle w_1, w_2\rangle \mid \textit{there is a } \langle B_1, B_2\rangle \in L_T(G_s), \textit{ such that } w_1 = yield(B_1) \textit{ and } w_2 = yield(B_2)\}$$

4. The left projection string language of G_s is

$$L_{left}(G_s) := \{w \mid \textit{there is a } w' \textit{ such that } \langle w, w'\rangle \in L_S(G_s)\}$$

The right projection string language of G_s is

$$L_{right}(G_s) := \{w \mid \textit{there is a } w' \textit{ such that } \langle w', w\rangle \in L_S(G_s)\}$$

With this definition of a tree language of synchronous local TDGs, the synchronization relation does not influence the choice of minimal trees for a given pair of descriptions in the description language.

A simple example of a pair of synchronous local TDGs is shown in Fig. 4.17. In this case the synchronization relations between the sets of node names are used to express relations between arguments: with $\langle k_6, k_{11}\rangle$ in κ_s, the predicate *love'* (in ϕ_S^2) is first applied to the denotation of the object of *loves* (in ϕ_S^1), and (with $\langle k_2, k_{12}\rangle$) the resulting predicate is then applied to the denotation of the subject.

4.4.2 Expressivity of synchronous local TDGs

Concerning TAGs there arise some problems with synchronous grammars, or at least some unintended and unexpected phenomena. As shown in Shieber 1994, the class of left projection string languages of synchronous TAGs is a proper superset of the class of Tree Adjoining Languages. Shieber gives the synchronous TAG shown in Fig. 2.14 with the language

$\{a^n b^n \dots h^n \mid n \geq 0\}$ as left projection. As mentioned before, this language is not a TAL. This is something that obviously was not intended because synchronous TAGs have been developed to establish a relation between two Tree Adjoining Languages. Whether this is a problem in the case of TAGs is difficult to say. We have seen that TAGs are not powerful enough to deal with all natural language phenomena, and therefore it might even be interesting to increase the generative capacity of TAGs by controlling the derivation by a second TAG.

In the case of synchronous local TDGs I will prove in this section that each left or right projection string language of a pair of synchronous local TDGs is a local TDL and vice versa. This is a nice result because local TDGs are already more powerful than TAGs and probably powerful enough to handle natural languages. Therefore it would not be desirable to increase the set of string languages of local TDGs by the synchronization mechanism.

It is obvious that each local TDL can be defined as a right or left projection string language of a pair of synchronous TDGs. But the reverse is not easy to see.

Proposition 4.11 (Expressivity of synchronous local TDGs) *For each language L that is either the right or the left projection string language of a pair G_s of synchronous local TDGs, there is a local TDG G such that $L = L_S(G)$.*

In order to facilitate the proof of this proposition, I will first show the following lemma. This lemma says that for each pair of synchronous local TDGs, there is an equivalent pair of synchronous local TDGs where minimal or marked names occuring in the synchronization relation cannot be used to introduce new node name equivalences in further derivation steps.

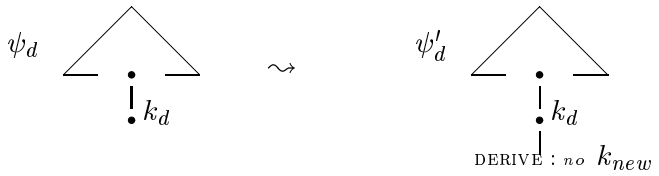
Lemma 4.2 *For each synchronous pair G_s of local TDGs there is a pair $G'_s = \langle G'_1, G'_2, \Sigma', \kappa'_s \rangle$ of local TDGs such that*

(i) $L_S(G_s) = L_S(G'_s)$, *and*

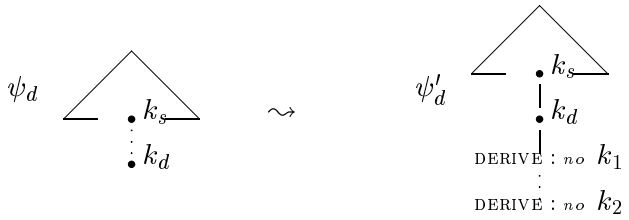
(ii) for all $\langle \psi'_1, \psi'_2, \sigma' \rangle \in \Sigma'$ and all $\langle \psi_1, \psi_2, \sigma \rangle$ K-equivalent to $\langle \psi'_1, \psi'_2, \sigma' \rangle$:
for all $i \in \{1,2\}$ and all $k \in M_{\psi_i}$ such that there is a k' with $\langle k, k' \rangle \in \sigma$ or $\langle k', k \rangle \in \sigma$:
if there is a derivation $\kappa_s \stackrel{*}{\Rightarrow} \kappa_1 \stackrel{\langle \psi_1, \psi_2, \sigma \rangle}{\Rightarrow} \langle \phi_1, \phi_2, \sigma_\phi \rangle$,
then there is no derivation $\langle \phi_1, \phi_2, \sigma_\phi \rangle \stackrel{*}{\Rightarrow} \langle \phi'_1, \phi'_2, \sigma'_\phi \rangle$ such that there is a $k_\approx$ with $\phi_i \not\vdash k \approx k_\approx$ and $\phi'_i \vdash k \approx k_\approx$.

The proof idea for this lemma is to construct the equivalent local TDGs G'_s as follows: a new attribute *derive* is introduced with possible values *yes* or *no* which signifies either that a node name can be used to introduce new equivalences in further derivation steps or that this is not allowed. In G_s, a marked or minimal name k in some elementary description ψ occuring in the synchronization relation is no leaf name and does not dominate any other marked names (see Def. 4.18). When ψ is added in a derivation step in G_s, k must be identified with a leaf name k_d in the derivation description ψ_d. k_d either 1) is unmarked and has a parent or 2) is unmarked and is strongly dominated or 3) is marked and not bottom-underspecified. In G_s, k_d may be used later for a new equivalence. This should not be possible in G'_s. Therefore ψ_d is replaced by an elementary descriptions ψ'_d that can be obtained from ψ_d by adding another name k_{new}. This name can be used later instead of k_d.

In the case 1) (k_d has a parent in ψ_d), k_{new} is a daughter of k_d and has the same labels. Additionally, $derive(\delta(k_{new})) \approx no$ is added and k_d is replaced by k_{new} in the synchronization relation (see (a)(ii) in the proof).



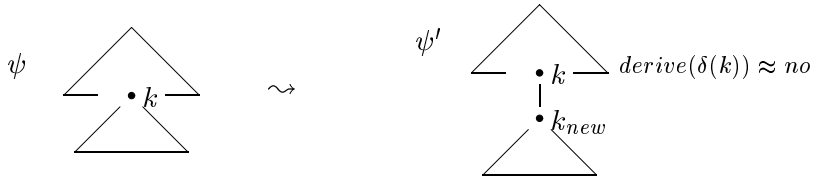
In the case 2), i.e. k_d is strongly dominated by some k_s, even two new names k_1 and k_2 must be introduced. k_1 has the same attribute value pairs as k_s and k_2 has the same as k_d. $derive(\delta(k_s)) \approx yes \wedge derive(\delta(k_d)) \approx yes \wedge derive(\delta(k_1)) \approx no \wedge derive(\delta(k_2)) \approx no$ is added (see (a)(iii) in the proof).



In the case 3) (k_d is marked and not bottom-underspecified) k_d might be identified with a leaf when adding ψ_d. For this case, a new daughter k_{new} of k_d is added with the same labels and $derive(\delta(k_{new})) \approx no$. k_d is replaced by k_{new} in the synchronization relation (see (a)(iv)).

Furthermore, in all three cases, there must also be a second description ψ''_d that can be obtained from ψ_d by adding $derive(\delta(k_d)) \approx yes$. This description is used for equivalences with leaf names (see (a)(i)).

In G_s, the name k from ψ is not a leaf name and may be later used for a new equivalence with an a-marked name. Therefore, also in ψ a new name must be added (either as daughter or as mother of k) that can be used instead of k. For k, the conjunct $derive(\delta(k)) \approx no$ is added to ψ. ψ is replaced by one new elementary description that can be obtained by adding a new daughter k_{new} to k with the same attribute value pairs as k (see (b)(i)):



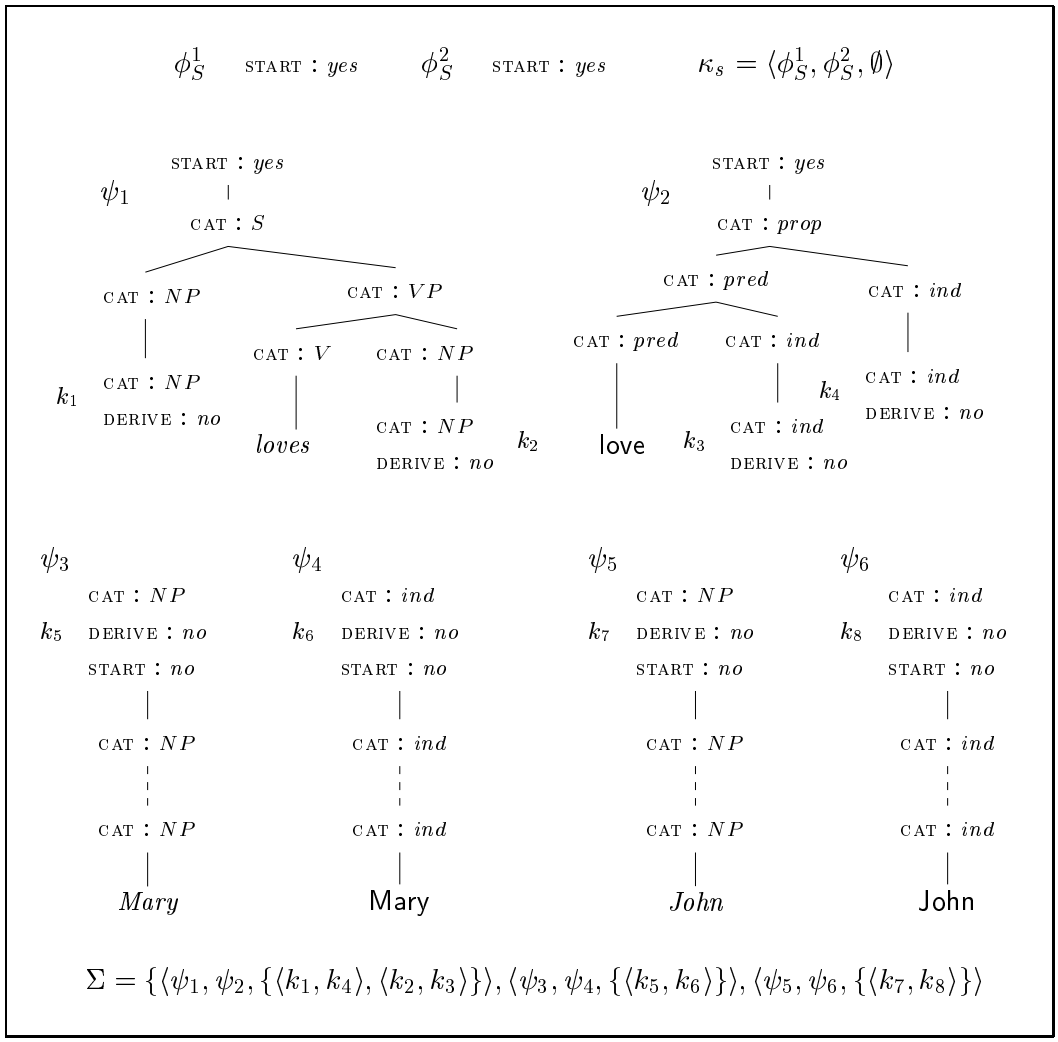
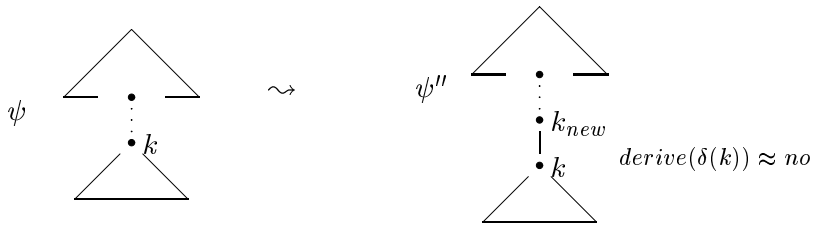


FIGURE 4.18: SYNCHRONOUS LOCAL TDGS SATISFYING LEMMA 4.2 EQUIVALENT TO THE LOCAL TDGS IN FIG. 4.17

Furthermore, if there is a k_d strongly dominating k in ψ, then a second elementary description ψ'' is added where the new name k_{new} is a parent of k and not a daughter (see (b)(ii)).



For all names k' that are either unmarked or that cannot occur in the synchronization relation, $derive(\delta(k')) \approx yes$ is added (see (c)). Therefore, when k and k_d are no longer leaf names, they cannot be used for further node name equivalences.

Fig. 4.18 shows the synchronous local TDGs constructed according to the proof of Lemma 4.2 for the local TDGs in Fig. 4.17.

Proof Let $G_s = \langle G_1, G_2, \Sigma, \kappa_s \rangle$ be a pair of synchronous local TDGs.
Without loss of generality let G_1 and G_2 be label complete.
Construction of $G'_s = \langle G'_1, G'_2, \Sigma', \kappa'_s \rangle$ with $G'_1 = (A'_1, V'_1, T_1, D'_1, \phi'_{S1})$ and $G'_2 = (A'_2, V'_2, T_2, D'_2, \phi'_{S2})$:

1. $A'_1 := A_1 \cup \{derive, start\}$ and $A'_2 := A_2 \cup \{derive, start\}$ where $derive, start \notin A_1 \cup A_2$.
2. $V'_1 := V_1 \cup \{no, yes\}$, $V'_2 := V_2 \cup \{no, yes\}$.
3. Synchronization Σ' and start configuration κ'_s: $\Sigma_s := \Sigma \cup \{\kappa_s\}$.

(*) If there is a $\kappa = \langle \psi_1, \psi_2, \sigma \rangle \in \Sigma_s$, a $i \in \{1, 2\}$, and a $k \in node(\psi_i)$ such that
- $\psi \nvdash derive(\delta(k)) \approx yes$,
- $\psi \nvdash derive(\delta(k)) \approx no$,
- and there is no $t \in T \cup \{\epsilon\}$ with $\psi_i \vdash \delta(k_l) \approx t$

then:

(a) if k is leaf name, then:

(i) a configuration κ_1 is added to Σ_s that can be obtained from κ by replacing ψ_i by $\psi_i \wedge derive(\delta(k)) \approx yes$,

(ii) if $k \notin M_{\psi_i}$, and there is a k_p with $\psi_i \vdash k_p \lhd k$, then a configuration κ_2 is added to Σ_s that can be obtained from κ as follows:
take a new $k_l \notin node(\psi_i)$,
replace ψ_i by $\psi_i \wedge k \lhd k_l \wedge derive(\delta(k_l)) \approx no$
$\bigwedge\{a(\delta(k_l)) \approx v \mid \psi_i \vdash a(\delta(k)) \approx v, a \in A_i, v \in V_i \cup \{\bot\}\}$,
and replace k by k_l in σ.

(iii) if $k \notin M_{\psi_i}$, and there is a k_d with $\psi_i \vdash_s k_d \lhd^* k$,
then a configuration κ_3 is added to Σ_s that can be obtained from κ as follows:
take two new names $k_1, k_2 \notin node(\psi_i)$,

replace ψ_i by
$\psi_i \wedge k_d \lhd k \wedge k \lhd k_1 \wedge k_1 \lhd^* k_2$
$\wedge derive(\delta(k)) \approx yes \wedge derive(\delta(k_1)) \approx no \wedge derive(\delta(k_2)) \approx no$
$\bigwedge\{a(\delta(k_1)) \approx v \mid \psi_i \vdash a(\delta(k_d)) \approx v, a \in A_i, v \in V_i \cup \{\bot\}\}$
$\bigwedge\{a(\delta(k_2)) \approx v \mid \psi_i \vdash a(\delta(k)) \approx v, a \in A_i, v \in V_i \cup \{\bot\}\}$

(iv) if $k \in M_{\psi_i}$, and k is not bottom underspecified,
then a configuration κ_4 is added to Σ_s that can be obtained from κ as follows:
take a new $k_l \notin node(\psi_i)$,
replace ψ_i by $\psi_i \wedge k \lhd k_l \wedge derive(\delta(k)) \approx yes \wedge derive(\delta(k_l)) \approx no$
$\bigwedge\{a(\delta(k_l)) \approx v \mid \psi_i \vdash a(\delta(k)) \approx v, a \in A_i, v \in V_i \cup \{\bot\}\}$,

(v) then κ is removed from Σ_s.

(b) if $k \in M_\psi$ and k occurs in σ,
then

(i) a configuration κ_1 is added that can be obtained from κ as follows:
take a new $k_{new} \notin node(\psi_i)$,
replace k by k_{new} in all conjuncts of the form $k \lhd^* k'$ or $k \lhd k'$ for some $k' \in node(\psi_i)$ in ψ_i,
and replace ψ_i by
$\psi_i \wedge k \lhd k_{new} \wedge derive(\delta(k)) \approx no \wedge derive(\delta(k_{new})) \approx yes$
$\bigwedge\{a(\delta(k_{new})) \approx v \mid \psi \vdash a(\delta(k)) \approx v, a \in A_i, v \in V_i \cup \{\bot\}\}$

(ii) if there is a k_d with $\psi_i \vdash_s k_d \lhd^* k$, then a second configuration κ_2 is added that can be obtained from κ by
taking a new $k_{new} \notin node(\psi_i)$ and
replacing ψ_i by
$\psi_i \wedge k_d \lhd^* k_{new} \wedge k_{new} \lhd k$
$\wedge\, derive(\delta(k)) \approx no \wedge derive(\delta(k_{new})) \approx yes$
$\bigwedge\{a(\delta(k_{new})) \approx v \mid \psi \vdash a(\delta(k)) \approx v, a \in A_i, v \in V_i \cup \{\bot\}\}$.

(iii) then κ is removed from Σ_s.

(c) if neither condition (a) nor (b) holds for k, then replace ψ_i in κ by $\psi_i \wedge derive(\delta(k)) \approx yes$

Repeat **(*)** until no such k is left.

4. $\kappa'_s := \langle start(\delta(k_1)) \approx yes, start(\delta(k_2)) \approx yes, \emptyset \rangle$
For each configuration $\kappa = \langle \phi_1, \phi_2, \sigma \rangle$ constructed from κ_s, for each $i \in \{1,2\}$: if k_m is the minimal name in ϕ_i, then take a new name $k \notin \phi_i$ and replace ϕ_i by $\phi_i \wedge k \lhd k_m \wedge start(\delta(k)) \approx yes$.
For each configuration $\kappa = \langle \phi_1, \phi_2, \sigma \rangle$ that was not constructed from κ_s, for each $i \in \{1,2\}$: if k_m is the minimal name in ϕ_i, then replace ϕ_i by $\phi_i \wedge start(\delta(k_m)) \approx no$.
$\Sigma' := \Sigma_s$.

The result is a synchronous pair G'_s of local TDGs satisfying condition *(ii)* in the lemma.

To show: $L_S(G_s) = L_S(G'_s)$.

First, it can be shown that, roughly said, in a configuration κ derived in G'_s, names with *no* as value of *derive* that are no leaf names can be removed without modifying the set of derivable pairs of strings:
let $\kappa = \langle \phi_1, \phi_2, \sigma \rangle$ be a configuration with $\kappa'_s \stackrel{*}{\Rightarrow} \kappa$ in G'_s. Then the configuration $mod(\kappa)$ can be obtained from κ as follows:

(M) if there is an $i \in \{1,2\}$ and a $k \in node(\phi_i)$ coming from the elementary ψ such that $\psi_i \vdash derive(\delta(k)) \approx no$ and k is no leaf name,
then

1. if, when constructing ψ, k was added as k_l in (a)(ii) or (a)(iv) or as k_{new} in (b)(i), and if k' is the name with $\psi \vdash k' \lhd k$,
then remove the conjuncts $k' \lhd k$ and $derive(\delta(k)) \approx no$ from ϕ_i and replace in ϕ_i all occurrences of k by k'.
2. if, when constructing ψ, k was added as k_1 in (a)(iii), and if k' was added as k_2, then: if k'_1, k'_2 are the old names in (a)(iii) with $\psi \vdash k'_1 \lhd k'_2$, then remove the conjuncts $k'_1 \lhd k'_2$, $k'_2 \lhd k$, $derive(\delta(k)) \approx no$ and $derive(\delta(k')) \approx no$, and replace all occurrences of k by k'_1 and all occurrences of k' by k'_2.
3. if, when constructing ψ, $derive(\delta(k)) \approx no$ was added as in (b)(ii), and if k_{new} was the new name with $\psi \vdash k_{new} \lhd k$, then
remove the conjuncts $k_{new} \lhd k$ and $derive(\delta(k)) \approx no$ from ϕ_i and replace in ϕ_i all occurrences of k by k_{new}.

repeat (M) until no such k is left.

The following can be easily shown: for each κ with $\kappa'_s \stackrel{*}{\Rightarrow} \kappa$ in G'_s and each elementary κ_e in G'_s:
$\kappa \stackrel{\kappa_e}{\Rightarrow} \kappa'$ with new equivalences $k_1 \approx k'_1, \ldots, k_n \approx k'_n$ iff $mod(\kappa) \stackrel{\kappa_e}{\Rightarrow} \kappa''$ with new equivalences $k_1 \approx k'_1, \ldots, k_n \approx k'_n$ and $mod(\kappa') = mod(\kappa'')$.
Since the modification does not influence the strings yielded by the two descriptions, this means that, instead of one derivation step $\kappa_1 \Rightarrow \kappa_2$ in G'_s, a derivation mode $\Rightarrow_{mod}$ can be used with
$\kappa_1 \Rightarrow_{mod} \kappa_2$ iff there is a κ with $\kappa_1 \Rightarrow \kappa$ (in G'_s) and $\kappa_2 = mod(\kappa)$.
Then, with the construction of G'_s, for all configurations κ: $\kappa_s \stackrel{*}{\Rightarrow} \kappa$ in G_s iff there is a κ' that can be constructed for κ as in (a) to (c) with $\kappa'_s \stackrel{*}{\Rightarrow}_{mod} \kappa'$.

□

With this lemma, the proof of Prop. 4.11 is easy to do. For a pair of synchronous local TDGs G_s, a local TDG G generating the left (or right) projection string language of G_s can be constructed in the following way (in general, G is not one of the TDGs in the pair G_s):

In one elementary description of G, the two elementary descriptions of one pair in G_s occur as two subdescriptions. In the case of the left projection language, all terminals in the second TDG are replaced by ϵ. For the right projection, of course all terminals in the first TDG are replaced by ϵ.

The start description consists of the two descriptions in κ_s together with a new name that is parent of the minimal names of these two descriptions. New attributes $\overline{k}$ are added for all leaf names k with nonterminal labels. Instead of the synchronization relation, these

attributes control the derivation process. Their values are 0 or 1. For each of these leaf names k, $\overline{k}(\delta(k)) \approx 1$ holds.

An elementary description in the new TDG is obtained from two descriptions ψ_1, ψ_2 related by Σ and, furthermore, it depends on the pairs of derivation descriptions ψ_d^1, ψ_d^2 that can be used in a dervation step where ψ_1 and ψ_2 are added. The idea is that for a marked name k_m in a description ψ that occurs in the synchronization relation, the attributes $\overline{k}$ specify the node name that must become equivalent to k_m in a derivation step where ψ is added. If k_d is this name, then $\overline{k_d}(\delta(k_m)) \approx 1$ holds and for all $\overline{k}$ with $k \neq k_d$, $\overline{k}(\delta(k_m)) \approx 0$. The elementary description consists of ψ_1 and ψ_2 with the additional attributes $\overline{k}$ and a new name dominating the minimal names of ψ_1 and ψ_2.

Fig. 4.19 shows the local TDG constructed in such a way for the left projection language of the synchronous TDGs in Fig. 4.18 (some of the new attributes $\overline{k}$ and the attributes *derive* are omitted in this figure).

Without loss of generality, one can suppose that for none of the names k with $\overline{k'}(\delta(k)) \approx 0$ for some k', any further equivalences are possible (see Lemma 4.2). Therefore these attributes are used to control only one derivation step and there arise no problems with further new equivalences.

Proof of Prop. 4.11 I will consider only the case of the left projection string language, the other case is analogous.
Let $G_s = \langle G_1, G_2, \Sigma, \kappa_s \rangle$ be a synchronous local TDG with $G_1 = (A_1, V_1, T_1, D_1, \phi_S^1)$ and $G_2 = (A_2, V_2, T_2, D_2, \phi_S^2)$.
Without loss of generality, I assume that G_s satisfies the conditions of the G_s' in Lemma 4.2 and that the sets of node names of all elementary descriptions occuring in Σ or κ_s are pairwise disjoint.
To show: $L := L_{left}(G_s)$ is a local TDL.
Construction of a local TDG $G = (A, V, T, D, \phi_S)$ with $L = L_S^l(G)$:

$\Sigma_s := \Sigma \cup \{\kappa_s\}$. Without loss of generality suppose that for all $\langle \psi_1, \psi_2, \sigma_1 \rangle, \langle \psi_3, \psi_4, \sigma_2 \rangle \in \Sigma_s$: $node(\psi_i) \cap node(\psi_j) = \emptyset$ for $i \neq j$, $i, j \in \{1, 2, 3, 4\}$.
First, a set Σ_G will be constructed, and then from each element in Σ_G, one elementary description in G can be obtained.
$\Sigma_G := \emptyset$.
For each $\langle \psi_d^1, \psi_d^2, \sigma_d \rangle \in \Sigma_s$ and for all $\kappa \in \Sigma_s \setminus \{\kappa_s\}$, $\kappa = \langle \psi_1, \psi_2, \sigma \rangle$:
For all $k_1, \ldots, k_n \in node(\psi_1) \cup node(\psi_2)$ and $k_1', \ldots, k_n' \in node(\psi_d^1) \cup node(\psi_d^2)$ such that:

(a) $k \in \{k_1, \ldots, k_n\}$ iff $k \in M_{\psi_1} \cup M_{\psi_2}$, and there is a k' such that $\langle k, k' \rangle \in \sigma$ or $\langle k', k \rangle \in \sigma$,
(b) $k_1', \ldots, k_n'$ are leaf names in ψ_d^1 or ψ_d^2,
(c) for all $i, 1 \leq i \leq n$: $k_i \in node(\psi_1)$ iff $k_i' \in node(\psi_d^1)$.
(d) for all $i, j, 1 \leq i, j \leq n$: if $\langle k_i, k_j \rangle \in \sigma$, then either $\langle k_i', k_j' \rangle \in \sigma_d$, or there is no k with $\langle k_i', k \rangle \in \sigma_d$ or $\langle k, k_j' \rangle \in \sigma'$.
(e) for all $i, j, 1 \leq i, j \leq n$: if $\langle k_i', k_j' \rangle \in \sigma_d$, then $\langle k_i, k_j \rangle \in \sigma$.

$\langle \langle \psi_1, \psi_2, \sigma \rangle, \{\langle k_1, k_1' \rangle, \ldots, \langle k_n, k_n' \rangle\} \rangle \in \Sigma_G$.
These are all elements in Σ_G.
For all $\langle \langle \psi_1, \psi_2, \sigma \rangle, X \rangle \in \Sigma_G$, replace all terminals occuring in ψ_2 by ϵ.

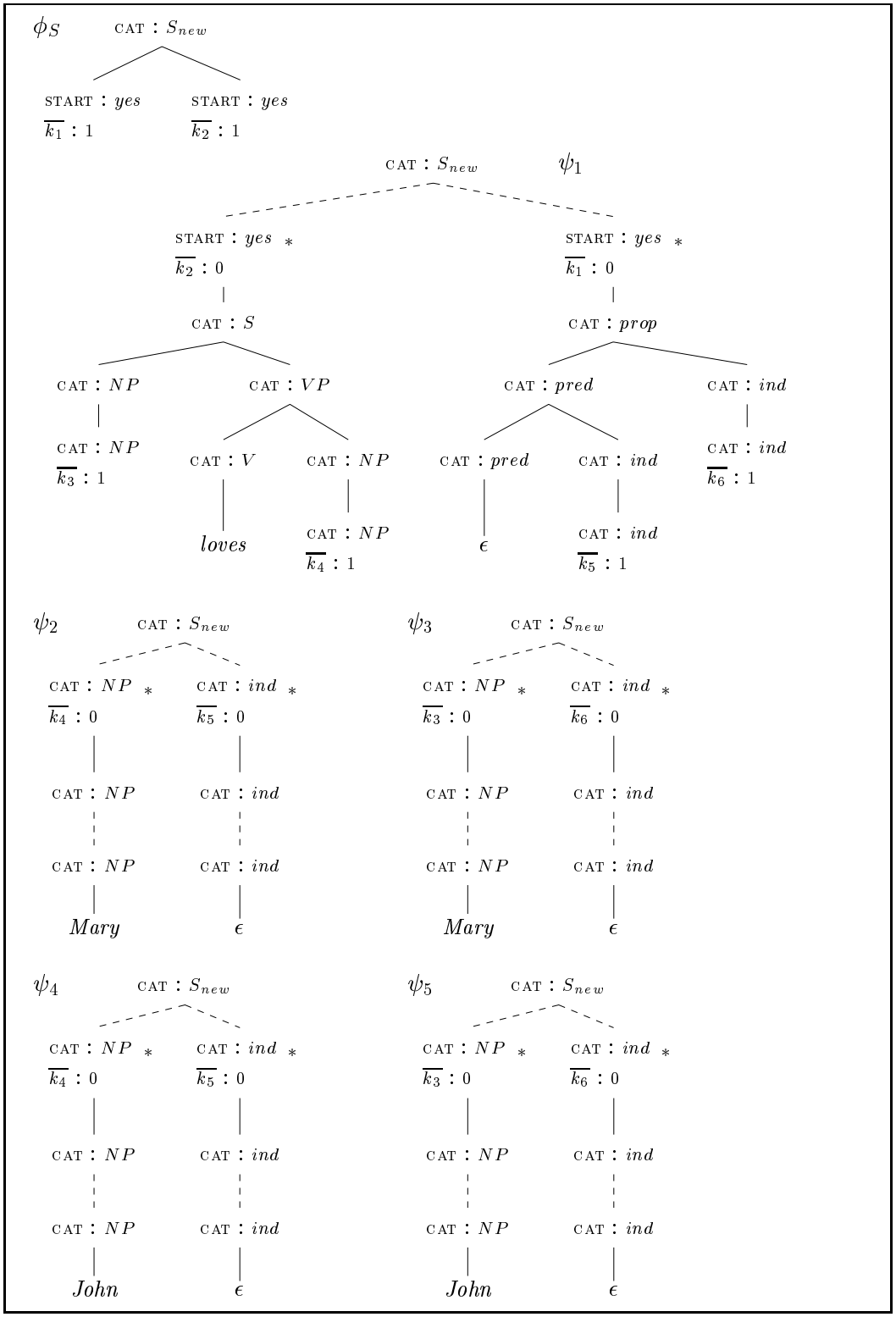


FIGURE 4.19: LOCAL TDG FOR THE LEFT PROJECTION LANGUAGE OF THE SYNCHRONOUS TDGS IN FIG. 4.18

$$A_K := \{\overline{k} \mid k \text{ is a leaf name in some } \psi \text{ in } \Sigma_G \text{ and there is no } t \in T \cup \{\epsilon\} \text{ with } \psi \vdash \delta(k) \approx t\}$$

Now the local TDG G will be constructed:

1. $T := T_1$.
2. $A := A_1 \cup A_2 \cup A_K \cup \{cat\}$ (without loss of generality I suppose that $A_K, A_1 \cup A_2$ are pairwise disjoint).
3. $V := V_1 \cup V_2 \cup \{1, 0, S_{new}\}$ (without loss of generality I suppose that $1, 0, S_{new} \notin V_1 \cup V_2$).
4. Elementary descriptions D:
   For all $\langle\langle\psi_1, \psi_2, \sigma\rangle, X\rangle \in \Sigma_G$ with $X \neq \emptyset$ and with k_1^m and k_2^m minimal in ψ_1 and ψ_2 respectively, there is a $\langle\psi, K_\psi\rangle \in D$ of the following form:
   There is a new $k \notin node(\psi_1) \cup node(\psi_2)$ such that $K_\psi = K_{\psi_1} \cup K_{\psi_2} \cup \{k_1^m, k_2^m\}$ and

$$\begin{aligned}\psi \;=\; & \psi_1 \wedge \psi_2 \wedge k \lhd^* k_1^m \wedge k \lhd^* k_2^m \wedge cat(\delta(k)) \approx S_{new} \\ & \bigwedge\{\overline{k}(\delta(k)) \approx 1 \mid \overline{k} \in A_K \text{ and } k \in node(\psi_1) \cup node(\psi_2)\} \\ & \bigwedge\{\overline{k'}(\delta(k)) \approx 0 \mid \overline{k'} \in A_K, \langle k, k'\rangle \notin X, \text{ and there is a } k'' \text{ with } \langle k, k''\rangle \in X\}\end{aligned}$$

   These are all elementary descriptions in D.
5. Start description: let k_1^m, k_2^m be the two minimal names of ϕ_S^1 and ϕ_S^2 respectively. Take a new $k \notin node(\phi_S^1) \cup node(\phi_S^2)$. Then the start description is

$$\phi_S := \phi_S^1 \wedge \phi_S^2 \wedge k \lhd k_1^m \wedge k \lhd k_2^m \wedge k_1^m \prec k_2^m \wedge cat(\delta(k)) \approx S_{new}$$

The result is a local TDG.

$L_S^l(G) = L_{left}(G_s)$ is easy to see.

□

With respect to the rank of the local TDG constructed in the preceding proof, the following holds: if $G_s = \langle G_1, G_2, \Sigma, \kappa_s\rangle$ is the original pair of synchronous local TDGs and if G_1 is of rank n_1 and G_2 of rank n_2, then the rank of the resulting local TDG generating the left (or right) projection string language of G_s is $2 + n_1 + n_2$. In other words, the proof of Prop. 4.11 shows not only that the left and right projection string languages of synchronous local TDGs are local TDLs but also that the rank of these languages is limited by the ranks of the two synchronous local TDGs.

4.5 Conclusion: properties of local TDGs

In this chapter, a local variant of the TDGs presented in Chapter 3 has been proposed. I will now summarize some of the results shown for local TDGs, and I will raise some interesting issues for further work.

Local TDGs preserve the attractive properties of TDGs with respect to natural languages, namely the extended domain of locality, the possibility of multicomponent derivation, a uniform distinction between complementation and modification, and the possibility to generate underspecified representations. As an example for natural language applications, I will

present a syntax-semantics interface in the following chapter. We will see then, that local TDGs enable us to derive underspecified representations for quantifier scope ambiguities. For these phenomena local TDGs are even more appropriate than general TDGs since island constraints hold simply as a consequence of the locality restriction in the grammar formalism.

With respect to the formal properties of local TDGs, some of the results shown for general TDGs still hold for local TDGs, in particular the decidability of the word problem for lexicalized local TDGs.

Concerning the generative capacity of local TDGs, I have shown that local TDGs generate only semilinear languages. This indicates that local TDGs are suitable to describe natural languages, since the languages generated by local TDGs satisfy the Constant Growth Property. Furthermore, I have shown that local TDGs are at least as powerful as set-local MC-TAGs. Their derivational generative capacity is even greater than the one of set-local MC-TAGs.

It is interesting that the local TDG for an indexed string language that cannot be generated by a set-local MC-TAG contains elementary descriptions with underspecification pairs. The fact that the locality restriction concerns only the derivation of descriptions and not the construction of corresponding minimal trees, seems to be crucial for the generative power of local TDGs. Local TDGs without underspecification pairs are probably less powerful than local TDGs with arbitrary local descriptions. It might be no problem to show that local TDGs without underspecification are equivalent to linear context-free rewriting systems as presented by Weir (1988). Then it would be desirable to find a characterization of the extension of the generative capacity caused by the possibility of underspecified dominance relations. However, this issue is not in the scope of this dissertation and I leave it for further research.

Another question that suggests itself concerns a comparison between (general) TDGs and local TDGs. One might suppose that there are at least languages generated by general TDGs that are not local TDLs. But it seems hard to compare the two formalisms in a general way. There is perhaps even no subset relation between the two classes of languages, i.e. it might be the case that there are also local TDLs that are not generated by TDGs.

A subject that has been left aside in this thesis is the problem of parsing and in particular parsing complexity. I hope to deal with this question in the future. Perhaps it is possible to show that for a specific $n \in \mathbb{N}$, local TDGs of rank n are mildly context-sensitive, i.e. that besides their restriction to semilinear languages, they are polynomially parsable and they allow only limited cross-serial dependencies.

A further problem that I would like to consider in future research is the question whether there is a description-based grammar defining (local) TDLs that is not derivation-based, and how such a grammar looks like. For monadic second order logic for example, Rogers (1994, 1996) has shown that only context-free languages or more precisely only local tree sets can be characterized with this logic, i.e. it is not possible to define local TDLs using a monadic second order logic. The problem of defining local TDLs without derivations is probably a complex issue and shall be left for further work.

Chapter 5

A syntax-semantics interface

In the preceding chapter, local TDGs, a variant of the TDGs introduced in Chapter 3, were presented. In particular, synchronous local TDGs, which can be used to describe relations between two languages, were introduced. This formalism will be used to describe the relation between syntax and semantics.

In Chapter 2, we saw the shift from trees to tree descriptions to be motivated by a number of linguistic observations: Kroch and Joshi (1987) have shown that the derivational generative capacity of TAGs is insufficient for natural languages and that a kind of multicomponent derivation is necessary. This can be captured by tree descriptions. Furthermore, Rambow (1994a) treats certain non-local dependencies such as long-distance scrambling by relaxing the parent relation, which means a replacing of trees with tree descriptions. Another advantage of tree descriptions is the possibility to have a uniform complementation operation besides adjunction-like operations. And of course, the use of tree descriptions allows us to obtain underspecified representations without adding any special mechanisms (as in Bos 1995 and Reyle 1993 for example). Tree descriptions are underspecified by nature and therefore, in a formalism based on tree descriptions, underspecified representations can be obtained in an elegant and natural way.

To incorporate semantics into the TAG formalism, Shieber and Schabes (1990) propose to use a second TAG for semantic representations besides the one for syntactic structures. These two TAGs are connected by a synchronization relation (see also 2.4.2 on page 38 in Chapter 2). Following these ideas, I will present a syntax-semantics interface for French using synchronous local TDGs. For semantic representations a TDG G_{sem} will be developed that gives a truth conditional semantics of French. This TDG depends on the syntax in some compositional way which is realized by synchronizing the syntax TDG G_{syn} (with French as its string language) with G_{sem}.

Most theories in the Montagovian tradition assume syntax and semantics to be closely related. Usually, one syntactical phrase structure rule is connected to a semantic rule. In other words, a tree of height 1 (with one mother and n daughters) is related to a rule describing how to obtain the semantic interpretation of the mother node from the interpretations of the daughters. Because of this close relation, these systems need higher order logics for truth conditions and rules like *Quantifier Raising* or *Quantifying In*. Furthermore, in some cases different types and interpretations of a lexical item (i.e. different readings) must be distinguished depending on the syntactic context in which the lexical item may occur. As

we will see, with synchronous local TDGs, the semantics also depends on the syntax in a compositional way, but the relation between syntax and semantics is less close than in Montagovian theories. In the framework used in this chapter, a syntactic tree description is related to a second tree descriptions that characterizes the contribution of the first one to the semantic interpretation. Since these tree descriptions can be larger than a tree of height 1 and since there need not be a direct correspondence between the node names in these two descriptions, syntax and semantics are less closely related to each other. As a consequence of this, no additional mechanisms are required to account for differences between (surface) word order and semantic interpretation. Furthermore, the truth-conditional logic can be relatively simple, and, except in cases of lexical ambiguity, each lexical item has a unique type and interpretation.

The work presented here is limited to certain selected phenomena, and there are many important issues in the context of underspecified semantics that are not taken into account here, e.g. the problem of reasoning with underspecified representations. They must be subject of further research. This chapter is mainly concerned with the generation of (underspecified) semantic representations.

Parts of this work are also presented in Kallmeyer 1997b, 1999a.

The syntax-semantics interface presented in the course of this chapter covers a fragment including the following data:

- Restrictive relative clauses, e.g.

  un étudiant, qui est dans la bibliothèque, lit un livre

  ('*a student who is in the library reads a book*')

- Quantifier scope ambiguities, e.g.

  un homme aime chaque femme

  ('*a man loves every woman*')

- De re - de dicto ambiguities, e.g.

  Jean croit que Marie lit un livre

  ('*Jean believes that Marie reads a book*')

- Structural ambiguities with nominal modifiers, e.g.

  ancien château royal

  ('*former royal castle / royal former castle*')

For the local TDG G_{syn} describing the syntax of French, I will adopt analyses proposed by Anne Abeillé for a TAG framework (see Abeillé 1988a,b, 1994).

This chapter is structured in the following way: firstly I will give a brief and informal introduction to synchronous local TDGs. In Section 5.2 I will present Montague's approach to quantifiers and Cooper's quantifier storage. Furthermore, I will list some constraints that hold for quantifier scope. In the following two sections (5.3 and 5.4) a fragment of a syntax-semantics interface for French will be presented: in Section 5.3 a logic for the truth conditional semantics is proposed. In the fourth section, the elementary configurations of

the synchronous local TDGs are presented. After that I will show in 5.5 that this grammar allows the derivation of underspecified representations for scope ambiguities and that these underspecified representations are such that so-called "island" constraints for quantifier scope are respected. The approach presented in this chapter will be compared to other related theories. Finally, in the conclusion, the main properties of the syntax-semantics interface presented in this chapter wil be summarized.

5.1 Synchronous local TDGs

In this section, I will briefly and informally revise the formalism of synchronous local TDGs as it is defined in Sections 4.1 and 4.4 in Chapter 4. This is mainly for those readers who want to follow this chapter without reading all the formal details in the previous chapter.

5.1.1 Local TDGs

First, I will repeat the idea of local TDGs: local TDGs consist of tree descriptions, so-called *elementary descriptions*, and a specific *start description*. These tree descriptions are formulas in a quantifier-free first order logic. They are negation- and disjunction-free. The tree logic was presented in Section 3.1 in Chapter 3. It allows the description of relations between node names k_1, k_2 such as parent relation (i.e. immediate dominance) $k_1 \lhd k_2$, dominance (reflexive transitive closure of the parent relation) $k_1 \lhd^* k_2$, linear precedence $k_1 \prec k_2$ and equality $k_1 \approx k_2$. Furthermore, nodes are supposed to be labelled by terminals or by atomic feature structures (see Def. 3.2, page 57). The labeling function is denoted by the symbol δ, and for a node name k, $\delta(k) \approx t$ signifies that k has a terminal label t, and $a(\delta(k)) \approx v$ signifies that k is labelled by a feature structure containing the attribute value pair $\langle a, v\rangle$. In an elementary description ψ, some of the node names are marked (those in the set K_ψ).

A sample local TDG is shown in Fig. 5.1. In the graphical representations, the marked names are those with an asterisk. Later I will often omitt some of the node names, sometimes even all node names, in the graphical representations. Only labels and parent and dominance relations are always depicted.

In order to avoid confusion, I want to emphasize once again that local TDGs are description-based, i.e. that the elements of a local TDG are descriptions and not structures.

Local TDGs generate tree descriptions. A derivation in a local TDG starts with the start description. In each derivation step, a derived description ϕ_1 and an elementary description ψ are used to obtain a new description ϕ_2. Roughly said, ϕ_2 can be viewed as a conjunction of ϕ_1, ψ and new equivalences between node names from ϕ_1 and from ψ, i.e. $\phi_2 = \phi_1 \wedge \psi \wedge k_1 \approx k_1' \wedge \cdots \wedge k_n \approx k_n'$ where k_i are names from ϕ_1 and k_i' are names from ψ for $1 \leq i \leq n$.[1] This derivation step must be such that

1. for a node name k_ψ in ψ, there is a new equivalence iff k_ψ is either marked (i.e. in K_ψ as k_9 in ψ_1 in Fig. 5.1) or k_ψ is minimal (dominated by no other name, e.g. k_4 in ψ_1 in Fig. 5.1),

[1] In some cases new dominance relations are also added, but I will leave this aside for this informal presentation. The derivation steps in this chapter are all such that only node name equivalences are added.

Local TDG with start description ϕ_S and elementary descriptions ψ_1, ψ_2:

$$\begin{aligned}
\phi_S &= k_1 \lhd^* k_2 \wedge k_2 \lhd k_3 \wedge cat(\delta(k_1)) \approx N \wedge cat(\delta(k_2)) \approx N \wedge \delta(k_3) \approx \textit{professor} \\
\psi_1 &= k_4 \lhd^* k_5 \wedge k_5 \lhd k_6 \wedge k_6 \lhd k_7 \wedge k_5 \lhd k_8 \wedge k_8 \lhd^* k_9 \wedge k_6 \prec k_8 \\
&\quad \wedge cat(\delta(k_4)) \approx N \wedge cat(\delta(k_5)) \approx N \wedge cat(\delta(k_6)) \approx AP \wedge \delta(k_7) \approx \textit{former} \\
&\quad \wedge cat(\delta(k_8)) \approx N \wedge cat(\delta(k_9)) \approx N \\
K_{\psi_1} &= \{k_9\} \\
\psi_2 &= k_{10} \lhd^* k_{11} \wedge k_{11} \lhd k_{12} \wedge k_{12} \lhd^* k_{13} \wedge k_{11} \lhd k_{14} \wedge k_{14} \lhd k_{15} \wedge k_{12} \prec k_{14} \\
&\quad \wedge cat(\delta(k_{10})) \approx N \wedge cat(\delta(k_{11})) \approx N \wedge cat(\delta(k_{12})) \approx N \wedge cat(\delta(k_{13})) \approx N \\
&\quad \wedge cat(\delta(k_{14})) \approx PP \wedge \delta(k_{15}) \approx \textit{in Tübingen} \\
K_{\psi_2} &= \{k_{13}\}
\end{aligned}$$

Graphical representations:

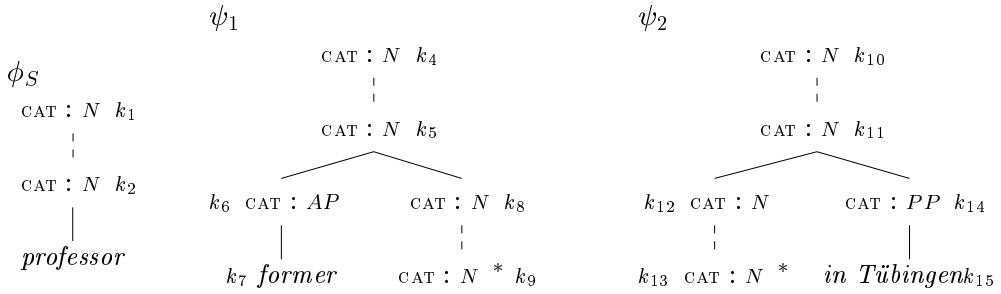


FIGURE 5.1: SAMPLE LOCAL TDG

2. the names $k_1, \ldots k_n$ from ϕ_1 that are used for the new equivalences must be part of one single elementary or start description, the so-called *derivation description* of this derivation step (locality condition),

3. and the result ϕ_2 must be as underspecified as possible, i.e. up to some renaming of node names, ϕ_2 must not entail any other possible result of the derivation step.

Of course, some details are omitted here. But this presentation of the derivation is sufficient to understand the examples treated in this chapter. For a more detailed and formal presentation see Section 4.1 in Chapter 4.

Sample derivation steps with the local TDG in Fig. 5.1 are the following: $\phi_S \overset{\psi_1}{\Rightarrow} \phi_2$ with $\phi_2 = \phi_S \wedge \psi_1 \wedge k_1 \approx k_4 \wedge k_2 \approx k_9$ and also $\phi_2 \overset{\psi_2}{\Rightarrow} \phi_3$ with $\phi_3 = \phi_2 \wedge \psi_2 \wedge k_1 \approx k_{10} \wedge k_2 \approx k_{13}$. Graphical representations of ϕ_2 and ϕ_3 are shown in Fig. 5.2. (The fact that one node name is depicted left of another node names does not necessarily mean that there is really a linear precedence relation between the two, e.g. k_8 is depicted left of k_{12} in ϕ_3 but $\phi_3 \vdash k_8 \prec k_{12}$ does not hold.) In the second derivation step, ϕ_3 is the only description that can be derived by putting ϕ_2 and ψ_2 together with ϕ_S as derivation description because the result of a derivation step must be maximally underspecified (for a formalization of this condition, see

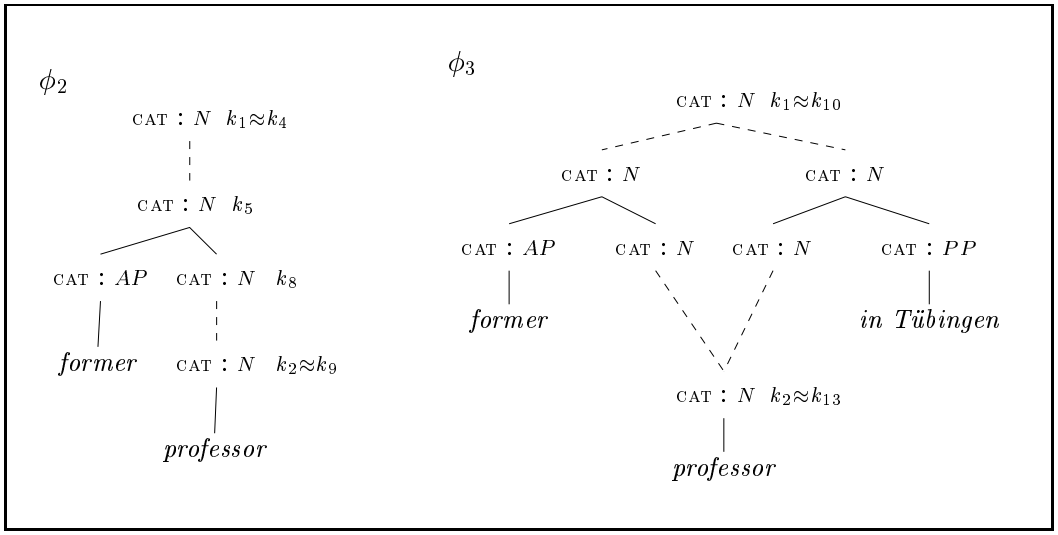


FIGURE 5.2: SAMPLE DERIVED DESCRIPTIONS

axiom *(L4)* in Def. 4.8, p. 120).

5.1.2 Synchronization

Synchronous local TDGs are presented in Section 4.4, page 154 in Chapter 4. A pair G_s of synchronous local TDGs (see Def. 4.18) consists of two local TDGs G_1, G_2 and a synchronization relation Σ between the elementary descriptions of these two TDGs. For each pair of descriptions ψ_1 and ψ_2 in Σ, there is an additional relation σ between the node names of the two descriptions, i.e. the elements of Σ are triples $\langle \psi_1, \psi_2, \sigma \rangle$, so-called *elementary configurations*. Besides Σ, there is a *start configuration* κ_s that consists of the two start descriptions of G_1 and G_2 and a relation between the two sets of node names of these descriptions. For a sample pair of synchronous local TDGs see Fig. 4.17 on page 157.

Derivation in the two TDGs is done in parallel. Starting from the start configuration, further configurations (triples of two descriptions and a relation between node names) are derived by adding an elementary configuration in each derivation step. In such a derivation step $\langle \phi_1, \phi_2, \sigma \rangle \Rightarrow \langle \phi_1', \phi_2', \sigma' \rangle$ with an elementary $\langle \psi_1, \psi_2, \sigma_\psi \rangle$, the following must hold (see Def. 4.20):

1. $\phi_1 \overset{\psi_1}{\Rightarrow} \phi_1'$ is a local derivation step in G_1, and $\phi_2 \overset{\psi_2}{\Rightarrow} \phi_2'$ is a local derivation step in G_2,
2. the derivation descriptions used in these two steps are related by Σ,
3. and if equivalences $k_1 \approx k_2$ and $k_3 \approx k_4$ are added to derive ϕ_1 and ϕ_2 respectively, such that k_1 is related to k_3 and there is a node name related to k_2 or k_4, then k_2 and k_4 must be also related to each other.
4. σ' is the union of σ and σ_ψ.

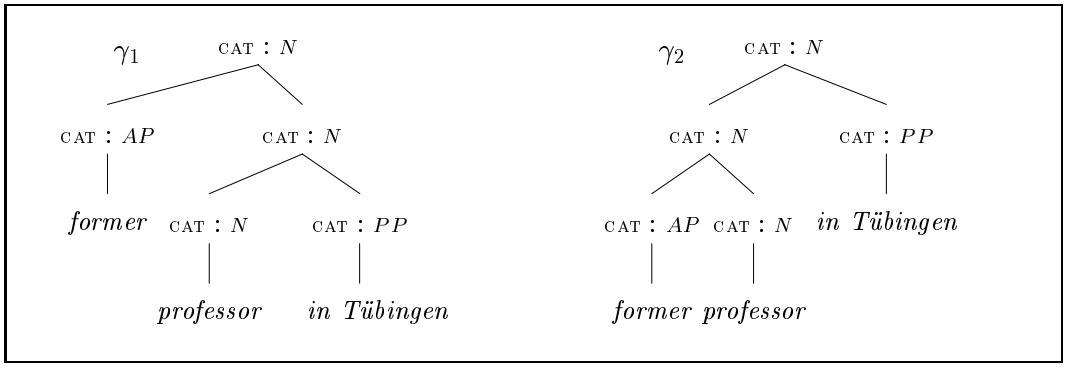


FIGURE 5.3: SYNTACTIC STRUCTURES OF *former professor in Tübingen*

5.1.3 Minimal trees

The set generated by a pair G_s of synchronous local TDGs is a set of pairs of descriptions, those pairs $\langle \phi_1, \phi_2 \rangle$ where there is a relation $\sigma \subseteq node(\phi_1) \times node(\phi_2)$ such that $\kappa_s \stackrel{*}{\Rightarrow} \langle \phi_1, \phi_2, \sigma \rangle$. This set of pairs of description is called the *descripition language* of G_s (written $L_D(G_s)$).

From these pairs of descriptions, corresponding pairs of so-called *minimal trees* (see Def. 3.15 on page 79 in Chapter 3) can be obtained. The notion of minimal trees is motivated by the following consideration: a description such as ϕ_3 in Fig. 5.2 is satisfied by an infinite number of trees. However, as "smallest" models this description is intended to desribe the two syntactic structures of *former professor in Tübingen*, i.e. the two trees in Fig. 5.3. Therefore, a tree γ is a minimal tree of a description ϕ, if γ satisfies ϕ in such a way that

1. all parent relations in γ are described in ϕ, and
2. if two different node names in ϕ denote the same node in γ, then these two names neither have both a parent in ϕ nor have both a daughter in ϕ.

The first condition makes sure that everything in γ is described in ϕ, and the second condition says that no parent relation in the tree is described more than once in ϕ.

For a given description, the corresponding minimal trees can be obtained by adding equivalences between node names and equivalences between labels. A detailed example was shown in Chapter 3, page 80 after the definition of minimal trees. In the case of ϕ_3 in Fig. 5.2, the minimal tree γ_1 can be obtained by adding the equivalences $k_1 \approx k_5$, $k_8 \approx k_{11}$ and $k_{12} \approx k_2$ whereas γ_2 can be obtained by adding $k_1 \approx k_{11}$, $k_{12} \approx k_5$ and $k_8 \approx k_2$.

The *tree language* of a pair of synchronous local TDGs is then the set of pairs of trees $\langle \gamma_1, \gamma_2 \rangle$ such that there is a pair of descriptions $\langle \phi_1, \phi_2 \rangle$ in the description language and γ_1 is a minimal tree of ϕ_1 and γ_2 a minimal tree of ϕ_2. The synchronization relation σ is not considered in this case. It might be interesting to view σ also as a restriction on the choice of new equivalences when constructing a pair of minimal trees. However, I suspect that this would influence the result of Prop. 4.11, page 158, which says that the synchronization with a second local TDG does not increase the generative capacity of local TDGs. The problem

in the case of minimal trees is that the process of constructing a minimal tree is non-local in the sense that the choice of the names for new equivalences is not restricted by a locality restriction similar to the one for the choice of node name equivalences in a local derivation step. Since the main motivation for the work presented in this dissertation is the desire to generate suitable underspecified representations, I will leave this issue aside and define the pair of minimal trees for a pair of descriptions independent from the synchronization between the node names of these descriptions.

5.2 Quantifier scope

In this section I will sketch two important approaches proposed for quantifying noun phrases, namely the proper treatment of quantification (PTQ) in Montague 1974 (see also Dowty et al. 1981 and Morrill 1994 for introductions to PTQ), and the quantifier storage approach in Cooper 1983.

5.2.1 Montague's treatment of quantifiers

A first question when dealing with quantifiers concerns the nature of the denotation of noun phrases such as *une femme ('a woman')* and *chaque homme ('every man')*. Clearly, these noun phrases cannot denote single individuals. An alternative would be the assumption that noun phrases denote sets of individuals, e.g. *chaque homme* denotes the set of all men.

(14) a. chaque homme chante
every man sings

b. un homme chante
a man sings

The truth conditions of a sentence such as (14)a. could be stated in terms of the set of all men and set of all individuals that are singing: (14)a. is true iff the first set is a subset of the second set. However, the problem with this approach is that in (14)b. the truth conditions depend also on the set of all men and the set of all individuals that are singing: (14)b. is true iff the intersection of these two sets is not empty. *un homme* does not denote a single man, since (14)b. does not say about a specific man that this man sings as it is for example the case in (15):

(15) Jean chante

The difference between the truth conditions of (14)a. and b. is caused by the different determiners which occur inside the noun phrases. Therefore it is preferable to have different denotations for the noun phrases in (14)a. and b. depending on the different determiners. But then the denotations of noun phrases cannot be sets of individuals.

As a solution, in his "proper treatment of quantification" (PTQ), Montague (1974) proposes to analyse noun phrases as sets of properties. The denotation of *chaque homme* is the set of all properties that are true for all men, and the denotation of *un homme* is the set of properties that are true for at least one man. The determiner takes a property and gives

a set of properties. In the case of (14)b. the sentence can be interpreted in the following way: there is some individual x such that: x is a man and x is singing. The part "there is some individual x such that: x is P_1 and x is P_2" can be regarded as the denotation of the determiner *un*. This must be applied to two predicates (P_1 and P_2), in this case the denotations of *homme* and *chante*. An analysis of (14)b. is then as follows:

$$\begin{aligned}
[\![\textsf{un}]\!] &= \lambda P_1 \lambda P_2(\text{there is an } x \text{ such that } P_1(x) \text{ and } P_2(x)) \\
[\![\textsf{un homme}]\!] &= [\![\textsf{un}]\!]([\![\textsf{homme}]\!]) \\
&= \lambda P_2(\text{there is an } x \text{ such that } [\![\textsf{homme}]\!](x) \text{ and } P_2(x)) \\
[\![\textsf{un homme chante}]\!] &= [\![\textsf{un homme}]\!]([\![\textsf{chante}]\!]) \\
&= \text{there is an } x \text{ such that } [\![\textsf{homme}]\!](x) \text{ and } [\![\textsf{chante}]\!](x)
\end{aligned}$$

There are several terminologies concerning the interpretations of determiners and of noun phrases. I will follow Barwise and Cooper 1981 by calling the interpretation of a noun phrase *quantifier*. The interpretation of a determiner will be called *quantifying phrase*. A quantifying phrase has two arguments denoting both properties. Applied to these arguments it gives a proposition. Henceforward the first argument of a quantifying phrase is called its *restriction* and the second is called its *body*. In (14)b. for example the restriction of the quantifying phrase un is the predicate homme, and the body is the predicate chante.

Montague's PTQ contains one rule for the combination of a subject NP and an intransitive verb that works as illustrated above for (14)b. Furthermore, another rule combines a transitive verb with an object NP. Transitive verbs take a quantifier and give a predicate, i.e. in this case the verb is applied to the quantifier.

(16) chaque homme aime une femme
every man loves a woman

With these rules, an analysis of (16) is as follows:

$$\begin{aligned}
[\![\textsf{chaque homme}]\!] &= \lambda P_1(\text{for all } x: \text{ if } [\![\textsf{homme}]\!](x), \text{ then } P_1(x)) \\
[\![\textsf{une femme}]\!] &= \lambda P_2(\text{there is an } y \text{ such that } [\![\textsf{femme}]\!](y) \\
&\quad \text{and } P_2(y)) \\
[\![\textsf{aime}]\!] &= \lambda X \lambda z(X(\lambda v(z \text{ loves } v))) \\
[\![\textsf{aime une femme}]\!] &= [\![\textsf{aime}]\!]([\![\textsf{une femme}]\!]) \\
&= \lambda z(\text{there is an } y \text{ such that } [\![\textsf{femme}]\!](y) \\
&\quad \text{and } z \text{ loves } y \\
[\![\textsf{chaque homme aime une femme}]\!] &= \text{for all } x: \text{ if } [\![\textsf{homme}]\!](x), \text{ then there is an } y \\
&\quad \text{such that } [\![\textsf{femme}]\!](y) \text{ and } x \text{ loves } y
\end{aligned}$$

However, with this analysis only the reading of (16) with narrow scope of *une femme* is obtained. (16) has a second reading, namely the wide scope reading of *une femme*: there is an y such that y is a woman and for all x: if x is a man, then x loves y. For wide scope of quantifiers Montague proposes a *quantifying-in* mechanism. Roughly said, the idea is that a sentence can be formed by lowering an NP with denotation α into a position marked by a

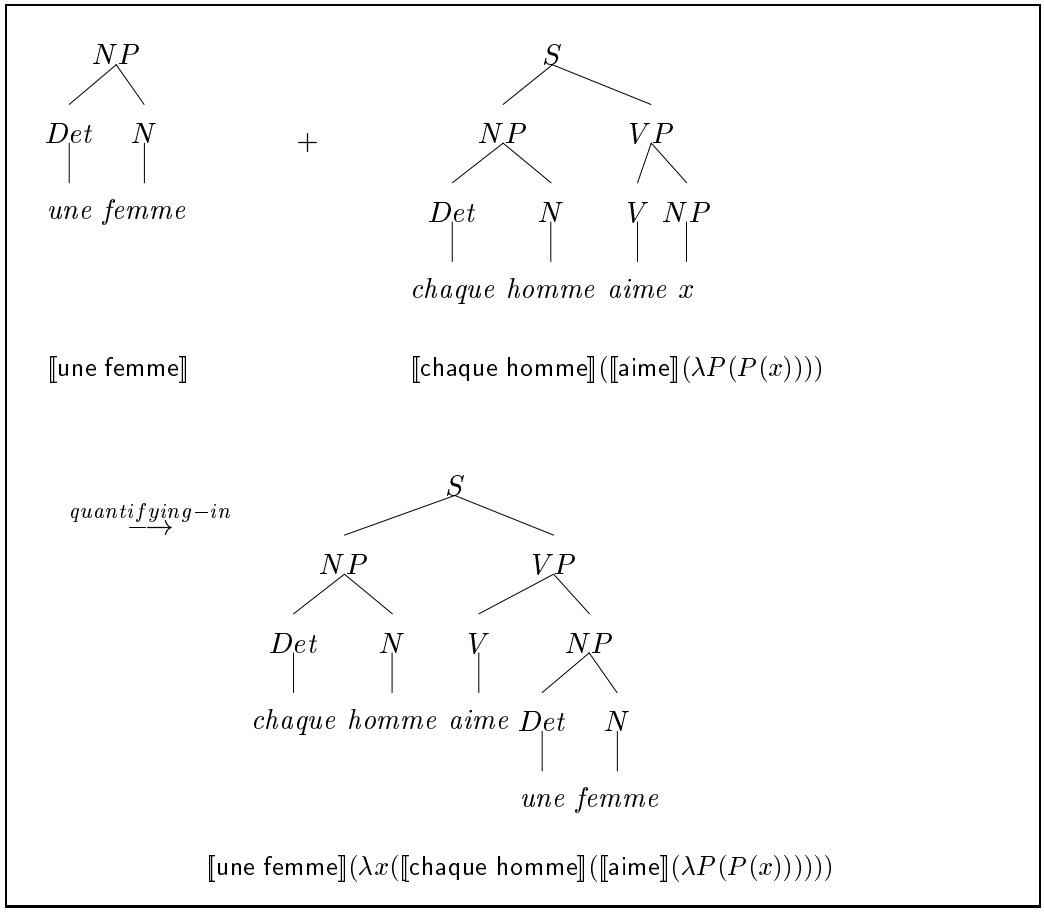


FIGURE 5.4: QUANTIFYING-IN FOR WIDE SCOPE OF *une femme* IN (16)

variable x, and then the sentence is true iff $\alpha(\lambda x\phi)$ is true, where ϕ is the denotation of the sentence with the variable x. Cooper (1983) calls this mechanism the rule of NP-lowering. This name indicates more clearly that Montague's quantifying-in is a syntactic mechanism, i.e. the wide scope reading is accounted for by a specific derivation of the syntactic structure. The difference on the semantic level is a consequence of the syntactic ambiguity.

As an example, Fig. 5.4 shows an application of quantifying-in that leads to the wide scope reading of *une femme* for (16):

$[\![\text{une femme}]\!](\lambda(x)([\![\text{chaque homme}]\!]([\![\text{aime}]\!](\lambda P(P(x))))))$
$\Rightarrow$ there is a y such that $[\![\text{femme}]\!](y)$ and $[\![\text{chaque homme}]\!]([\![\text{aime}]\!](\lambda P(P(y))))$
$\Rightarrow$ there is a y such that $[\![\text{femme}]\!](y)$ and for all x: if $[\![\text{homme}]\!](x)$, then $[\![\text{aime}]\!](\lambda P(P(y)))(x)$
$\Rightarrow$ there is a y such that $[\![\text{femme}]\!](y)$ and for all x: if $[\![\text{homme}]\!](x)$, then $\lambda X \lambda z(X(\lambda v(z \text{ loves } v)))(\lambda P(P(y)))(x)$
$\Rightarrow$ there is a y such that $[\![\text{femme}]\!](y)$ and for all x: if $[\![\text{homme}]\!](x)$, then $\lambda P(P(y))(\lambda v(x \text{ loves } v))$
$\Rightarrow$ there is a y such that $[\![\text{femme}]\!](y)$ and for all x: if $[\![\text{homme}]\!](x)$, then x loves y

5.2.2 Quantifier Storage

As Cooper (1983) points out, Montague's quantifying-in mechanism creates unnecessary syntactic ambiguity. Even examples as in (14) can be derived either by quantifying-in or by the rule for the combination of a subject-NP and an intransitive verb. In the case of (16), there is also more than one syntactic derivation for each of the two scope orders. Furthermore, the syntactic structures derived for different scope orders are the same, they differ only in the way they are derived, and they can also be derived without quantifying-in. For these reasons, Cooper argues that there is no syntactic motivation for including such a rule into the grammar.

Cooper proposes a storage mechanism, known as *Cooper Storage*, a technique of semantic interpretation, in order to eliminate the operation of quantifying-in from the syntactic domain. With this technique, additional wide-scope readings can be obtained without a corresponding NP-lowering in the syntax. Quantifying-in becomes a semantic mechanism.

The idea of this storage mechanism is the following: if a quantifier is supposed to have wide scope, it is put into a store, and later, at an appropriate moment, it is quantified in. The store is part of the meaning, i.e. meaning consists of a semantic expression (e.g. a λ-term) and a store. It gives a way to defer the disambiguation process: quantifiers or logical operators in general with underspecified scope can be put in a kind of temporary store and by doing so, an underspecified representation is obtained. This process is called *quantifier storage*. In the course of a disambiguation called *quantifier retrieval*, the order of the quantifiers is specified.

In Cooper 1983 this storage mechanism is integrated in a Montague style grammar fragment for English. Pollard and Sag (1994) incorporate the Cooper Storage mechanism into Head-Driven-Phrase-Strucure Grammar (HPSG). This storage mechanism is also adopted in the Quasi Logical Form (QLF) of the Core Language Engine (see Alshawi 1990, 1992, Alshawi and Crouch 1992 and Pulman 1994).

The idea of putting operators with underspecified scope into some extra set provides underspecified representations. However the problem is that this approach does not exclude any readings. Quantifiers can stay arbitrarily long (between their storage and their retrieval) in the storage and there is no restriction with respect to the order of retrieval. Therefore by quantifier retrieval, all combinatorially possible combinations of quantifier scope are created. Afterwards the ungrammatical readings must be explicitely excluded.

Therefore, if such a storage mechanism is adopted, extra rules are needed to exclude ungrammatical scope orders. In the disambiguation process for QLF for example such rules are explicitely stated.

Clearly, it would be preferable if the ungrammaticality of certain scope orders was already predicted by the form of the underspecified representation without the necessity of any extra rules.

5.2.3 Constraints for quantifier scope

In this subsection I will list some constraints for quantifier scope that hold universally. Besides these strict rules there are a lot of preferences that should be taken into account in a disambiguation process. But this is not within the scope of this work.

The strict constraints for quantifier scope should already hold for the readings of an underspecified representation, since they depend only on the syntactic structure and the semantic (i.e. the logical) representation. Later, in Section 5.5, we will see that these restrictions are respected within the analysis of quantifier scope ambiguities presented in this chapter.

I will distinguish two kinds of restrictions, *island constraints* and *logical scope restrictions.*

Island constraints

The islands pointed out by Ross (1968) for wh-movement are also relevant for quantifier raising (see Fauconnier 1976 and Rodman 1976). Relative clauses and coordinations give rise to island constraints for quantifier scope.

Coordination constraint: A quantifier cannot be raised out of a coordination.

This means that a quantifier occuring in the second conjunct of a coordination cannot have scope over the first conjunct. As an example consider (17):

(17) a. chaque grenouille est enchantée et une princesse arrive
each frog is enchanted and a princess arrives

b. [chaque grenouille]$_i$ est enchantée et une princesse la$_i$ jete contre le mur
[each frog]$_i$ is enchanted and a princess throws it$_i$ against the wall

c. une grenouille enchantée veut être sauvée et chaque princesse arrive
an enchanted frog wants to be saved and each princess arrives

In (17)a. the quantifier *chaque grenouille* occurs in the first conjunct and can have wide scope: one can interprete (17)a. such that for each enchanted frog, a princess arrives. This is shown in (17)b. In contrast to this, (17)c. does not allow a wide scope reading of the quantifier *chaque princesse.* The interpretation of (17)c. is that there is one single frog that is enchanted and wants to be saved, and for this single frog each princess arrives (probably to throw him against the wall).

The second constraint says that relative clauses are islands with respect to quantifier scope:

Relative clause constraint: A quantifier cannot be raised out of a relative clause.

In other words, a quantifier occurring in a relative clause cannot have scope over the element modified by this relative clause.

(18) chaque homme qui aime une femme est heureux
each man who loves a woman is lucky

In (18) for example, a wide scope reading of *une femme* is not possible.

Logical restrictions

Logical restrictions for quantifier scope are pointed out by Hobbs and Shieber (1987) (see also Alshawi 1992). The following constraint holds for the relative scope of quantifiers:

Logical rule: A quantifier A that is not in the restriction of a quantifier B and that occurs within the scope of B cannot outscope any of the quantifiers in the restriction of B.

This formulation of the logical rule is taken from Alshawi 1992. The logical rule concerns examples such as (19):

(19) every representative of a company saw most samples

In (19) there are three quantifiers and therefore, if each permutation was allowed for the scope order, there would be $3! = 6$ readings. This is not the case however because the scope order where *every representative* has scope over *most samples* and *most samples* has scope over *a company* is not possible.

This restriction is called *logical* because it follows from the logical structure of the sentence: if *every representative* is not outscoped by *a company*, then the restriction of the quantifying phrase *every* must be a predicate of the following form:

$\lambda r(\mathsf{a}(\mathsf{company})(\lambda c(\mathsf{representative\text{-}of}(c)(r))))$

(The quantifier phrase a has an existential interpretation similar to un.)

Then there are only two possibilities for the quantifier *most samples*: either is is in the scope of *every representative*, and then it must be in the body of *every* which means that *most samples* is outscoped by both, *every representative* and *a company*. Or *most samples* has wide scope over *every representative* (and then also over *a company*).

An example where some scope orders are excluded by the relative clause constraint and some are excluded by the logical rule, is given in (20):

(20) chaque élève lit un livre qu'un professeur recommande
each student reads a book that a professor recommends

Here we have three quantifers, *chaque élève*, *un livre* and *un professeur*, but not all $3! = 6$ scope orders are possible. The quantifier *un professeur* may not have scope over *un livre* because of the relative clause constraint. Furthermore, *chaque élève* may not be in the scope of *un livre* and at the same time outscope *un professeur*. This is excluded by the logical rule. Therefore (20) has only the following two readings:

1. $\mathsf{un}(\lambda x((\mathsf{livre}(x)) \wedge (\mathsf{un}(\mathsf{professeur})(\lambda y(\mathsf{recommander}(x)(y))))))$
$(\lambda z(\mathsf{chaque}(\mathsf{eleve})(\lambda v(\mathsf{lire}(z)(v)))))$

2. $\mathsf{chaque(eleve)}(\lambda v(\mathsf{un}(\lambda x((\mathsf{livre}(x)) \wedge (\mathsf{un(professeur)}(\lambda y(\mathsf{recommander}(x)(y))))))$
$(\lambda z(\mathsf{lire}(z)(v))))$

In the Cooper Storage mechanism neither island constraints nor logical restrictions are taken into account (at least in the general quantifier storage approach). In the general disambiguation process, island constraints are not respected and for scope orders excluded by the logical rule, expressions containing free variables are generated.

5.3 Architecture of the local TDG for the semantics

In the following two sections I will propose a syntax-semantics interface based on synchronous local TDGs. This system covers a fragment of French. The architecture of the system is such that one local TDG G_{syn} describes the syntactic structure of French expressions, and a second local TDG G_{sem} gives a truth-conditional semantics for these expressions. We will see that the relation between syntax and semantics is less close than in a Montagovian system. This enables one to account for wide scope readings of quantifiers without producing unnecessary syntactic ambiguities and without an additional storage mechanism. Furthermore, in Section 5.5 we will see that underspecified representations for quantifier scope ambiguities are generated in such a way that island constraints are respected because of the locality of the local TDG formalism. The incorporation of this locality restriction into TDGs in Chapter 4 was motivated by the desire to show semilinearity, i.e. by reasons independent from island constraints.

In this section, I will discuss the architecture of G_{sem}. The developement of G_{sem} comprises the following tasks:

- The design of a logic (syntax and semantics) describing the truth conditions. The string language of G_{sem} then contains expressions of this logic.
- The nonterminal symbols of G_{sem} must be chosen. This could be types or some kind of syntactic categories ('syntactic' with respect to the truth conditional logic) or names of semantic operations.
- Depending on the logic and the nonterminals, the form of the minimal trees and the elementary descriptions must be defined.

There are two different alternatives with respect to the architecture of G_{sem}:

1. Either semantic interpretations are defined only for the logic, i.e. for the expressions in the string language of G_{sem}. Then the trees of the tree language are only syntax trees for logical expressions, they do not have a meaning themselves. In this case the nonterminals might be symbols for syntactic categories, e.g. types or classes of types.
2. Or the semantic interpretation depends on the whole minimal trees: first interpretations of atomic expressions in the logic are given. In each tree for a more complex logical expression the internal nodes are labelled by specifications of operations (e.g. functional application), perhaps together with type specifications. Then the interpretation of a complex expression depends on the tree of this expression. The operation specified in the root of this tree is applied to the interpretations of the subtrees.

In both cases, the derivation process in the local TDG (i.e. the derivation of tree descriptions) is purely syntactical. The semantics is later given either for the trees in the tree language or for the strings yielded by these trees.

I choose the first alternative, i.e. I will regard the trees of the tree language only as syntax trees of some logic expressions. This decision is based on the following considerations:

One of the advantages of TDGs (and also of TAGs) is that we have an extended domain of locality, i.e. that the elementary descriptions can describe more than one node and its daughters as it is the case in context-free grammars. Therefore the semantics can be relatively independent from word order phenomena, in contrast to most frameworks in the Montagovian tradition, e.g. the approach proposed by Muskens (1995) or the system described by von Stechow and Nohl (1995), where each phrase structure rule must be connected with a semantic operation. This independence from the syntax allows us to choose elementary descriptions in such a way that it is possible to use functional application as the only semantic operation (if λ-abstraction is possible). Rules such as Montague's quantifying-in rule are not necessary here because there need not be a direct correspondence between the syntactic position of a lexical item and its position in the semantic expression. It can be interpreted at a position that is different from its surface position. Clearly, if functional application is the only operation, then it is not necessary to specify any operations for internal nodes. Therefore it is possible to choose the simpler alternative by defining a truth conditional semantics only for the string language and not for the whole tree language.

This decision means that the following must be done to develop a TDG G_{sem} for the semantics that is synchronized with the syntactic TDG G_{syn}: first a logic must be defined that should contain all expressions of the string language of G_{sem}. After that for each elementary description of G_{syn} the contribution of this description to the semantic interpretation of the whole sentence must be specified in a syntactic way. In other words it must be specified which part of the syntactic structure of the truth-conditional logical expression for a whole sentence is determined by the description of G_{syn}. According to this a corresponding elementary description in G_{sem} must be introduced and connected with the elementary description in G_{syn} by the synchronization relation.

In the course of this section, the logic for the semantic expressions is defined and the nonterminals (attribute value pairs) are chosen that are necessary for the fragment presented in this chapter.

5.3.1 Syntax of semantic expressions

In this subsection, the syntax of the string language of G_{sem}, i.e. of the truth-conditional logic is introduced.

As a truth-conditional logic for the semantics I choose a classical typed first order logic allowing λ-abstraction. Basic types are individuals, truth values and situations. This means that the logic is a kind of intensional typed lambda calculus in the style of Montague's Intensional Logic (IL) (see Dowty et al. 1981 and Carpenter 1997).

In contrast to Montague's IL, only individual variables are allowed, the logic is a typed first order logic. The logic is called $TY(2)$ indicating that there are two other basic types besides truth values.

Starting from the basic types, types are inductively defined in the usual way:

Definition 5.1 *(Types of $TY(2)$)*

1. *Basic types of $TY(2)$ are e, t and s.*
2. *If T_1 and T_2 are types of $TY(2)$, then $\langle T_1, T_2\rangle$ is also a type of $TY(2)$.*
3. *Nothing else is a type of $TY(2)$.*

Expressions (terms) in $TY(2)$ are words on an alphabet containing constants of all types, individual variables, the λ-abstractor and brackets. Henceforth I assume that V is a set of (individual) variables and for all types T, K_T is a set of constants of type T.

Terms in $TY(2)$ must be such that they are well formed with respect to types, i.e. the argument of an expression of type $\langle T_1, T_2\rangle$ must be of type T_1. This restriction is a syntactical one, functional applications that are excluded because the types do not fit are already excluded in the syntax, i.e. such terms cannot occur.

Definition 5.2 *(Terms in $TY(2)$)*

1. *For each type T, each constant in K_T is a term of type T.*
2. *Each variable in V is a term of type e.*
3. *For all types T_1, T_2: if τ_1 is a term of type $\langle T_1, T_2\rangle$ and τ_2 a term of type T_1, then $\tau_1(\tau_2)$ is a term of type T_2.*
4. *For each type T: if τ is a term of type T and x is a variable, then $\lambda x(\tau)$ is a term of type $\langle e, T\rangle$.*
5. *Nothing else is a term in $TY(2)$.*

5.3.2 Semantics of $TY(2)$

In this subsection I will give a model-theoretic semantics in the traditional way for the terms defined above.

A model consists of objects where each object is of a (unique) type. Furthermore there is an interpretation function from constants to objects of appropriate type in the model. In a model, only the objects of basic types are explicitely specified. All other objects (of some type $\langle T_1, T_2\rangle$) are inductively defined as functions from one set of objects to another.

Definition 5.3 *($TY(2)$-model)*
A $TY(2)$-model is a tuple $\langle D_t, D_e, D_s, I\rangle$ such that:

1. *D_t, D_e and D_s are pairwise disjoint nonempty sets of objects such that:*
   $D_t = \{true, false\}$ is called the set of truth values. D_e is a finite set and is called the set of individuals, and D_s is called the set of situations.
2. *For all types T_1, T_2, the set of objects of type $\langle T_1, T_2\rangle$ is $D_{\langle T_1, T_2\rangle} := D_{T_2}^{D_{T_1}}$ (set of all functions from D_{T_1} to D_{T_2}).*
3. *I is a function from constants to objects such that for each type T and each $c_T \in K_T$: $I(c_T) \in D_T$.*

The interpretation of a term with respect to a given model M depends also on the interpretation of the variables. This is given by a variable assignment function g that maps variables to individual constants.

Definition 5.4 *(Interpretation)*
Let $M = \langle D_t, D_e, D_s, I\rangle$ *be a model,* $g : V \to D_e$ *a variable assignment function. The interpretation of a term* τ *wrt* M *and* g, $[\![\tau]\!]^g_M$ *is inductively defined:*

(a) For all constants c: $[\![c]\!]^g_M = I(c)$.
(b) For all variables x: $[\![x]\!]^g_M = g(x)$.
(c) For all terms of the form $\tau_1(\tau_2)$: $[\![\tau_1(\tau_2)]\!]^g_M = [\![\tau_1]\!]^g_M([\![\tau_2]\!]^g_M)$.
(d) For all terms $\lambda x(\tau)$ *with* τ *of type* T, $[\![\lambda x(\tau)]\!]^g_M$ *is the (unique) function* $f \in D_{\langle e,T\rangle}$ *with* $f(a) = [\![\tau]\!]^{g[x:=a]}_M$ *for all* $a \in D_e$, *where* $[\![\tau]\!]^{g[x:=a]}_M$ *is defined as follows:*
$[\![\tau]\!]^{g[x:=a]}_M := [\![\tau]\!]^h_M$ *for the assignment* h *with* $h(x) := a$ *and* $h(y) := g(y)$ *for all* $y \neq x$.

Each minimal tree of a description generated in G_{sem} yields a unique string and therefore has a unique interpretation. Since descriptions may have several minimal trees, the interpretation of a description might be underspecified in the sense that there are several corresponding strings in the string language with different interpretations. I will define the interpretation of a description generated by G_{sem} as the set of all interpretations of strings yielded by minimal trees of this description.

Definition 5.5 *(Interpretation of a description)*
Let ϕ *be a description in* $L_D(G_{sem})$.
The interpretation of ϕ *in a model* M *wrt an assignment function* g *is*
$[\![\phi]\!]^g_M := \{[\![\tau]\!]^g_M \mid \tau = yield(B)$ *for a minimal tree* B *of* $\phi\}$.

5.3.3 Nonterminal symbols in G_{sem}

Before coming to the elementary descriptions in G_{sem} it must be decided which kind of nonterminal symbols should be used.

As the trees in the tree language should describe the syntax of logical expressions in $TY(2)$, nonterminals should be names of some kind of syntactic categories. It might be a problem to use types as nonterminals, since there is (at least theoretically) an infinite number of types. Furthermore, names for syntactic categories are preferable for reasons of readability. Therefore I will summarize several types under one semantic category.

There will be an attribute CAT with these semantic categories as possible values. The categories used for the fragment presented in this chapter are

- *prop* for propositions (type $\langle s, t\rangle$),
- *ind* for individuals (type e),
- *pred* for predicates that take individuals as arguments and that give a proposition (types $\langle e, \langle s, t\rangle\rangle$ or $\langle e, \langle e, \langle s, t\rangle\rangle\rangle$ or $\langle e, \langle e, \langle e, \langle s, t\rangle\rangle\rangle\rangle$ etc.),
- *ppred* for predicates taking a proposition as argument and and giving a predicate of category *pred* (types $\langle \langle s, t\rangle, \langle e, \langle s, t\rangle\rangle\rangle$, etc.),
- *mod* for objects of a type $\langle T, T\rangle$, where T is a type,
- *qp* for quantifying phrases (type $\langle \langle e, \langle s, t\rangle\rangle, \langle \langle e, \langle s, t\rangle\rangle, \langle s, t\rangle\rangle\rangle$),
- *quant* for quantifiers (type $\langle \langle e, \langle s, t\rangle\rangle, \langle s, t\rangle\rangle$)
- *con* for objects that take an argument of some type T and then give a modifier of type $\langle T, T\rangle$ (e.g. $\langle \langle s, t\rangle, \langle \langle s, t\rangle, \langle s, t\rangle\rangle\rangle$)

5.4 Elementary configurations

In this section, the synchronous local TDGs $\langle G_{syn}, G_{sem}, \Sigma \rangle$ for a basic fragment of the syntax-semantics interface are presented. More precisely, the elementary configurations are presented, i.e. the triples $\langle \psi_1, \psi_2, \sigma \rangle$ where ψ_1 is an elementary description in G_{syn}, ψ_2 an elementary description in G_{sem} and $\sigma \subseteq node(\psi_1) \times node(\psi_2)$ a relation. In particular, elementary configurations for proper names, common nouns, verbs, adjectives (intersective and non-intersective adjectives), quantifiers and relative clauses are introduced. For each lexical item, the elementary description in G_{syn} and the one in G_{sem} together with their synchronization relation must be specified.

In order to render the graphical representations more readable, henceforth I will adopt the following notations: elementary descriptions in G_{syn} are named $\psi, \psi_1, \ldots$ and elementary descriptions in G_{sem} are named $\chi, \chi_1, \ldots$. Derived descriptions in G_{syn} are named $\phi, \phi_1, \ldots$ and derived descriptions in G_{sem} are named $\varphi, \varphi_1, \ldots$. Node names occurring in G_{syn} are named $k_1, k_2, \ldots$ whereas node names in G_{sem} are named $n_1, n_2, \ldots$. Furthermore, in the figures, the description on the left side is always the one in G_{syn} and the one on the right side is the corresponding description in G_{sem}. As usual, marked node names are those with an asterisk. With respect to the labels, If *cat* is the only attribute, then for simplicity $\textsc{cat} : value$ is replaced by *value*.

5.4.1 Verbs and proper names

First, I will introduce the elementary configurations that are needed to analyse simple sentences such as

(21) a. Marie dort
Marie sleeps

b. Jean aime Marie
Jean loves Marie

Verbs

In order to derive a description in G_{syn} for the sentences in (21), one has to start with the elementary description for the verb, since this is the anchor of the whole sentence. Therefore, I will start by introducing elementary configurations for intransitive and transitive verbs.

The intransitive and transitive verbs in (21) denote objects of category *pred*. The verb *dort* takes an individual argument and gives a proposition, i.e. its denotation is of type $\langle e, \langle s, t \rangle \rangle$, and *aime* takes two individual arguments and gives a proposition, i.e. its denotation is of type $\langle e, \langle e, \langle s, t \rangle \rangle \rangle$. Since these verbs are anchors of whole sentences, the minimal names of the corresponding elementary descriptions in the syntax TDG G_{syn} have the category S. Accordingly, the semantic category of the minimal names of the related elementary descriptions in G_{sem} must be *prop* because expressions anchored by verbs denote propositions. Elementary configurations of verbs selecting for individual arguments are shown in Fig. 5.5.

The synchronization relation between the node names of the two descriptions of an elementary configuration mainly expresses correspondences between argument slots. In the case

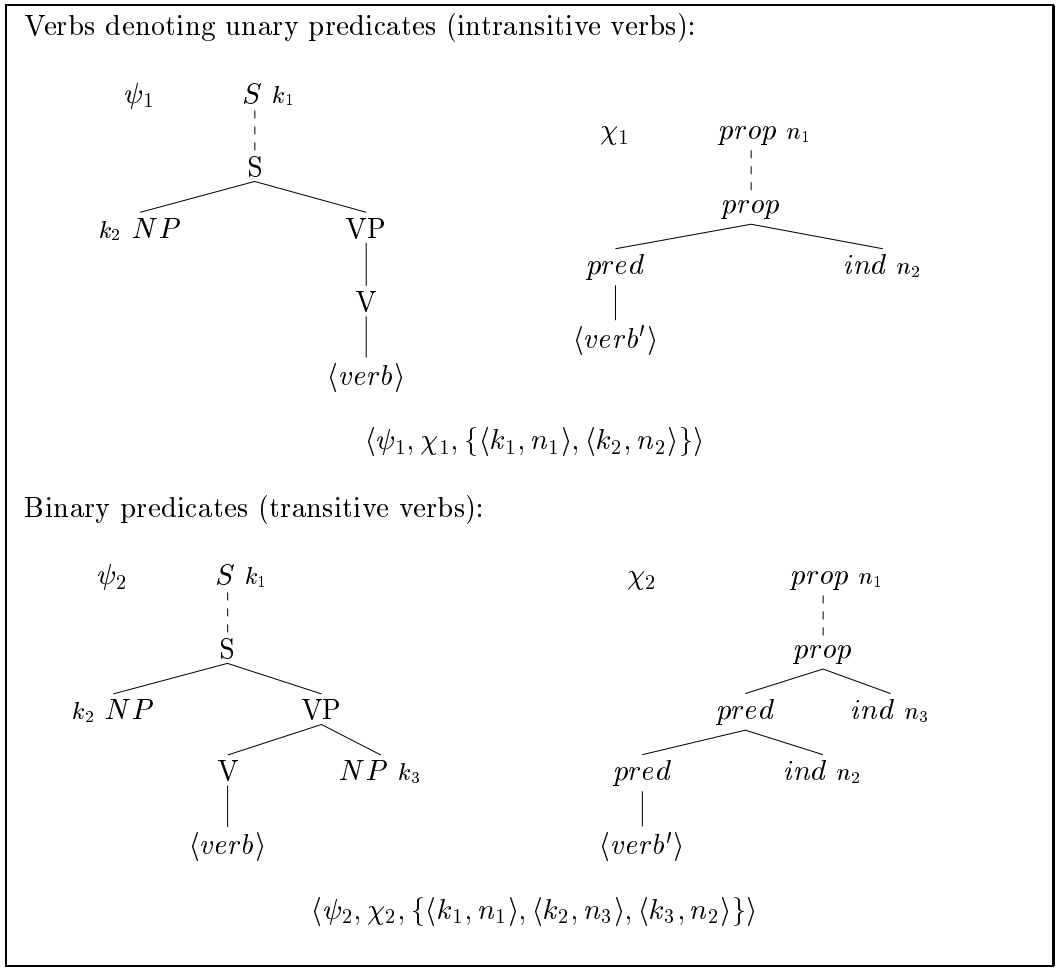


FIGURE 5.5: ELEMENTARY CONFIGURATIONS FOR VERBS

of an intransitive verb, the predicate denoted by the verb is applied to an individual argument. This argument corresponds to the subject-NP. (This does not necessarily mean that it is denoted by the subject-NP. If this NP is a quantifier, then it does not denote an individual but it introduces an argument to the verb.) Such correspondences between argument slots are expressed by the pair $\langle k_2, n_2 \rangle$ in the elementary configuration for intransitive verbs and by the pairs $\langle k_2, n_3 \rangle$ and $\langle k_3, n_2 \rangle$ in the elementary configuration for transitive verbs.

The pairs $\langle k_1, n_1 \rangle$ in the two elementary configurations express a denotation relation: the truth-conditional logical expression dominated by n_1 represents the denotation of the syntactic structure under k_1.

Proper names

Proper names shall be treated as rigid designators (see Kripke 1972), i.e. they are supposed to denote always the same individual, independently from the situation. Therefore proper names are constants of type e.

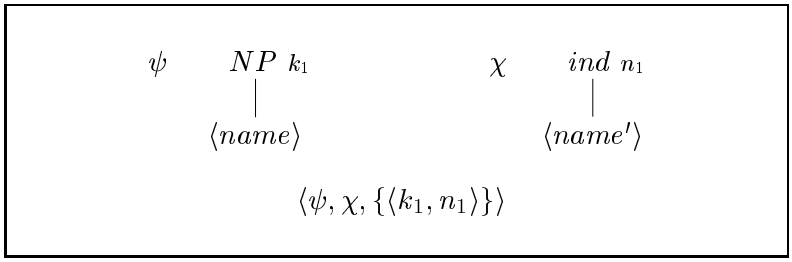


FIGURE 5.6: ELEMENTARY CONFIGURATIONS FOR PROPER NAMES

For proper names, I adopt elementary configurations as shown in Fig. 5.6. The relation between k_1 and n_1 signifies not only that n_1 dominates the denotation of the expression under k_1. Furthermore, it makes sure that, when adding ψ as an argument to a syntactic structure (anchored for example by a transitive verb), the related descriptions χ must be inserted at the corresonding argument slot in the semantics. This must be the case because the two argument slots are also related to each other and because of the third condition for derivation in synchronous local TDGs on page 173.

As an example consider the derivation shown in Fig. 5.7 and 5.8 for (21)b. This derivation starts with the elementary configuration for the transitive verb *aime*. In a first derivation step, the elementary configuration $\langle \psi_2, \chi_2, \{\langle k_4, n_4\rangle\}\rangle$ for *Jean* is added. Since *Jean* precedes the verb, the minimal name k_4 of ψ_2 must be identified with the leaf name k_2 in ψ_1, i.e. the equivalence $k_2 \approx k_4$ must be added in G_{syn}. In G_{sem}, an equivalent name for the minimal n_4 in χ_2 must be found. This must be a leaf name with category *ind*, i.e. either n_2 or n_3. Since n_4 is related to k_4, and n_3 is the only name related to k_2, n_3 must be chosen and the equivalence $n_3 \approx n_4$ is added in G_{sem}. In a second derivation step, in a similar way the elementary configuration for *Marie* is added with the new equivalences $k_3 \approx k_5$ and $n_2 \approx n_5$. The description φ_2 derived in G_{sem} has one minimal tree. If subtrees are viewed as bracketed expressions, the string yielded by this minimal tree is ((aimer)(Marie))(Jean).

In order to derive a configuration for (21)b., it is obligatory to start with the elementary configuration of the verb. But the two elementary configurations for *Jean* and *Marie* can be added in any order, one could as well start with adding the descriptions for *Marie* and in the second step adding those for *Jean*. The result would be the same.

5.4.2 Quantifiers

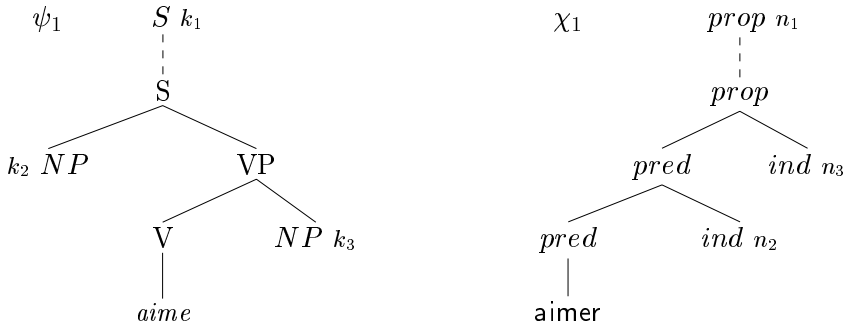
(22) a. chaque fille chante
every girl sings

b. Jean achète une maison
Jean buys a house

In this subsection, elementary configurations for quantifying phrases and common nouns are introduced, such that configurations for the sentences in (22) can be derived.

Since the predicate denoted by the noun in an NP such as *chaque fille* is an argument (the restriction) of the quantifying phrase, an analysis of such an NP must start with the

Elementary configuration for *aime*:
$\kappa_1 = \langle \psi_1, \chi_1, \{\langle k_1, n_1\rangle, \langle k_2, n_3\rangle, \langle k_3, n_2\rangle\}\rangle$ with



Elementary configuration for *Jean*: $\kappa_2 = \langle \psi_2, \chi_2, \{\langle k_4, n_4\rangle\}\rangle$ with

First derivation step: $\kappa_1 \overset{\kappa_2}{\Rightarrow} \kappa$ with $\kappa = \langle \phi_1, \varphi_1, \sigma\rangle$,
$\phi_1 = \psi_1 \wedge \psi_2 \wedge k_2 \approx k_4, \varphi_1 = \chi_1 \wedge \chi_2 \wedge n_3 \approx n_4$
and $\sigma = \{\langle k_1, n_1\rangle, \langle k_2, n_3\rangle, \langle k_3, n_2\rangle, \langle k_4, n_4\rangle\}$

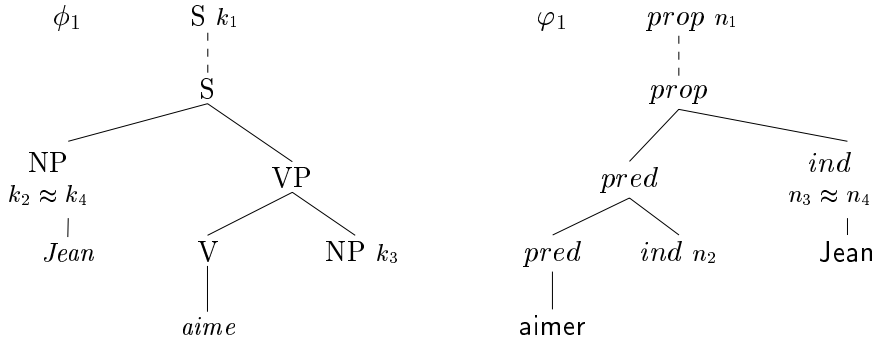
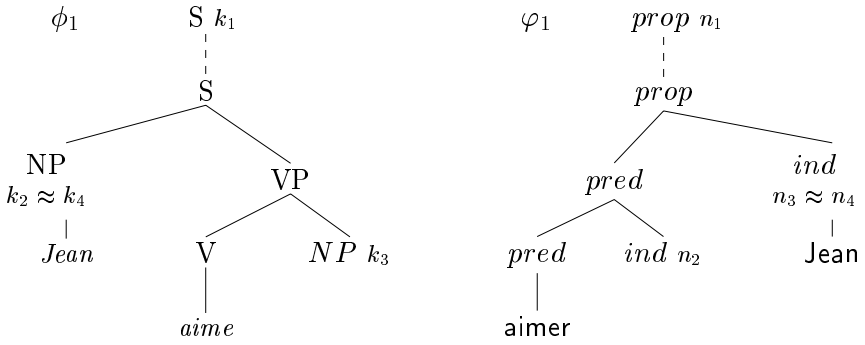


FIGURE 5.7: FIRST DERIVATION STEP FOR *Jean aime Marie*

$\kappa = \langle \phi_1, \varphi_1, \sigma \rangle$ with $\sigma = \{\langle k_1, n_1 \rangle, \langle k_2, n_3 \rangle, \langle k_3, n_2 \rangle, \langle k_4, n_4 \rangle\}$ and



Elementary configuration for *Marie*: $\kappa_3 = \langle \psi_3, \chi_3, \{\langle k_5, n_5 \rangle\} \rangle$ with

Second derivation step: $\kappa \overset{\kappa_3}{\Rightarrow} \kappa'$ with $\kappa' = \langle \phi_2, \varphi_2, \sigma' \rangle$,
$\phi_2 = \phi_1 \wedge \psi_3 \wedge k_3 \approx k_5, \varphi_2 = \varphi_1 \wedge \chi_3 \wedge n_2 \approx n_5$
and $\sigma' = \{\langle k_1, n_1 \rangle, \langle k_2, n_3 \rangle, \langle k_3, n_2 \rangle, \langle k_4, n_4 \rangle, \langle k_5, n_5 \rangle\}$

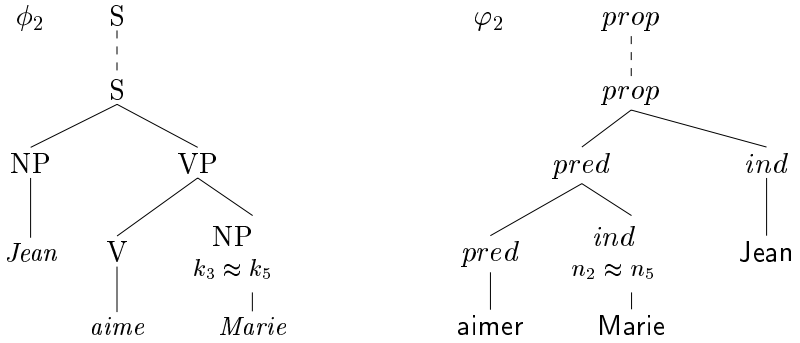


FIGURE 5.8: SECOND DERIVATION STEP FOR *Jean aime Marie*

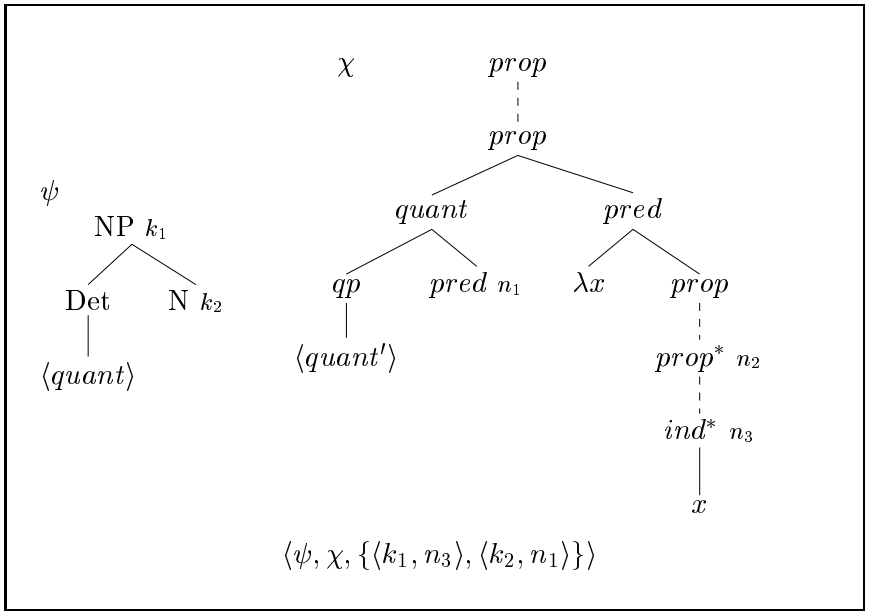


FIGURE 5.9: ELEMENTARY CONFIGURATIONS FOR QUANTIFYING PHRASES

elementary configuration for the quantifying phrase, and then the elementary configuration for the noun can be added.

Quantifying phrases

Following Montague, I will analyse quantified NPs as sets of predicates, and determiners (or quantifying phrases) as objects that take a predicate and give a set of predicates. Consequently, quantifying phrases are of type $\langle\langle e, \langle s, t\rangle\rangle, \langle\langle e, \langle s, t\rangle\rangle, \langle s, t\rangle\rangle\rangle$.

The elementary descriptions of quantifying phrases in G_{sem} must be such that the following holds: when the quantifying phrase is applied to the noun predicate of the NP, the result must be a quantifier. This quantifier can then be applied to a second predicate (the body of the quantifying phrase). This second predicate can be obtained by λ-abstraction of the whole proposition with respect to the argument corresponding to the NP. In other words, first the λ-abstraction of the proposition with respect to this argument is built and then the quantifier corresponding to the argument is applied to the result of the abstraction.

Because of this, for quantifying phrases I propose elementary configurations as shown in Fig. 5.9. The relation between k_2 and n_1 guarantees that the noun added in G_{syn} under k_2 denotes the restriction of the quantifying phrase. Furthermore, with the relation between k_1 and n_3, the body of the quantifying phrase must be a predicate that is applied to the individual corresponding to the whole NP (node name n_3). Each time such an elementary configuration is used, the variable x must be instantiated with a new individual variable.

We will see that these elementary configurations for quantifiers allow us to obtain wide scope readings without any quantifier raising and quantifying-in. The reason is that the relation between syntax and semantics is less close in this system than in a Montagovian

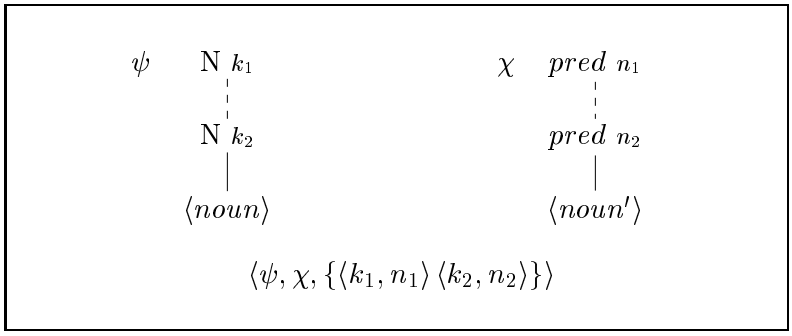


FIGURE 5.10: ELEMENTARY CONFIGURATIONS FOR COMMON NOUNS

framework. The descriptions in G_{sem} can be seen as partial descriptions of a kind of Logical Form.

The quantifying phrases *un* and *chaque* shall have the following usual existential and universal interpretations:

$I(\mathsf{un})$	with	$I(\mathsf{un})(P_1)(P_2) = p$, such that $p(s) = true$ iff there is an $a \in D_e$ with $P_1(a)(s) = true$ and $P_2(a)(s) = true$
$I(\mathsf{chaque})$	with	$I(\mathsf{chaque})(P_1)(P_2) = p$, such that $p(s) = true$ iff for all $a \in D_e$: if $P_1(a)(s) = true$, then $P_2(a)(s) = true$

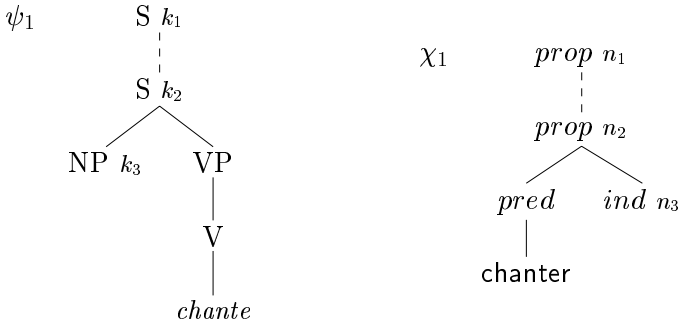
In a similar way the interpretation of other quantifying phrases such as *plusieurs* (*'several'*) and *quelques* (*'some'*) can be defined. I will leave this aside, I assume that for each quantifying phrase there is an appropriate interpretation of type $\langle\langle e, \langle s, t\rangle\rangle, \langle\langle e, \langle s, t\rangle\rangle, \langle s, t\rangle\rangle\rangle$.

Common Nouns

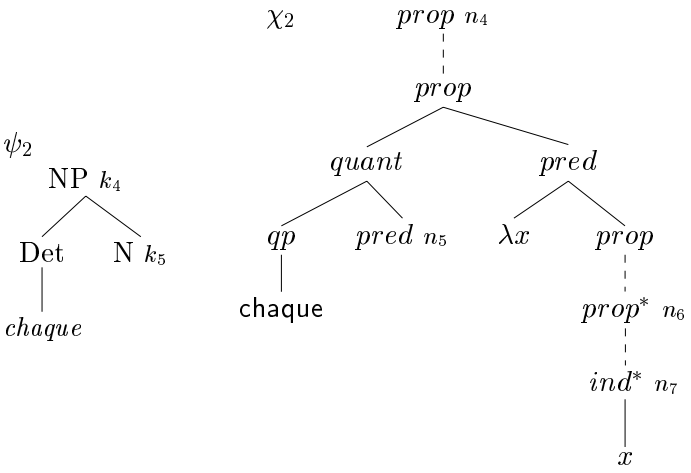
Common nouns like *maison* (*'house'*), *chien* (*'dog'*), *lettre* (*'letter'*) etc. denote unary predicates that take an individual argument, i.e. they are of type $\langle e, \langle s, t\rangle\rangle$ and their category is *pred*. (Relational nouns are not taken into account here.)

Nouns are usually embedded into an NP. Therefore it can be supposed that in a description derived in G_{sem} before adding the elementary description of the noun, there is a node name with category *pred*. This node name is part of the elementary description of the determiner of the NP the noun is embedded in. It must be identified with the minimal name in the elementary description of the noun. According to this, elementary configurations for predicates denoted by nouns are as shown in Fig. 5.10. The dominances between k_1 and k_2 and between n_1 and n_2 account for the fact that the predicate might be modified in further derivation steps (by adjectives or relative clauses). The synchronization relation between k_1 and n_1 guarantees that the noun and its denotation are inserted into corresponding slots in G_{syn} and G_{sem}. The relation of k_2 and n_2 signifies that the interpretation of the semantic expression under n_2 is the denotation of the string under k_2.

Elementary configuration $\kappa_1 = \langle \psi_1, \chi_1, \{\langle k_1, n_1\rangle, \langle k_3, n_3\rangle\}\rangle$ for *chante*:



Elementary configuration $\kappa_2 = \langle \psi_2, \chi_2, \{\langle k_4, n_7\rangle, \langle k_5, n_5\rangle\}\rangle$ for *chaque*:



$\kappa_1 \overset{\kappa_2}{\Rightarrow} \langle \phi_1, \varphi_1, \{\langle k_1, n_1\rangle, \langle k_3, n_3\rangle, \langle k_4, n_7\rangle, \langle k_5, n_5\rangle\}\rangle$ with
$\phi_1 = \psi_1 \wedge \psi_2 \wedge k_3 \approx k_4$ and $\varphi_1 = \chi_1 \wedge \chi_2 \wedge n_1 \approx n_4 \wedge n_2 \approx n_6 \wedge n_3 \approx n_7$

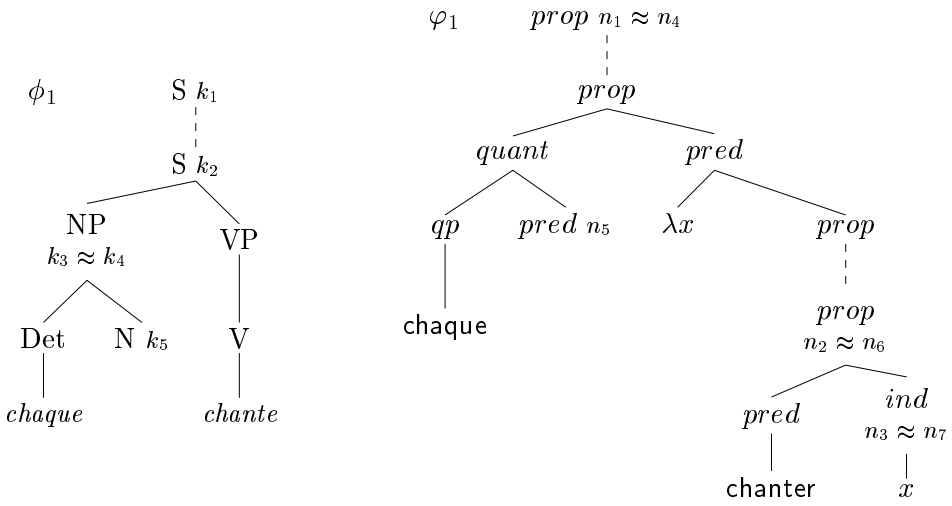
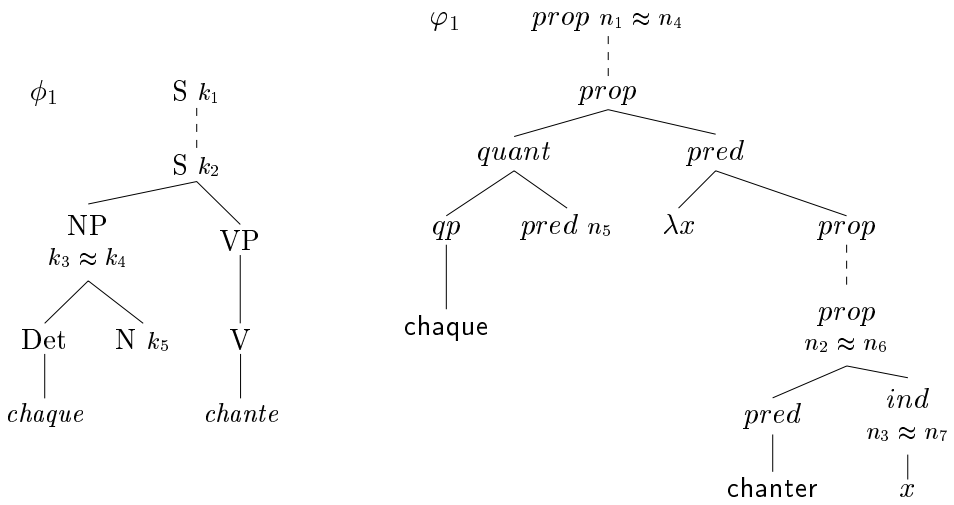
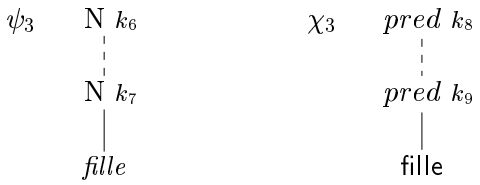


FIGURE 5.11: FIRST DERIVATION STEP FOR *chaque fille chante*

Configuration derived in Fig. 5.11:
$\kappa := \langle \phi_1, \varphi_1, \{\langle k_1, n_1\rangle, \langle k_3, n_3\rangle, \langle k_4, n_7\rangle, \langle k_5, n_5\rangle\}\rangle$ with



Elementary configuration $\kappa_3 = \langle \psi_3, \chi_3, \{\langle k_6, n_8\rangle, \langle k_7, n_9\rangle\}\rangle$ for *fille*:



$\kappa \overset{\kappa_3}{\Rightarrow} \langle \phi_2, \varphi_2, \{\langle k_1, n_1\rangle, \langle k_3, n_3\rangle, \langle k_4, n_7\rangle, \langle k_5, n_5\rangle, \langle k_6, n_8\rangle, \langle k_7, n_9\rangle\}\rangle$ with
$\phi_2 = \phi_1 \wedge \psi_2 \wedge k_5 \approx k_6$ and $\varphi_2 = \varphi_1 \wedge \chi_2 \wedge n_5 \approx n_8$

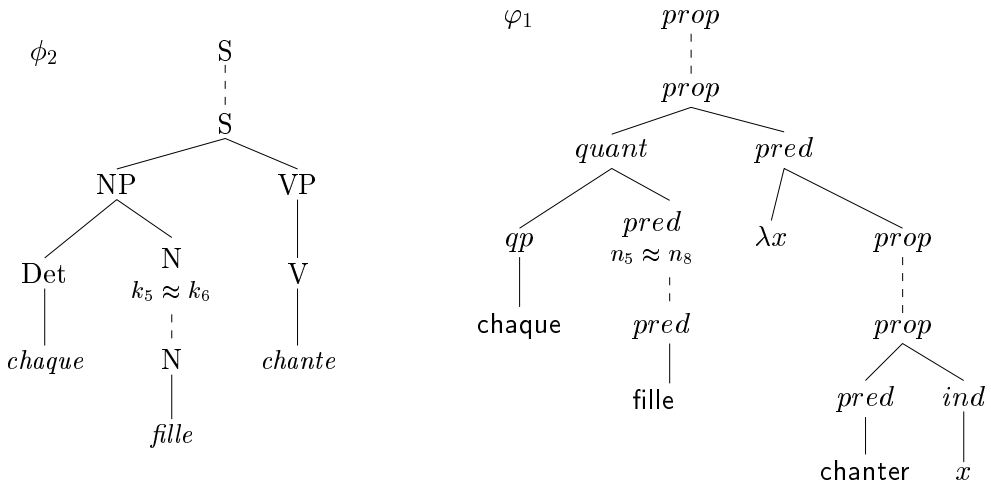


FIGURE 5.12: SECOND DERIVATION STEP FOR *chaque fille chante*

As an example consider the analysis of (22)a. *chaque fille chante* (page 187). The derivation starts with the elementary configuration for the verb *chante*, the anchor of the sentence. In the first step (see Fig. 5.11), the elementary configuration for *chaque* is added, and in the second step (Fig. 5.12), the elementary configuration for *fille* is added. In this case this is the only possible order of the derivation since the quantifying phrase introduces an argument of the verb and the noun introduces an argument of the quantifying phrase. The resulting description φ_2 derived in G_{sem} has exactly one minimal tree and the corresponding interpretation is:

$$\begin{array}{lll} p & = & [\![\mathsf{chaque}(\mathsf{fille})(\lambda x(\mathsf{chanter}(x)))]\!] \\ & \text{with} & p(s) = true \\ & \text{iff} & \text{for all } a \in D_e: \text{ if } [\![\mathsf{fille}]\!](a)(s) = true, \text{ then } [\![\mathsf{chanter}]\!](a)(s) = true \end{array}$$

5.4.3 Complement clauses

In this subsection, I will introduce elementary configurations for verbs selecting for a complement clause such as *croire* in (23):

(23) Marie croit que Jean arrive
Marie believes that Jean arrives

Verbs of category *ppred*, i.e. selecting a propositional argument, can be treated similarly to verbs of category *pred*. The elementary configuration for the finite verb *croit* is shown in Fig. 5.13. In both cases (verbs of category *pred* as in Fig. 5.5 and verbs of category *ppred*), the synchronization relations express the correspondence between syntactic and semantic argument slots.

The verb *croit* creates a so-called oblique or referentially opaque construction. This means that if an expression with the verb *croit* is evaluated in a situation s, then the propositional argument of *croit* might be evaluated in a situation different from s. If (23) for example is true in a situation s, then this does not necessarily mean that *Jean arrive* is also true in s. It signifies only that *Jean arrive* is true in all situations where everything is the case that Marie believes to be true in s.

This property of *croire* (and also *désirer ('wish', 'desire')*, *penser ('think')*, *imaginer ('imagine')* etc.) does not influence the elementary configuration but it concerns the lexical semantics of *croire*: croire is a constant of type $\langle\langle s,t\rangle,\langle e,\langle s,t\rangle\rangle\rangle$ and its interpretation must be something similar to the following (I will not discuss the lexical semantics of verbs such as *croire* here in detail because this is not within the scope of this work):

$$\begin{array}{lll} I(\mathsf{croire}) & \text{with} & I(\mathsf{croire})(q)(x) = p \text{ with } p(s) = true \text{ iff for all } s' \in D_s, \\ & & \text{such that in } s' \text{ everything is true that } x \text{ believes in } s, \\ & & q(s') = true \text{ holds.} \end{array}$$

The elementary configuration of a verb such as *arrive* in (23) that is the anchor of a complement clause differs from those shown in Fig. 5.5 in so far as there is one more name with category S in the syntax and one more name of category *prop* in the semantics. The

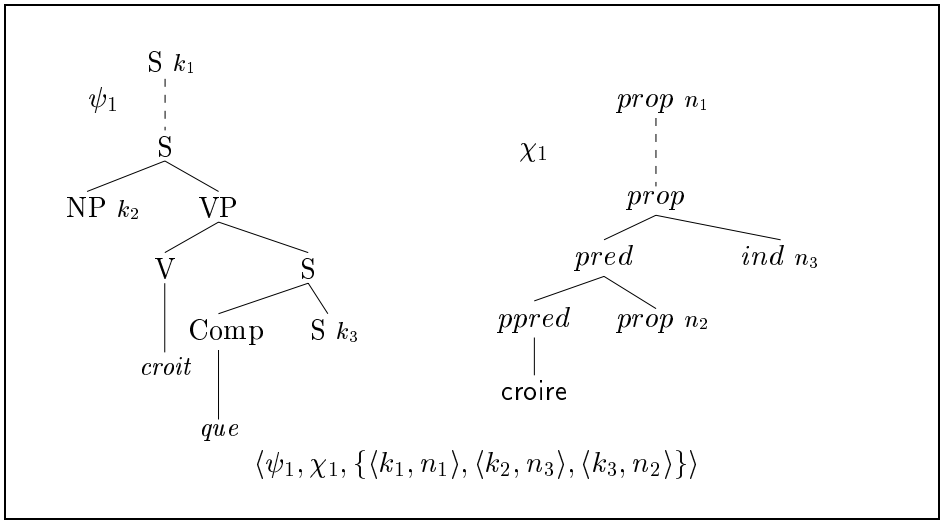


FIGURE 5.13: ELEMENTARY CONFIGURATION FOR *croit ('believes')*

elementary configuration of *arrive* in (23) is shown in Fig. 5.14. In this work I will not deal with the organization of the lexicon, but following proposals made for TAGs (see Abeillé 1990 and Vijay-Shanker and Schabes 1992), I assume that there are "basic" elementary configurations (e.g. those introduced for verbs in Fig. 5.5 and 5.13), whereas other elementary configurations are generated from these basic configurations by lexical rules. Elementary configurations for verbs that are anchor of a complement clause as in Fig. 5.14 are derived via a lexical rule. The same holds for anchors of relative clauses that will be introduced in 5.4.5.

The reason for the additional node name n_4 with category *prop* in χ in Fig. 5.14 is that it must be possible to move quantifiers out of the complement clause, in contrast to relative clauses for example.[2] This contrast is exemplified in (24).

(24) a. une fille croit que chaque grenouille est un prince enchanté
a girl believes that every frog is an enchanted prince

b. le roi a une fille qui aime chaque grenouille enchantée
the king has a daughter who loves every enchanted frog

(24)a. has a reading with wide scope of *chaque grenouille*, i.e. a reading where for each frog there is a girl believing that this frog is an enchanted prince. Consequently complement clauses are no islands for quantifier raising. In (24)b. *chaque grenouille enchantée* is embedded in a relative clause and in this case wide scope of *chaque grenouille enchantée* is not possible.

When adding the elementary configuration for *arrive* $\langle \psi_2, \chi_2, \{\langle k_5, n_5\rangle, \langle k_6, n_6\rangle\}\rangle$ in Fig. 5.14 to the configuration $\langle \psi_1, \chi_1, \{\langle k_1, n_1\rangle, \langle k_2, n_3\rangle, \langle k_3, n_2\rangle\}\rangle$ for *croit* in Fig. 5.13, the new equivalences in G_{sem} are $n_1 \approx n_4$ and $n_2 \approx n_5$. Therefore n_4 is minimal in the resulting

[2] This raising of quantifiers out of complement clauses is similar to wh-movement out of complement clauses as shown in Fig. 3.23 on page 103.

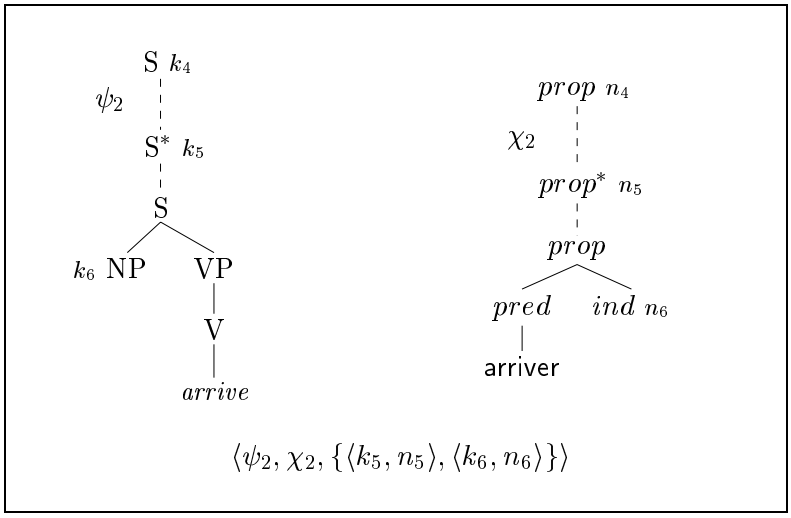


FIGURE 5.14: ELEMENTARY CONFIGURATION FOR *arrive* AS ANCHOR OF A COMPLEMENT CLAUSE

description, and a quantifier added to the complement clause can have wide scope. This phenomenon will be examined more closely when de re – de dicto ambiguities will be considered on page 208 (see Fig. 5.25).

Fig. 5.15 shows the description that is derived in G_{sem} for (23). This description has only one minimal tree, and the string yielded by this tree has the following interpretation:

$[\![\mathsf{croire}(\mathsf{arriver}(\mathsf{Jean}))(\mathsf{Marie})]\!]^g_M(s) = true$
iff for all $s' \in D_s$, such that in s' everything is true that *Marie* believes in s: $[\![\mathsf{arriver}]\!]^g_M(Jean)(s') = true$.

5.4.4 Adjectives

In this subsection I will consider adjectives that modify nouns as in (25):

(25) a. petit papillon bleu
small butterfly blue
small blue butterfly / blue small butterfly

b. futur président
future president

Two kinds of adjectives must be distinguished. On the one hand, adjectives such as *petit* and *bleu* in (25)a. are adjectives that modify a predicate and that are predicates themselves. These are called *intersective* adjectives. Intersective adjectives are for example color adjectives or adjectives as *beau* (*'beautiful'*), *grand* (*'big, huge'*), etc. On the other hand, adjectives such as *futur* in (25)b. also modify a predicate. But, in contrast to *petit* and *bleu* in (25)a., they are not predicates themselves. These are the *non-intersective* adjectives. Because of this

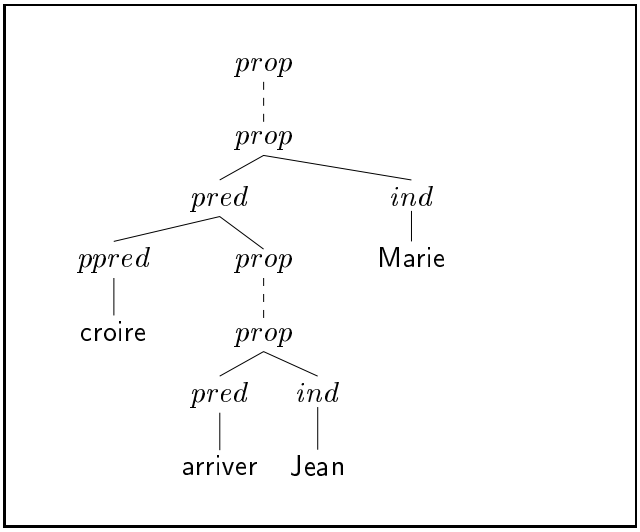


FIGURE 5.15: DESCRIPTION DERIVED FOR (23) *Marie croit que Jean arrive* IN G_{sem}

difference in interpretation intersective and non-intersective adjectives should be assigned different types.

Intersective adjectives

Since intersective adjectives denote predicates, they are of type $\langle e, \langle s, t\rangle\rangle$. Combining such an adjective with a noun yields a predicate that is true for an individual a in a situation s iff both the interpretation of the adjective and the interpretation of the noun are true for a in s. In other words, there are two sets of situations, those situations where the predicate denoted by the adjective is true for a and those where the predicate denoted by the noun is true for a. These two sets of situations are intersected, and this is the reason why these adjectives are called intersective.

The problem now is how to analyse intersective adjectives in the logic $TY(2)$ that allows only functional application and λ-abstraction as operations. Obviously, if intersective adjectives are supposed to denote predicates, then it is not possible to apply an intersective adjective directly to the predicate it modifies. This is excluded because both predicates have the same type. But using λ-abstraction and a constant et (a conjunction) to combine two propositions, an elementary description can be constructed that captures the intersective combination of the two predicates. The idea of the elementary description in G_{sem} for an intersective adjective that modifies a predicate denoted by a noun is the following: first both predicates are applied to a variable x, then the conjunction of the resulting propositions is built, and finally the λ-abstraction of the conjunction with respect to x is obtained.

The combination et of two propositions is the conjunction of the two propositions and its type is $\langle\langle s, t\rangle, \langle\langle s, t\rangle, \langle s, t\rangle\rangle\rangle$. The interpretation of et is:

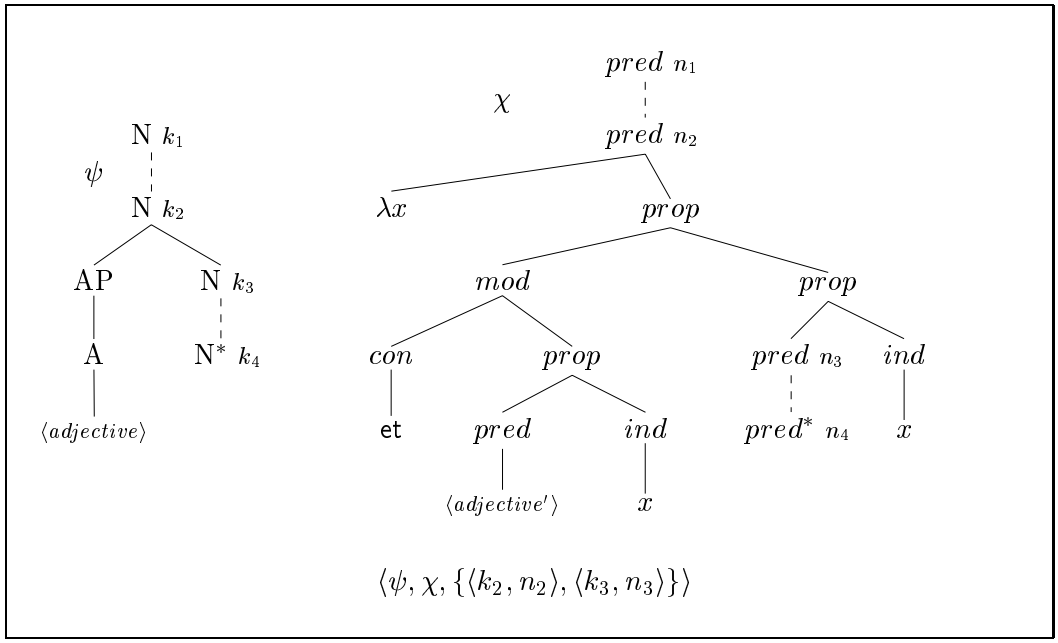


FIGURE 5.16: ELEMENTARY CONFIGURATIONS OF INTERSECTIVE ADJECTIVES PRECEDING THE MODIFIED NOUN

$$I(\mathsf{et}) \in D_{\langle\langle s,t\rangle,\langle\langle s,t\rangle\langle s,t\rangle\rangle\rangle} \quad \text{with} \quad I(\mathsf{et})(p_1)(p_2)(s) = true \text{ iff } p_1(s) = true \text{ and } p_2(s) = true$$

Fig. 5.16 shows the form of elementary descriptions for intersective adjectives preceding the noun as *petit* in (25)a. (As in the case of quantifying phrases, each time such a description is added in a derivation step, the variable x in χ must be instantiated by a new individual variable.) The synchronization signifies that the syntactic constituent under k_2 denotes the predicate under n_2 and the constituent under k_3 denotes the predicate under n_3. For intersective adjectives as *bleu* in (25)a. that follow the noun, the semantic description is the same and the syntactic description differs from ψ in Fig. 5.16 in so far as k_3 and k_4 are left of the node names with AP, A and the adjective (e.g. see Fig. 5.26 for the elementary configuration of *bleue*).

Combining such an elementary description with the elementary description of a noun, the result is the desired interpretation. Fig. 5.17 shows the description that is derived in G_{sem} for the semantics of *papillon bleu* (*'blue butterfly'*). This description has one minimal tree, the yield of this tree is the following term of type $\langle e, \langle s, t\rangle\rangle$:

$$\lambda x_1(\mathsf{et}(\mathsf{bleu}(x_1))(\mathsf{papillon}(x_1)))$$

The interpretation of this term is the intersection of the predicates bleu and maison.

This analysis of intersective adjectives shows that in contrast to systems like Montague's IL, here it is not necessary to use other semantic operations besides functional application.

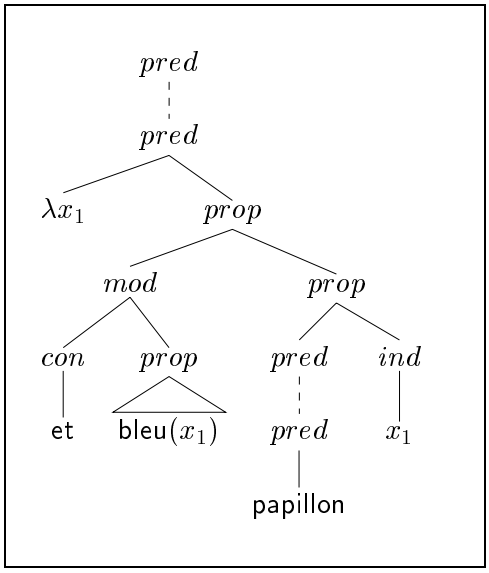


FIGURE 5.17: DESCRIPTION FOR *papillon bleu* (*'blue butterfly'*) IN G_{sem}

The larger domains of locality enable us to choose different semantic contexts for an expression depending on its elementary syntactic description in G_{syn}.

Non-intersective adjectives

Non-intersective adjectives modify predicates (e.g. nouns or other adjectives) but they are not predicates themselves and they do not have an intersective interpretation. Therefore their type is $\langle\langle e,\langle s,t\rangle\rangle,\langle e,\langle s,t\rangle\rangle\rangle$. Examples are *ancien* (*'former'*), *futur* (*'future'*), *prétendu* (*'alledged'*) and *soi-disant* (*'so-called'*). The elementary descriptions for these adjectives (see Fig. 5.18) are simpler than those for intersective adjectives, since non-intersective adjectives can be directly applied to the predicate they modify.

Similar to verbs such as *croire*, *penser* etc., these adjectives also create an oblique context. E.g. the phrase *futur président* denotes a predicate. If this predicate is true in a situation s for some individual a, then this signifies that there is some future situation s' such that a is a president in s'. Therefore the interpretations of futur and ancien might be as follows:

$I(\mathsf{futur}) \in D_{\langle\langle e,\langle s,t\rangle\rangle,\langle e,\langle s,t\rangle\rangle\rangle}$	with	$[\![\mathsf{futur}]\!](P_1) = P_2$ such that for all $a \in D_e$ and all $s \in D_s$: $P_2(a)(s) = true$ iff there is a $s' \in D_s$ in the future (wrt s) with $P_1(a)(s) = true$
$I(\mathsf{ancien}) \in D_{\langle\langle e,\langle s,t\rangle\rangle,\langle e,\langle s,t\rangle\rangle\rangle}$	with	$[\![\mathsf{ancien}]\!](P_1) = P_2$ such that for all $a \in D_e$ and all $s \in D_s$: $P_2(a)(s) = true$ iff there is a $s' \in D_s$ temporally preceding s with $P_1(a)(s) = true$

I won't discuss the notions "in the future" and "temporally preceding" used in the interpre-

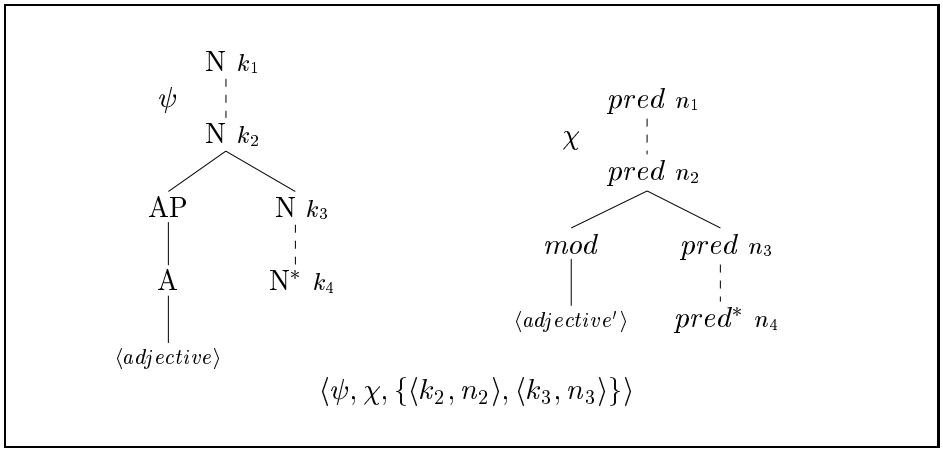


FIGURE 5.18: ELEMENTARY DESCRIPTIONS FOR NON-INTERSECTIVE ADJECTIVES PRECEDING THE MODIFIED NOUN

tations of futur and ancien. But see Prior 1968 for further reading on tense logic.

5.4.5 Relative clauses

Relative clauses may have different kinds of interpretations, depending on the context they occur in. On the one hand there are relative clauses similar to intersective adjectives as in (26):

(26) Marie lit un livre que Jean connnait
Marie reads a book that Jean knows

Roughly, the interpretation of this sentence can be described as follows: there is an object that is a book and that is known by Jean. For this object it holds that Marie reads it. The relative clause is then a restriction of the predicate livre. From the set of all books only those books are considered that are known by Jean. Therefore such relative clauses are called *restrictive* relative clauses.

On the other hand there are relative clauses as in (27):

(27) Jean, qui est arrivé hier soir, a commencé de travailler
Jean who arrived yesterday evening has started working

This sentence does not mean that among the persons named Jean one of those who arrived yesterday (in contrast to some other person called Jean who has not arrived yet) has started working. It means that Jean (the unique Jean) has arrived yesterday and has started working. In other words, the set of persons called 'Jean' is not restricted but the relative clause gives additional information about Jean. These relative clauses are called non-restrictive or *appositive* relative clauses.

In this subsection, I will only treat restrictive relative clauses, since for appositive relative clauses even the syntactic analysis is controversal. A restrictive relative clause as in (26)

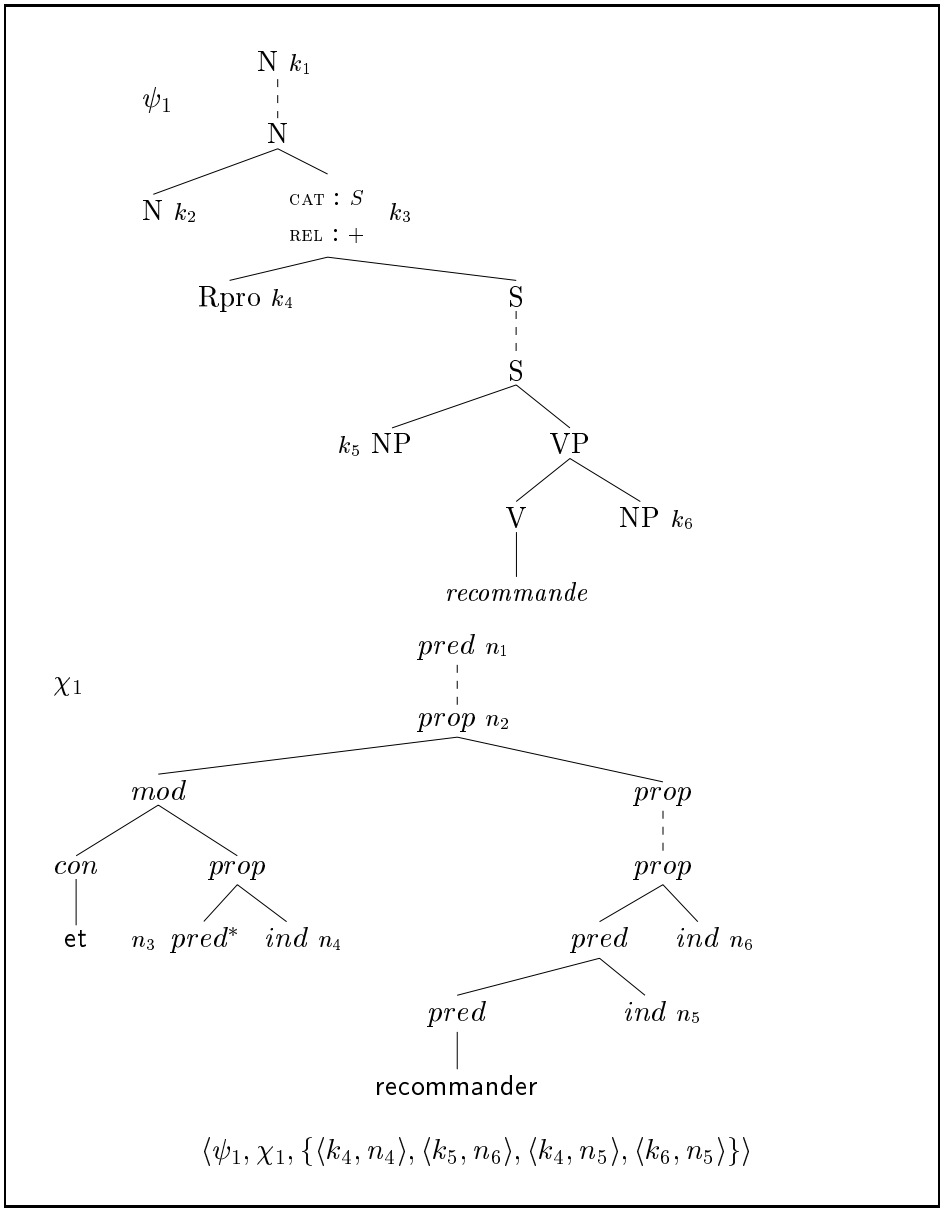


FIGURE 5.19: ELEMENTARY DESCRIPTIONS FOR *recommande* IN RESTRICTIVE RELATIVE CLAUSES

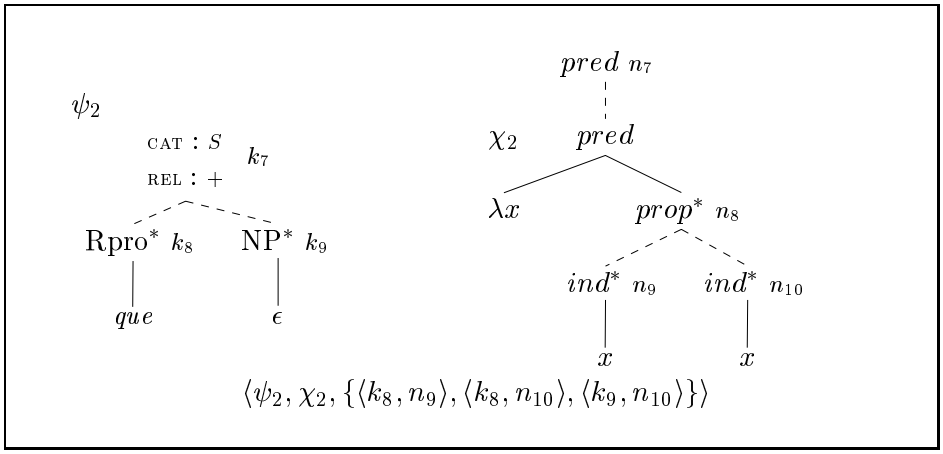


FIGURE 5.20: ELEMENTARY DESCRIPTIONS FOR A RELATIVE PRONOUN

forms a constituent with the noun that it modifies. However, for an appositive relative clause as in (27), it is not obvious whether the NP *Jean* and the relative clause form a constituent or not (see Lehmann 1993). As far as I see, once a specific syntactic analysis is adopted for appositive relative clauses, it should be no problem to extend the fragment presented here to such phenomena.

Restrictive relative clauses modify predicates in a way similar to intersective adjectives. A predicate P_1 is modified by adding a new predicate P_2. (In (26) the predicate livre is modified.) The resulting predicate is true for an individual a in a situation s iff both P_1 and P_2 are true for a in s. In order to account for this intersective interpretation, I propose elementary configurations as in Fig. 5.19 for verbs that are anchors of relative clauses. This configuration captures the case that the object-NP is relativized. Therefore, the slot for the relative pronoun in ψ_1 is related to the two slots n_4 and n_5. These node names describe the argument slots the predicates P_1 and P_2 are applied to. Then the conjunction of the resulting propositions is built.

The relative pronoun is a λ-abstractor, i.e. it introduces λx for a new variable x. This variable must occur twice in the scope of λx, once as argument of P_1 and once as argument of P_2. In the case of Fig. 5.19 it must be inserted under n_4 and under n_5. Therefore I propose elementary configurations for relative pronouns as shown in Fig. 5.20 for *que*. If such a configuration is added to $\langle \psi_1, \chi_1, \{\langle k_4, n_4\rangle, \langle k_5, n_6\rangle, \langle k_4, n_5\rangle, \langle k_6, n_5\rangle\}\rangle$ in Fig. 5.19, the synchronization relation makes sure that the NP with the empty word is inserted under k_6 and that the two variables x are inserted under the node names that are related to the relative pronoun in ψ_1.

It is easy to see that using these configurations (26) is analyzed in the desired way. Starting from the configuration derived for the matrix clause *Marie lit un livre*, first the configuration $\langle \psi_1, \chi_1, \{\langle k_4, n_4\rangle, \langle k_5, n_6\rangle, \langle k_4, n_5\rangle, \langle k_6, n_5\rangle\}\rangle$ for *recommande* (see Fig. 5.19) is added such that the derivation descriptions in this step are the two descriptions in the elementary configuration of the predicate *livre*. In the following two derivation steps, the elementary configuration $\langle \psi_2, \chi_2, \{\langle k_8, n_9\rangle, \langle k_8, n_{10}\rangle, \langle k_9, n_{10}\rangle\}\rangle$ for the relative pronoun *que* and the elementary configuration for *Jean* are added (in arbitrary order). When adding the configuration for *que*, the

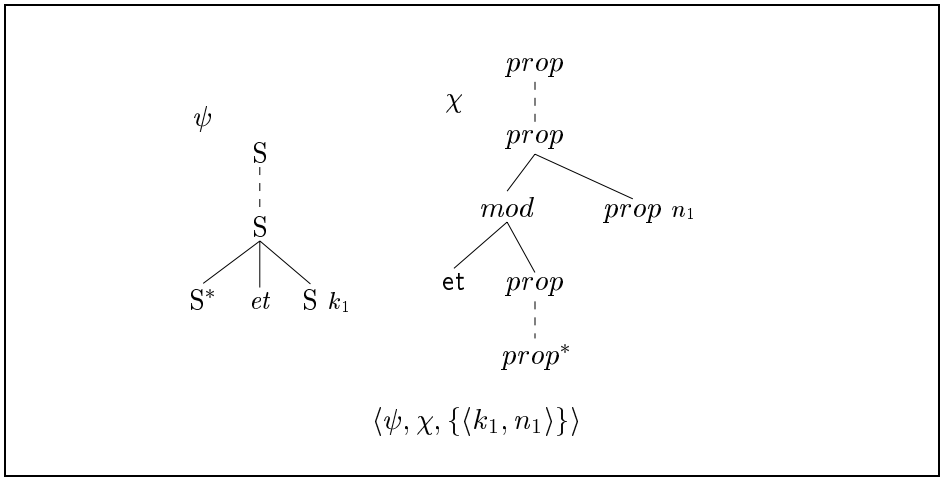


FIGURE 5.21: ELEMENTARY DESCRIPTIONS FOR THE CONJUNCTION *et*

synchronization relation guarantees that the new equivalences are $k_3 \approx k_7 \wedge k_4 \approx k_8 \wedge k_6 \approx k_9$ in G_{syn} and $n_1 \approx n_7 \wedge n_2 \approx n_8 \wedge n_4 \approx n_9 \wedge n_5 \approx n_{10}$ in G_{sem}. Instead of the predicate livre the modified predicate $\lambda x(\mathsf{et}(\mathsf{livre}(x))(\mathsf{recommander}(x)(\mathsf{Jean})))$ is then obtained.

5.4.6 Coordination

In the fragment presented in this chapter, I will restrict myself to coordination of sentences as in (28). Deletion phenomena as in (29) are left aside.

(28) Jean aime Marie et Pierre aime Lucile
Jean loves Marie and Pierre loves Lucile

(29) a. Jean achète et Pierre prépare les poissons
Jean buys and Pierre prepares the fishes

b. Jean aime la cuisine provençale et adore la bouillabaisse
Jean loves Provençal cooking and adores the Bouillabaisse

For coordination of sentences as in (28), I adopt the elementary configuration shown in Fig. 5.21 for the conjunction *et*. This configuration is added to the first conjunct (which is then dominated by the marked names), and then the two elementary descriptions for the anchor of the second conjunct are inserted under k_1 and n_1. An example will be shown later when considering island constraints.

5.5 Underspecification

A lot of attention has been paid to underspecification recently, mainly to underspecified semantic representations. There are two reasons for replacing several analyses of a single expression by one underspecified analysis. The more practical reason is the desire to avoid

combinatorial explosion. (1) in Chapter 1 for example, repeated as (30), is such a case of combinatorial explosion where we have 14400 readings (see also Poesio 1996).

(30) A politician can fool most voters on most issues most of the time, but no politician can fool all voters on every single issue all of the time.

Such examples clearly show that it is desirable to allow scope ambiguities to exist for some time instead of resolving them immediately. For this purpose a level of underspecified representations is necessary.

The second reason for exploring underspecification concerns considerations of psychological plausibility. From a cognitive point of view it does not seem plausible that we process sentences by first generating all readings and then testing them. (30) clearly shows that this is probably not the case. It appears more cognitively adequate to generate underspecified representations and to postpone the resolution of ambiguities.

Most recent theories of underspecified semantics propose to obtain underspecified representations by a relaxation of a relation that represents scope. Approaches of this kind are Underspecified Discourse Representation Structures (UDRT), proposed by Reyle (1993), the Ambiguous Logical Forms introduced by Muskens (1995), the context constraints in Niehren et al. 1997b or the Hole Semantics presented by Bos (1995). Richter and Sailer (1996, 1997) present a definition of Bos' semantic representation language in King's formalization of HPSG (King 1994).

A common property of these proposals is that scope relations are presented by a partial order. For UDRTs this is the *subordination* relation between DRSs, in Muskens' Ambiguous Logical Form scope is represented by a dominance relation between nodes in a tree, in the context constraints by Niehren, Pinkal and Ruhrberg it is the subtree relation and in Bos' Hole Semantics the partial order is also called *subordination*. Since the underspecified representation in these approaches are partial descriptions of semantic representations, the logical rule is respected.

The theories that are closest to the syntax-semantics interface presented in this chapter are Muskens' Ambiguous Logical Form and the context constraints in Niehren et al. 1997b because they both represent scope ambiguities by partial tree descriptions. In contrast to local TDGs, both theories assume a close relation between syntax and semantics in the sense that one phrase structure rule (or one subtree consisting of a mother node and its immediate daughters) is connected to a specific rule generating parts of a semantic expression. Since phrase structure rules do not offer an extended domain of locality, island constraints must be explicitely stated in order to be respected. In the last section of this chapter, we will have a more detailed look at Ambiguous Logical Form and context constraints, and both proposals will be compared to synchronous local TDGs.

In this section underspecified representations for scope ambiguities are presented that can be derived within the syntax-semantics interface introduced in the previous section. We will see that in the case of synchronous local TDGs, island constraints are a consequence of the grammar (i.e. the elementary configurations) and the locality of the derivation. Since the form of the elementary configurations is motivated by the predicate argument cooccurrence principle, island constraints follow from this principle and from the locality of the derivation. The definition of the locality restriction was motivated by the desire to show semilinearity of

the string languages generated by the grammar formalism. In other words, there are more general reasons for local derivations, and island constraints arise as a consequence of this.

5.5.1 Scope ambiguities

In the following, we will see that the fragment of the syntax-semantics interface presented in the previous sections allows for the derivation of suitable underspecified representations for scope ambiguities. Examples of quantifier scope ambiguities, de re – de dicto ambiguities and structural ambiguities with nominal modifiers are presented.

Quantifier scope

(31) un homme aime chaque femme
a man loves every woman

A well known example of a quantifier scope ambiguity is given in (31). We have already seen Montague's account of the two scope orders of (31) in 5.2.1. (31) has the following two readings with respect to the relative scope of the two quantifiers *un homme* and *chaque femme*:

1. either *un homme* has wide scope, and then in some situation s (31) signifies that there is one single man who loves all the women in this situation s.

2. or *chaque femme* has wide scope, and then in some situation s (31) signifies that for each woman in that situation s there is at least one man who loves this woman in s. In this case, the women need not necessarily all be loved by the same man.

Although the first reading implies the second interpretation, and is therefore stronger, there is good reason to believe that these are distinct readings of (31): Chierchia and McConnell-Ginet 1990, p. 118 points out that under negation the second reading becomes the stronger reading, i.e. the negation of the second reading implies the negation of the first reading.

For the quantifier scope ambiguity in (31), the derived configuration is the one shown in Fig. 5.22. The description φ that is derived in G_{sem} has two minimal trees: since all parent relations in minimal trees must occur in the description, either n_1 and n_2 denote the same node or n_1 and n_6. In the first case, n_5 and n_6 and also n_9 and n_{10} denote the same nodes as well and a minimal tree for the stronger reading is obtained. In the second case, n_9 and n_2 also denote the same node and n_5 and n_{10} as well. The two minimal trees of φ are shown in Fig. 5.23. The interpretations of the terms yielded as strings by these two trees are:

1. $[\![\mathsf{un}(\mathsf{homme})(\lambda x_1(\mathsf{chaque}(\mathsf{femme})(\lambda x_2(\mathsf{aimer}(x_2)(x_1)))]\!]^g_M = p$ with
   $[\![\mathsf{un}(\mathsf{homme})(\lambda x_1(\mathsf{chaque}(\mathsf{femme})(\lambda x_2(\mathsf{aimer}(x_2)(x_1)))]\!]^g_M(s) = true$
   iff there is a $a \in D_e$ with $[\![\mathsf{homme}]\!]^g_M(a)(s) = true$ and
   $[\![\lambda x_1(\mathsf{chaque}(\mathsf{femme})(\lambda x_2(\mathsf{aimer}(x_2)(x_1))))]\!]^g_M(a)(s) = true$
   iff there is a $a \in D_e$ with $[\![\mathsf{homme}]\!]^g_M(a)(s) = true$ and

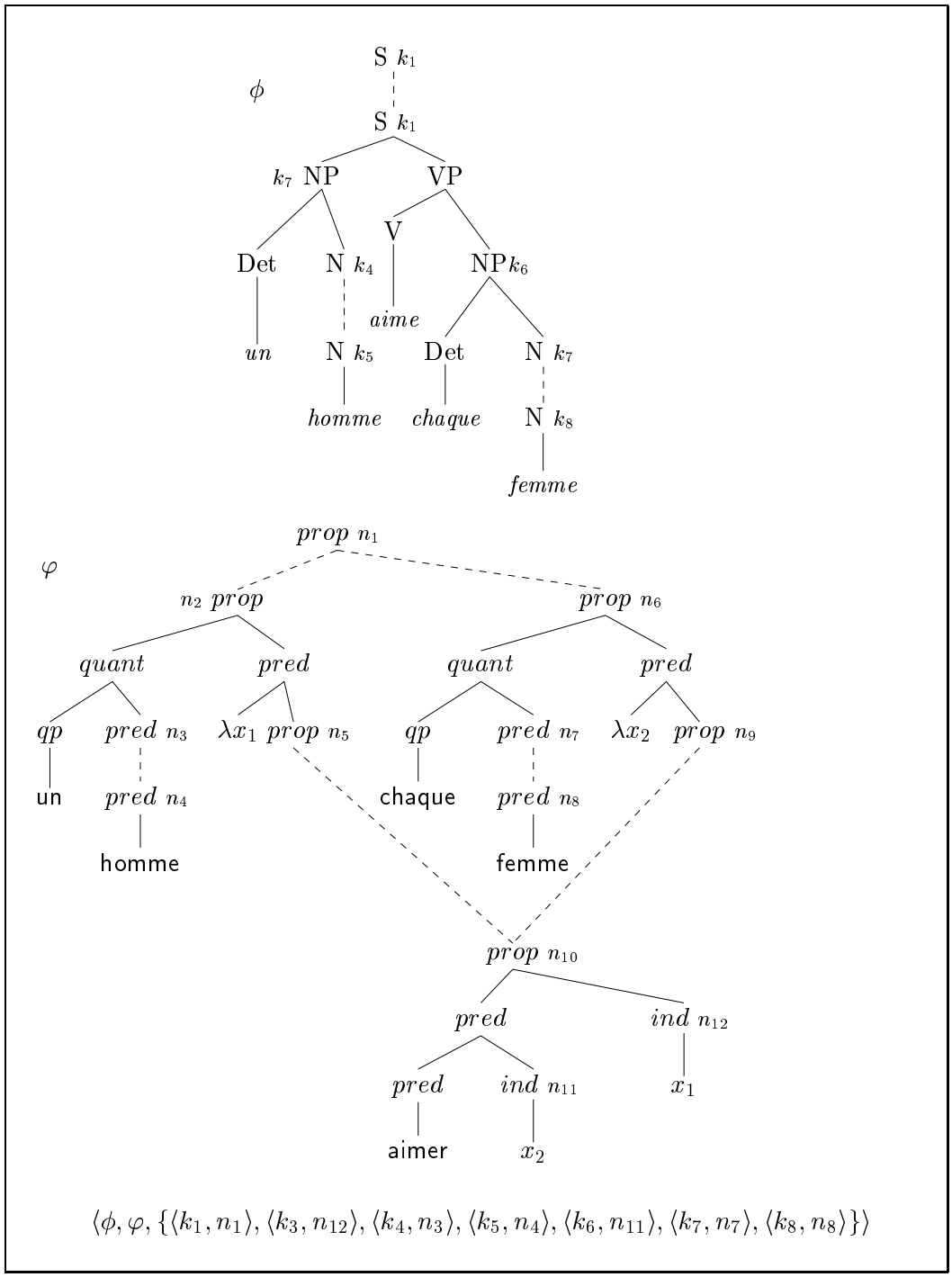


FIGURE 5.22: CONFIGURATION DERIVED FOR (31) *un homme aime chaque femme*

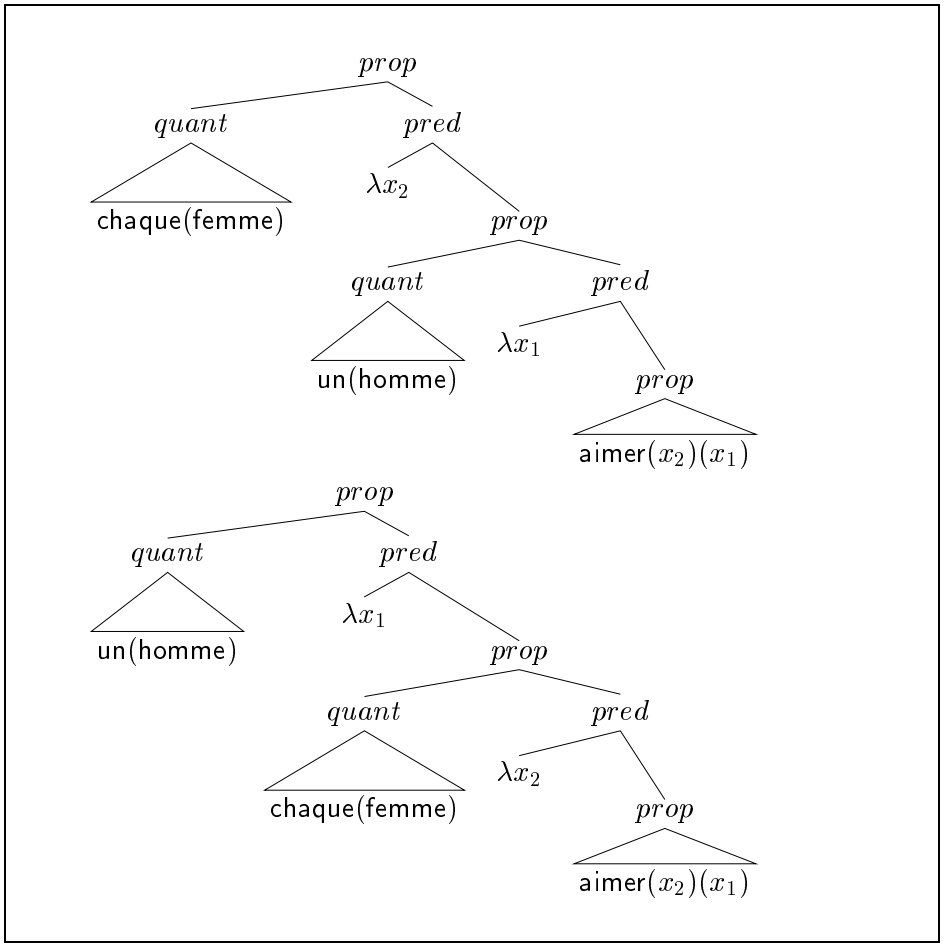


FIGURE 5.23: MINIMAL TREES OF φ IN FIG. 5.22

$[\![\mathsf{chaque(femme)}(\lambda x_2(\mathsf{aimer}(x_2)(x_1)))]\!]_M^{g[x_1=a]}(s) = true$

iff there is a $a \in D_e$ with $[\![\mathsf{homme}]\!]_M^g(a)(s) = true$, such that for all $b \in D_e$:

if $[\![\mathsf{femme}]\!]_M^g(b)(s) = true$, then $[\![\lambda x_2(\mathsf{aimer}(x_2)(x_1)))]\!]_M^{g[x_1=a]}(b)(s) = true$

iff there is a $a \in D_e$ with $[\![\mathsf{homme}]\!]_M^g(a)(s) = true$, such that for all $b \in D_e$:

if $[\![\mathsf{femme}]\!]_M^g(b)(s) = true$, then $[\![\mathsf{aimer}(x_2)(x_1)]\!]_M^{g[x_1=a,x_2=b]}(s) = true$

iff there is a $a \in D_e$ with $[\![\mathsf{homme}(a)]\!]_M^g(s) = true$, such that for all $b \in D_e$:

if $[\![\mathsf{femme}]\!]_M^g(b)(s) = true$, then $[\![\mathsf{aimer}]\!]_M^g(b)(a)(s) = true$

2. $[\![(\mathsf{chaque(femme)}(\lambda x_2(\mathsf{un(homme)}(\lambda x_1(\mathsf{aimer}(x_2)(x_1)))]\!]_M^g = p$ with
$[\![(\mathsf{chaque(femme)}(\lambda x_2(\mathsf{un(homme)}(\lambda x_1(\mathsf{aimer}(x_2)(x_1)))]\!]_M^g(s) = true$ iff

...

iff for all $b \in D_e$: if $[\![\mathsf{femme}]\!]_M^g(b)(s) = true$, then there is a $a \in D_e$ with

$[\![\mathsf{homme}]\!]^g_M(a)(s) = true$ and $[\![\mathsf{aimer}]\!]^g_M(b)(a)(s) = true$

As this example shows, synchronous local TDGs allow the derivation of underspecified representations for quantifier scope ambiguities in the style proposed in Fig. 2.22, page 51 in Chapter 2. The underspecification in this case arises from not fully specifying dominance and linear precedence for the node names in the description in G_{sem}. The description derived in G_{syn} is not ambiguous, i.e. in contrast to Montague's PTQ, in this system quantifier scope ambiguities are analyzed as semantic ambiguities and unnecessary syntactic ambiguities do not arise. Since syntax and semantics are less close than in Montague's PTQ or in Cooper 1983, a (syntactic or semantic) quantifying-in rule is not necessary in order to account for wide scope of quantifiers.

De dicto – de re ambiguities

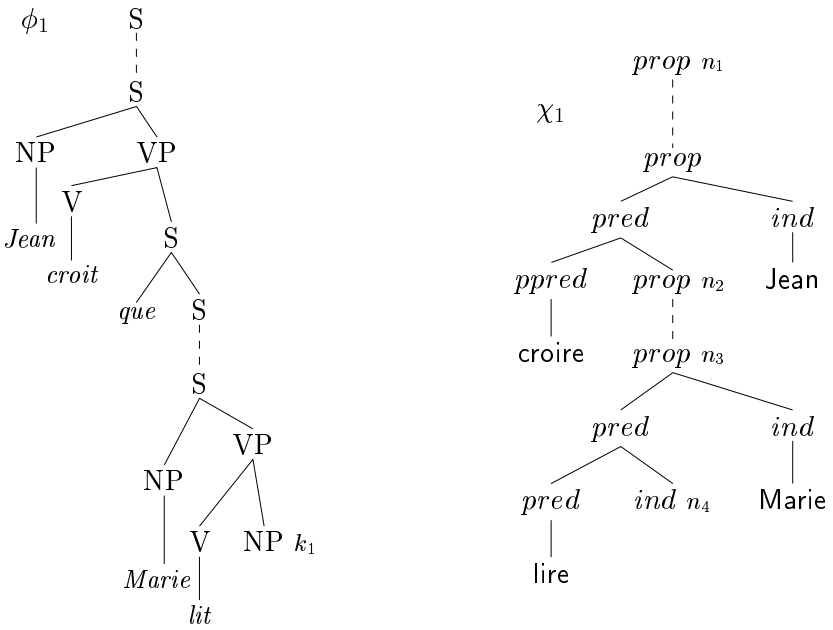
(32) Jean croit que Marie lit un livre
Jean believes that Marie reads a book

De dicto – de re ambiguities are scope ambiguities that arise when a quantifier is embedded in an oblique construction. The quantifier *un livre* in (32) for example can have a so-called *specific* or *de re* interpretation or an *unspecific* or *de dicto* interpretation. These two readings can be analyzed as involving a scope ambiguity (see Quine 1960 p. 139). In Montague's PTQ, the wide scope reading of the embedded quantifier can be obtained by quantifying-in.

The de re interpretation of (32) is the wide scope reading, i.e. the reading where there is one single specific book, and Jean believes about this specific book that Marie is reading it. The narrow scope or de dicto reading of *un livre* is the one where Jean only believes that Marie is reading some book but he does not have a specific book in mind. In other words, if this sentence is interpreted in a situation s, then in the de re reading, there must be a book in s whereas in the de dicto reading, a book must exist in all situations where everything is the case that Jean believes in s.

In the course of the derivation of a configuration for (32), in G_{sem} first the configuration $\langle \phi_1, \varphi_1, \sigma \rangle$ shown in Fig. 5.24 is derived for *Jean croit que Marie lit*, and then the elementary configurations for *un* and *livre* must be added. When the elementary configuration $\langle \psi_1, \chi_1, \{\langle k_2, n_8 \rangle, \langle k_3, n_6 \rangle\} \rangle$ of *un* is added, the derivation descriptions must be those added for the verb *lit*. In Fig. 5.24 only those node names in ϕ_1 and φ_1 are explicitely mentioned that are relevant for this derivation step. These are k_1 in ϕ_1, since k_1 must become equivalent to the minimal name of ψ_1, and in φ_1, the names n_1, n_2, n_3, n_4 are important. n_1, n_2 and n_3 are all node names from the elementary configuration of *lit* that have the category *prop* and are therefore candidates for an equivalence with the minimal n_5 and the marked n_6 in χ_1. Since *que Marie lit un livre* is a complement clause of *croit*, the minimal name n_1 in the elementary description of *lit* in G_{sem} was identified with a minimal name in the elementary description of *croit* (see Fig. 5.13 and 5.14 for the elementary configurations of *croit* and of anchors of complement clauses). Therefore n_1 is among those that can be chosen for new equivalences in this derivation step, and it is possible to move the quantifier out of the complement clause. n_4 must become equivalent to n_8. As the result of a derivation step in a local TDG must be maximally underspecified, the equivalences $n_1 \approx n_5$ and $n_3 \approx n_7$ must be chosen. Therefore, the description κ' shown in Fig. 5.25 is derived.

Configuration $\kappa := \langle \phi_1, \varphi_1, \sigma \rangle$ for *Jean croit que Marie lit* with $\langle k_1, n_4 \rangle \in \sigma$, and n_1, n_2 and n_3 are all node names with category *prop* from the elementary description in G_{sem} for lire.



Elementary configuration κ_1 for *un*: $\kappa_1 := \langle \psi_1, \chi_1, \{\langle k_2, n_8 \rangle, \langle k_3, n_6 \rangle\} \rangle$

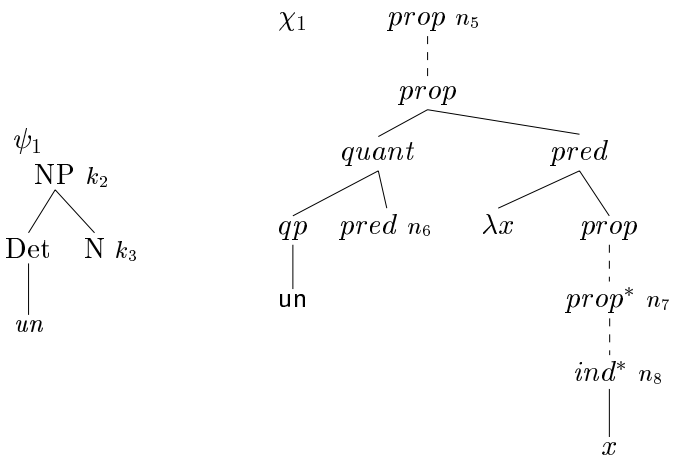


FIGURE 5.24: CONFIGURATION DERIVED FOR *Jean croit que Marie lit* AND ELEMENTARY CONFIGURATION FOR *un*

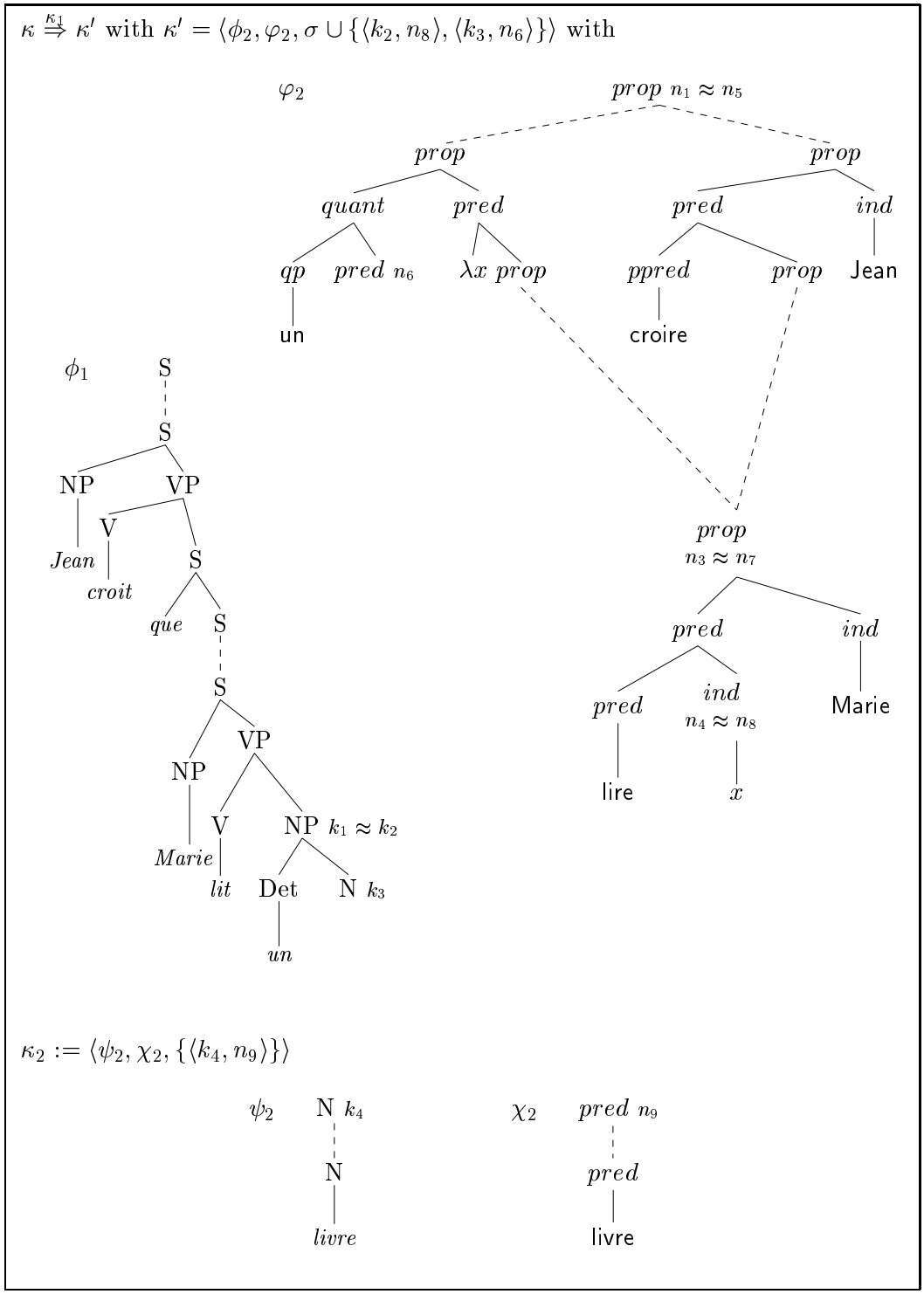


FIGURE 5.25: CONFIGURATION DERIVED FOR *Jean croit que Marie lit un* AND ELEMENTARY CONFIGURATION FOR *livre*

In the last derivation step, the elementary configuration $\langle \psi_2, \chi_2, \{\langle k_4, n_9\rangle\}\rangle$ for *livre* in Fig. 5.25 must be added, such that the derivation descriptions are the descriptions added for *un*. Consequently, the new equivalences must be $k_3 \approx k_4$ in G_{syn} and $n_6 \approx n_9$ in G_{sem}. The description $\varphi_2 \wedge \chi_2 \wedge n_6 \approx n_9$ derived in G_{sem} for (32) has two minimal trees yielding the following terms:

1. $\mathsf{un}(\mathsf{livre})(\lambda x(\mathsf{croire}(\mathsf{lire}(x)(\mathsf{Marie}))(\mathsf{Jean})))$
2. $\mathsf{croire}(\mathsf{un}(\mathsf{livre})(\lambda x(\mathsf{lire}(x)(\mathsf{Marie}))))(\mathsf{Jean})$

The interpretation of the first term is the de re or specific reading whereas the interpretation of the second term is the de dicto or unspecific reading.

Nominal modifiers

(33) a. grande maison bleue
large house blue
large blue house / blue large house

b. ancien château royal
former castle royal
former royal castle / royal former castle

Expressions with several nominal modifiers as in (33) are also ambiguous. However, in contrast to the quantifier scope ambiguities considered above, the examples in (33) are even ambiguous with respect to their syntactic structure and, corresponding to the two syntactic readings, there arises also a semantic ambiguity. This is not the case in English, since modifying adjectives in English must always precede the noun they are modifying. In French, some adjectives (e.g. color adjectives) must and some can follow the noun.

The elementary configurations for *maison*, *grande* and *bleue* are shown in Fig. 5.26. When deriving a configuration for (33)a., the configuration for *maison* is taken and κ_2 and κ_3 are added (no matter in which order), and the resulting description is such that the equivalences $k_1 \approx k_3 \wedge k_3 \approx k_7 \wedge k_2 \approx k_6 \wedge k_6 \approx k_{10}$ in G_{syn} and $n_1 \approx n_3 \wedge n_3 \approx n_7 \wedge n_2 \approx n_6 \wedge n_6 \approx n_{10}$ in G_{sem} hold. The resulting configuration is shown in Fig. 5.27.

The two descriptions in the configuration derived for (33)a. both have two minimal trees. The strings yielded by the description ϕ_{sem} derived in G_{sem} are the following two terms of type $\langle e, \langle s, t\rangle\rangle$:

1. $\lambda x(\mathsf{et}(\mathsf{grand}(x))((\lambda y(\mathsf{et}(\mathsf{bleu}(y))(\mathsf{maison}(y))))(x)))$
(equivalent to $\lambda x(\mathsf{et}(\mathsf{grand}(x))(\mathsf{et}(\mathsf{bleu}(x))(\mathsf{maison}(x)))))$
2. $\lambda y(\mathsf{et}(\mathsf{bleu}(y))((\lambda x(\mathsf{et}(\mathsf{grand}(x))(\mathsf{maison}(x))))(y)))$
(equivalent to $\lambda y(\mathsf{et}(\mathsf{bleu}(y))(\mathsf{et}(\mathsf{grand}(y))(\mathsf{maison}(y)))))$

Since the interpretation of et is an associative and commutative function (i.e. for all models M and assignments g: $[\![\mathsf{et}(p_1)(p_2)]\!]^g_M = [\![\mathsf{et}(p_2)(p_1)]\!]^g_M$ and $[\![\mathsf{et}(p_1)(\mathsf{et}(p_2)(p_3))]\!]^g_M = [\![\mathsf{et}(\mathsf{et}(p_1)(p_2))(p_3)]\!]^g_M$), the interpretations of these two terms are the same.

Elementary configurations for *maison*, *grande* and *bleue*:

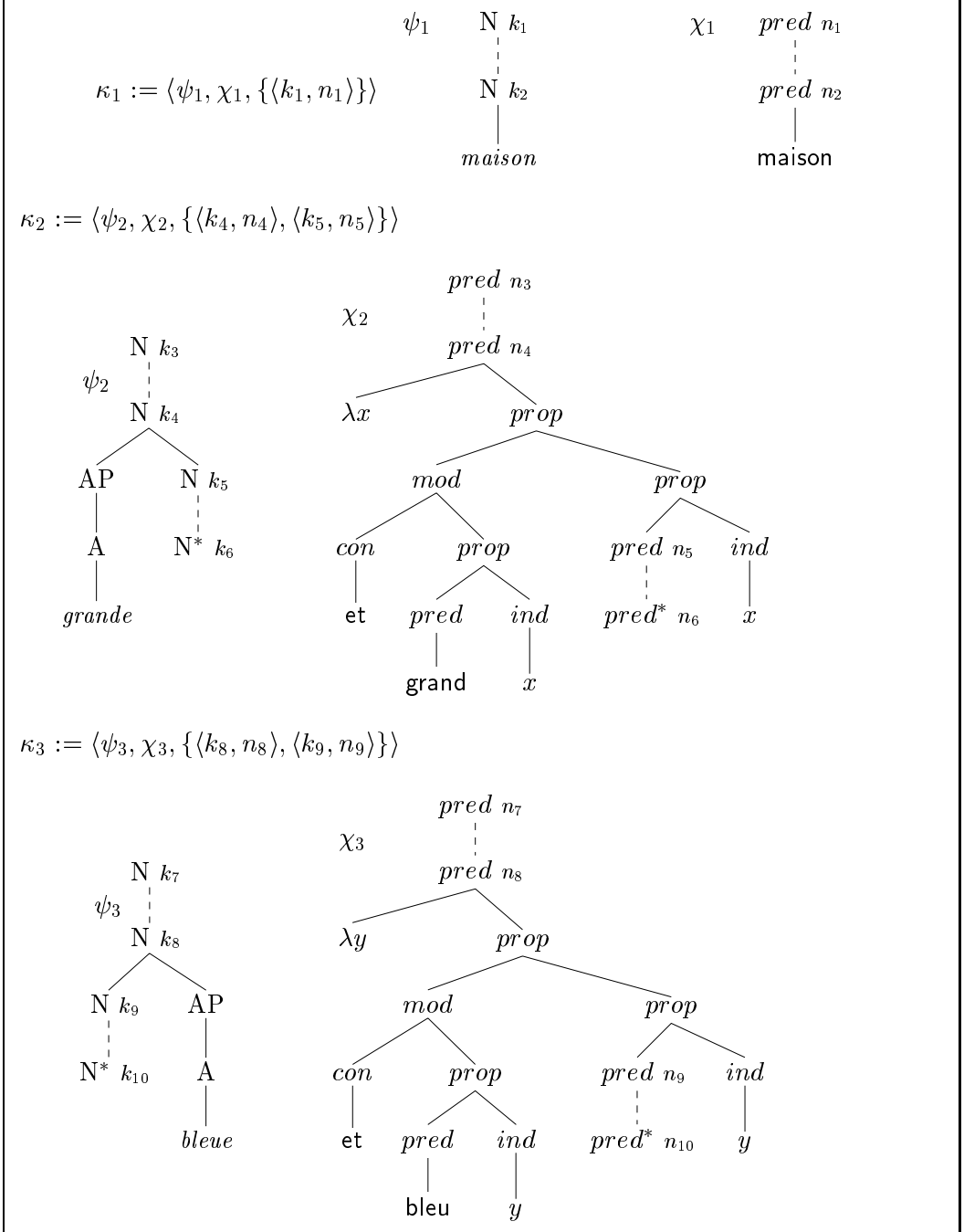


FIGURE 5.26: ELEMENTARY CONFIGURATIONS FOR *grande*, *maison* AND *bleue*

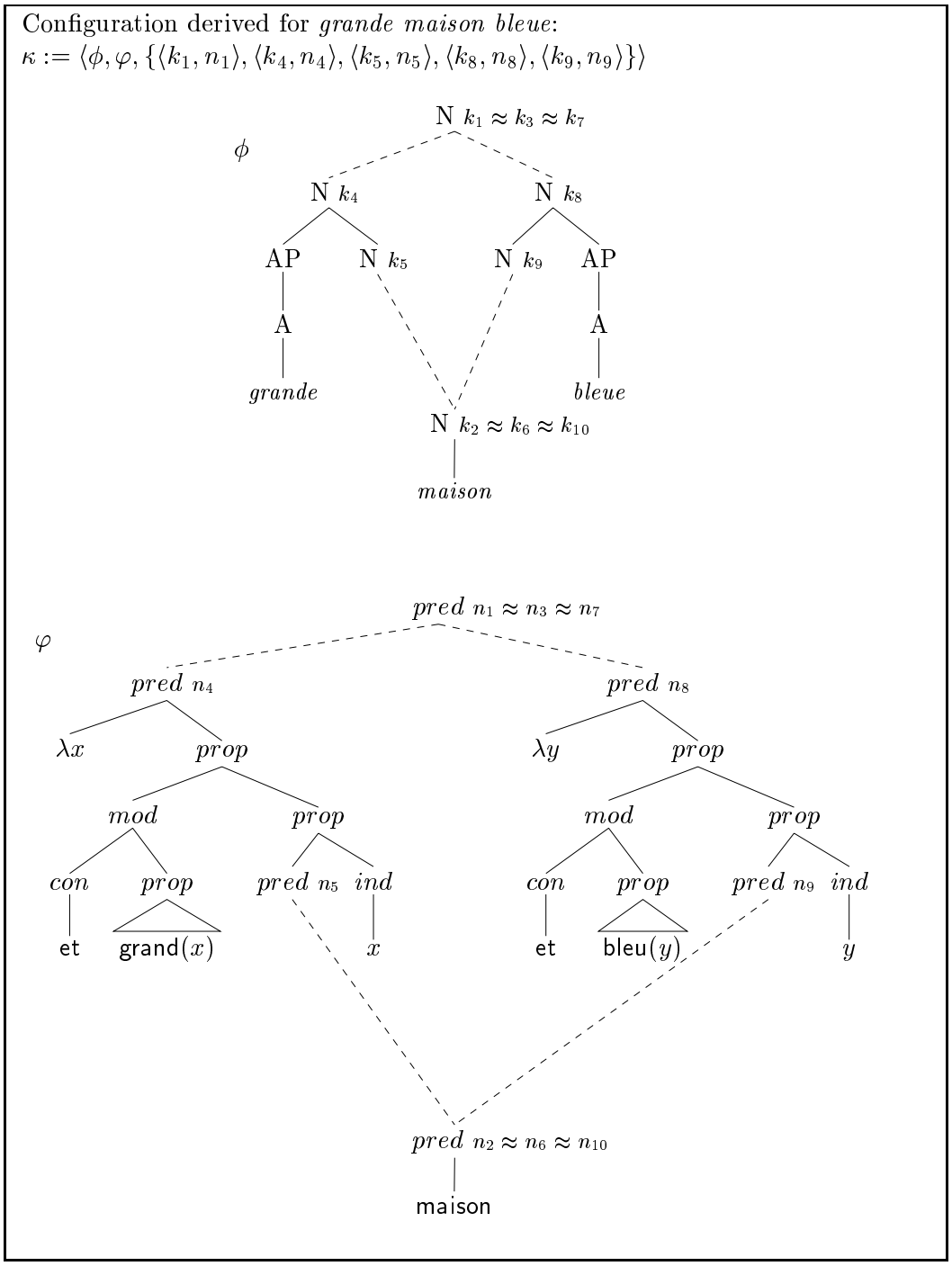


FIGURE 5.27: DESCRIPTION DERIVED IN G_{sem} FOR *grande maison bleue*

(33)b. is also syntactically ambiguous. The following similar example in English was already shown in Chapter 1:

(34) former professor in Tübingen

In (34) one of the modifiers is a PP. Since PPs follow the noun they modify, such examples in English are syntactically ambiguous.

The configuration derived for (33)b., shown in Fig. 5.28, is very similar to the one derived for (33)a. The two descriptions in the configuration for (33)b. each have two minimal trees. These minimal trees are shown in Fig. 5.29. The two strings yielded by the two minimal trees of the description in G_{sem} and their interpretations are:

1. $\mathsf{ancien}(\lambda y(\mathsf{et}(\mathsf{royal}(y))(\mathsf{chateau}(y))))$:
   $[\![\mathsf{ancien}(\lambda y(\mathsf{et}(\mathsf{royal}(y))(\mathsf{chateau}(y))))]\!](a)(s) = true$
   iff there is a $s' \in D_s$ temporally preceding s such that
   $[\![\lambda y(\mathsf{et}(\mathsf{royal}(y))(\mathsf{chateau}(y)))]\!](a)(s') = true$
   iff there is a $s' \in D_s$ temporally preceding s such that $[\![\mathsf{royal}]\!](a)(s') = true$ and
   $[\![\mathsf{chateau}]\!](a)(s') = true$.
2. $\lambda y(\mathsf{et}(\mathsf{royal}(y))((\mathsf{ancien}(\mathsf{chateau}))(y)))$:
   $[\![\lambda y(\mathsf{et}(\mathsf{royal}(y))((\mathsf{ancien}(\mathsf{chateau}))(y)))]\!](a)(s) = true$
   iff $[\![\mathsf{royal}]\!](a)(s) = true$ and there is a $s' \in D_s$ temporally preceding s such that
   $[\![\mathsf{chateau}]\!](a)(s') = true$.

According to this, the interpretation of $\mathsf{ancien}(\lambda y(\mathsf{et}(\mathsf{royal}(y))(\mathsf{chateau}(y))))$ is a predicate that is true for some object a in a situation s iff in some former days a was a royal castle (wide scope reading of ancien). The interpretation of $\lambda y(\mathsf{et}(\mathsf{royal}(y))((\mathsf{ancien}(\mathsf{chateau}))(y)))$ (narrow scope of ancien) is a predicate that is true for some object a in a situation s iff a is royal (in s), and in some former days, a was been a castle (in s there is perhaps only a ruin left).

Looking at the minimal trees for the descriptions in the configuration for (33)b., one can see that the first syntactic structure corresponds to the first minimal tree of φ (the wide scope reading of *ancien*), whereas the second syntactic structure corresponds to the second minimal tree for φ. This correspondence is expressed by the synchronization relation between the node names of ϕ and φ in Fig. 5.28.

5.5.2 Island constraints

In the following, I will consider the restrictions for quantifier scope mentioned in the beginning of this chapter, and I will show that these restrictions are respected within the syntax-semantics interface presented above.

In 5.2.3 we have seen that the following constraints for quantifier scope should hold:

Coordination constraint: A quantifier cannot be raised out of a coordination.

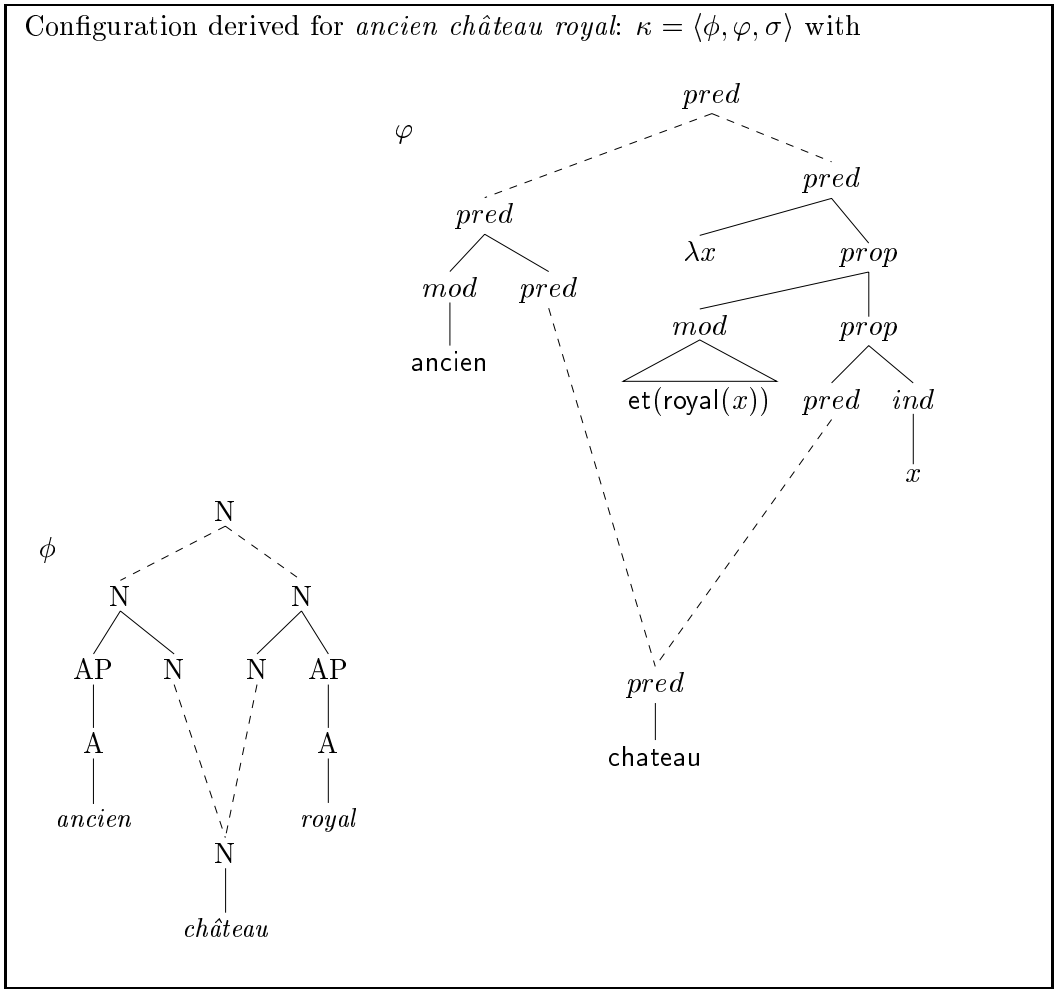


FIGURE 5.28: CONFIGURATION DERIVED IN G_{sem} FOR *ancien château royal*

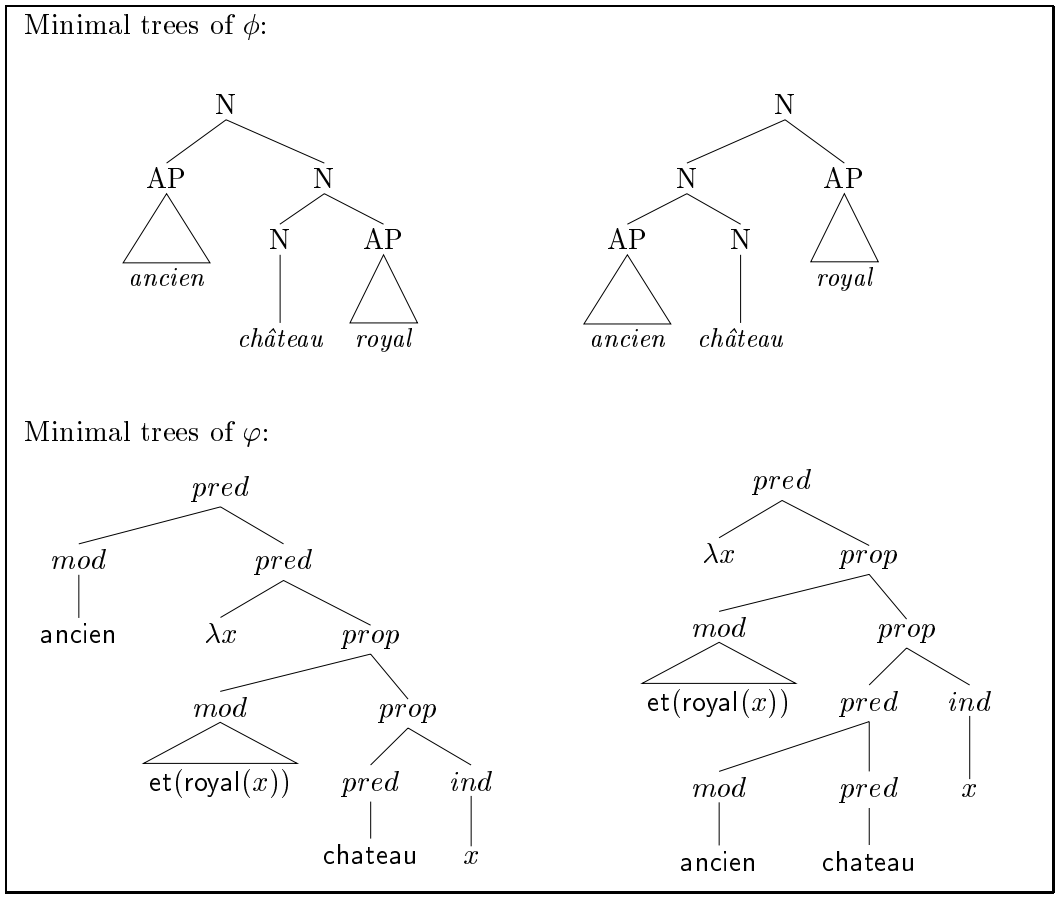


FIGURE 5.29: MINIMAL TREES OF THE DESCRIPTIONS DERIVED FOR *ancien château royal* SHOWN IN FIG. 5.28

Relative clause constraint: A quantifier cannot be raised out of a relative clause.

Logical rule: A quantifier A that is not in the restriction of a quantifier B and that occurs within the scope of B cannot outscope any of the quantifiers in the restriction of B.

In the course of this subsection we will see that within the grammar presented in Section 5.4, these constraints are satisfied without being explicitely stated. The island constraints, i.e. the first two constraints, follow from the locality of the derivation and from the specific form of the elementary configurations. The logical rule holds as a consequence of the fact that G_{sem} describes the syntactic structure of logical terms.

As an example for the coordination constraint consider (35):

(35) une princesse dort et chaque prince arrive
a princess sleeps and each prince arrives

In (35) wide scope of the quantifier *chaque prince* is not possible. The derivation of a configuration for (35) is shown in Fig. 5.30, 5.31 and 5.32. First the configuration κ for the first conjunct *une princesse dort* is derived. In the next step, the elementary configuration κ_1 (see Fig. 5.30) for *et* must be added with the new equivalences $k_1 \approx k_3$ and $k_2 \approx k_4$ in G_{syn} and $n_1 \approx n_3$ and $n_2 \approx n_4$ in G_{sem}. The anchor of the second conjunct is the verb *arrive*. Therefore in the following derivation step κ_2, the elementary configuration for *arrive* must be added. Since there are no marked names in κ_2, the two minimal names k_6 and n_6 both must become equivalent to leaf names in the old configuration. Consequently, the new equivalences $k_5 \approx k_6$ and $n_5 \approx n_6$ must be added, and the configuration $\langle \phi_3, \varphi_3, \sigma \cup \{\langle k_5, n_5\rangle, \langle k_8, n_8\rangle\}\rangle$ shown in Fig. 5.31 is obtained. In the last two derivation steps, first the elementary configuration for *chaque* must be added and then the elementary configuration for *prince*. The configuration for *chaque* must be added in such a way that the elementary descriptions for *arrive* are the derivation descriptions, since *chaque* introduces an argument of *arrive*. Therefore the minimal name (with category *prop*) of the elementary description for chaque in G_{sem} must become equivalent to n_6, i.e. the quantifier is "inserted" between n_6 and n_7. Finally, a configuration $\langle \phi_4, \varphi_4, \sigma' \rangle$ is derived for (35) where φ_4 is as shown in Fig. 5.32. Clearly, φ_4 has two minimal trees. The terms yielded by these minimal trees are:

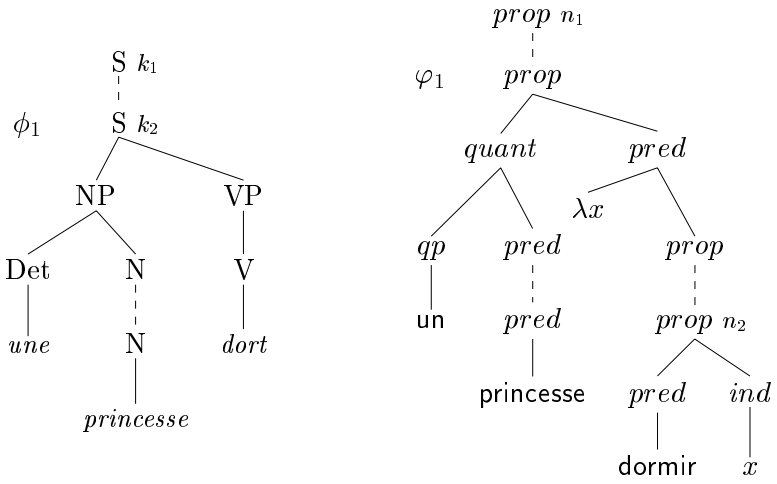
1. $\mathsf{un}(\mathsf{princesse})(\lambda x(\mathsf{et}(\mathsf{dormir}(x))(\mathsf{chaque}(\mathsf{prince})(\lambda y(\mathsf{arriver}(y))))))$
2. $\mathsf{et}(\mathsf{un}(\mathsf{princesse})(\lambda x(\mathsf{dormir}(x))))(\mathsf{chaque}(\mathsf{prince})(\lambda y(\mathsf{arriver}(y))))$

With the analysis obtained here, a quantifier occurring in the first conjunct can have wide scope whereas for a quantifier in the second conjunct, wide scope (i.e. scope over the first conjunct) is not possible. This is exactly what is predicted by the coordination constraint. The reason why the coordination constraint is respected in this framework is the fact that the conjuncts are matrix clauses and therefore the second conjunct must be inserted at two leaf names in G_{syn} and G_{sem}. This creates an island with respect to the locality restriction on the derivation.

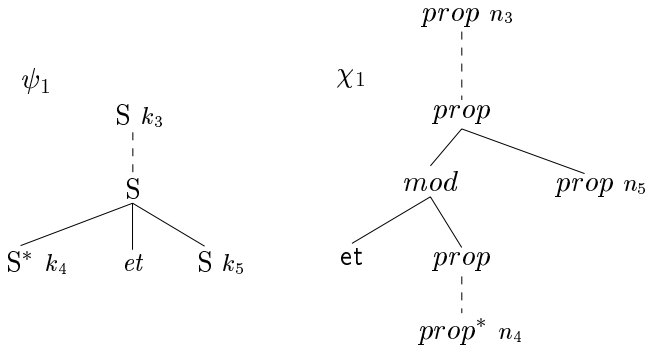
Next, the relative clause constraint is considered. An example is (36).

(36) Jean aime une princesse qui adore chaque grenouille
Jean loves a princess who adores each frog

Configuration derived for *une princesse dort*: $\kappa := \langle \phi_1, \varphi_1, \sigma \rangle$ with



Elementary configuration $\kappa_1 = \langle \psi_1, \chi_1, \{\langle k_5, n_5 \rangle\} \rangle$ for *et*:



Elementary configuration $\kappa_2 = \langle \psi_2, \chi_2, \{\langle k_6, n_6 \rangle, \langle k_8, n_8 \rangle\} \rangle$ for *arrive*:

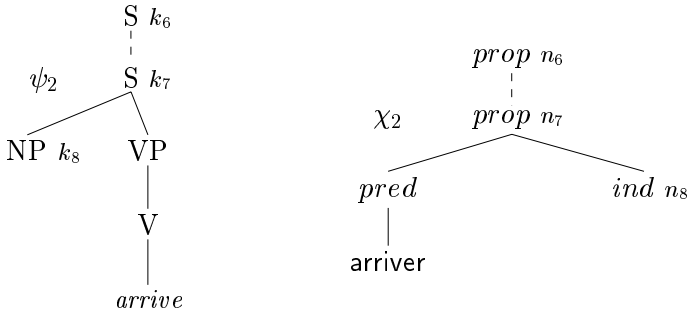


FIGURE 5.30: CONFIGURATION DERIVED FOR *une princesse dort* AND ELEMENTARY CONFIGURATIONS FOR *et* AND *arrive*

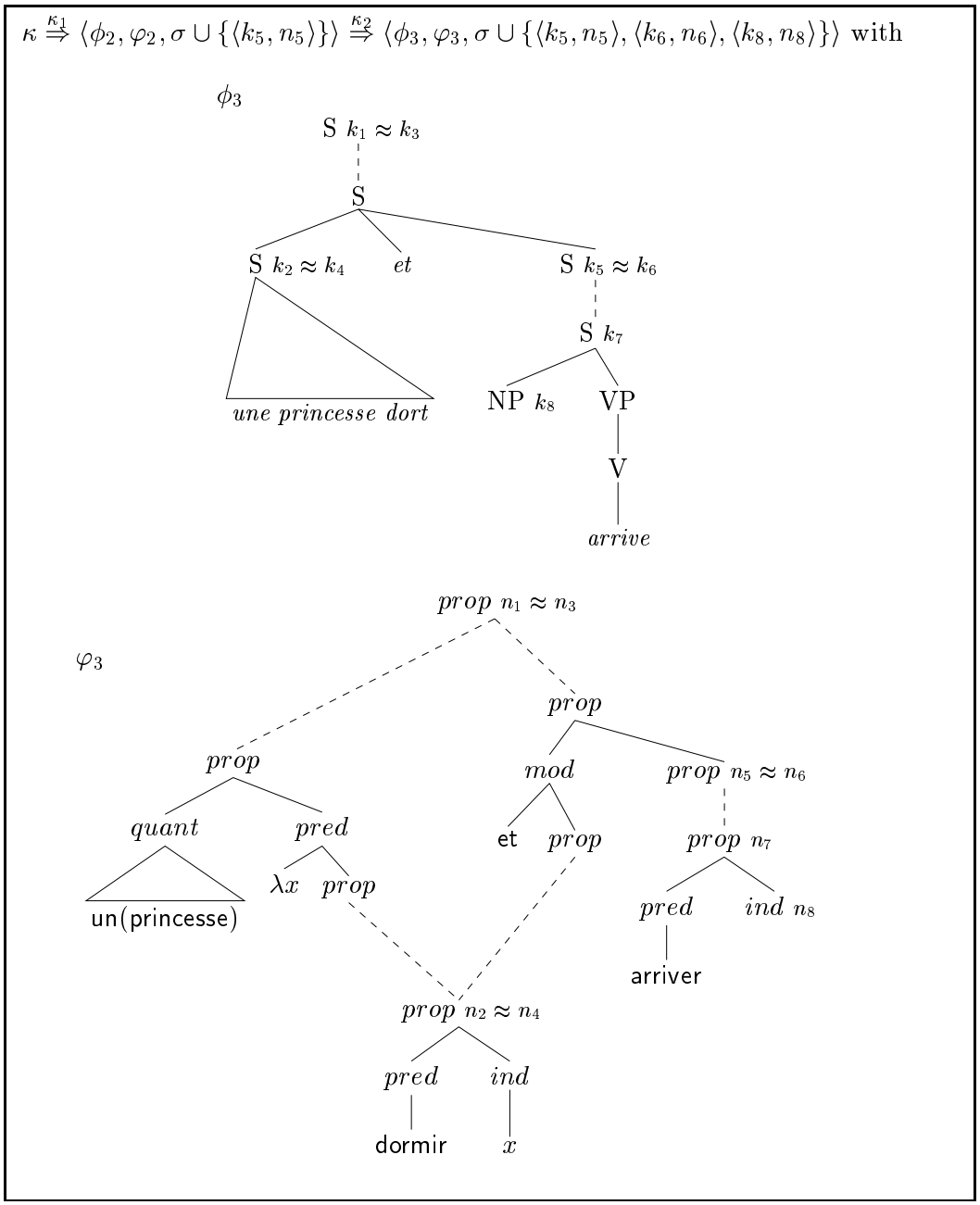


FIGURE 5.31: CONFIGURATION DERIVED BY ADDING THE ELEMENTARY CONFIGURATIONS FOR *et* AND *arrive* TO THE CONFIGURATION DERIVED FOR *une princesse dort*

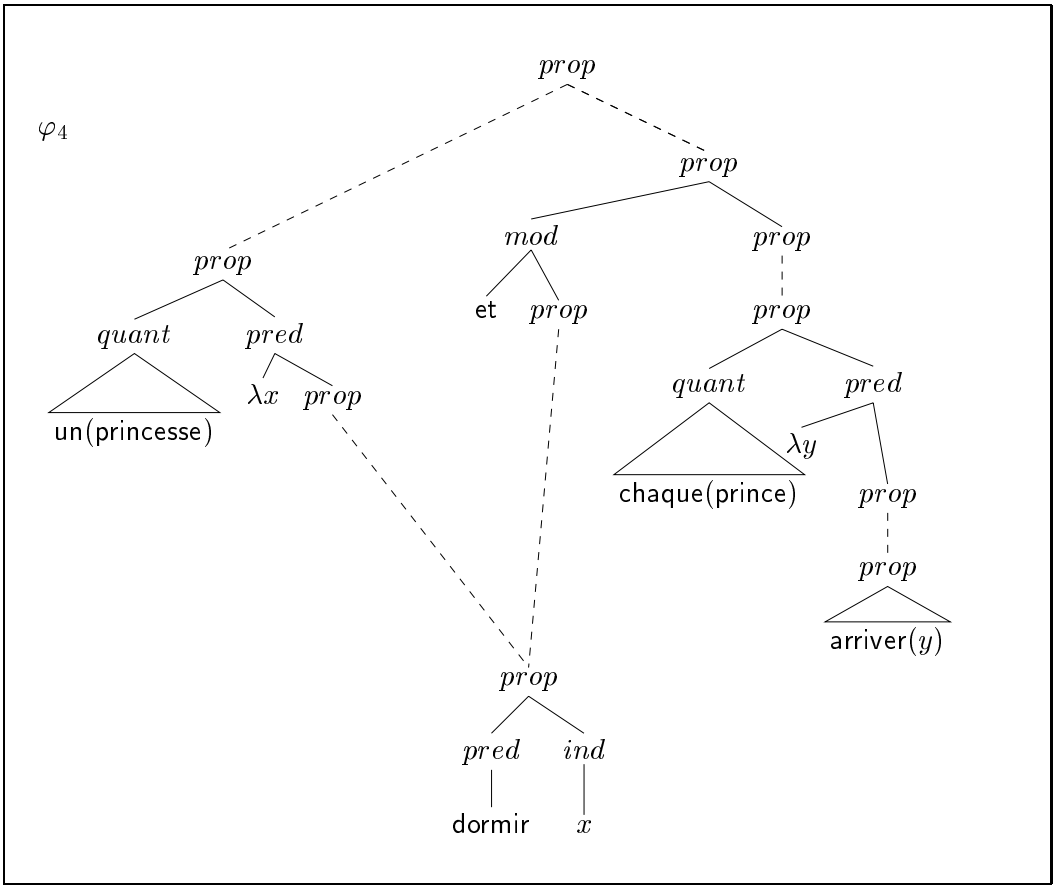


FIGURE 5.32: DESCRIPTION DERIVED IN G_{sem} FOR *une princesse dort et chaque prince arrive*

In (36) for the quantifier *chaque grenouille* it is not possible to outscope *une princesse*.

The derivation of a configuration for (36) is outlined in Fig. 5.33, 5.34 and 5.35. First the configuration κ in Fig. 5.33 for *Jean aime une princesse* is derived. In the next step, the elementary configuration with the verb of the relative clause, *adore*, as anchor must be added where the subject is relativized. This is the configuration κ_1 shown in Fig. 5.33. Since the noun *princesse* is modified by the relative clause, the new equivalences are $k_1 \approx k_3$ and $k_2 \approx k_8$ in G_{syn} and $n_1 \approx n_3$ and $n_2 \approx n_5$ in G_{sem}. In the next derivation step, either the elementary configuration κ_2 (Fig. 5.34) of the relative pronoun or the elementary configuration of the quantifier *chaque* can be added. (The order is of no importance.) When adding κ_2, the new equivalences must be $k_5 \approx k_9$, $k_6 \approx k_{10}$ and $k_8 \approx k_{11}$ in G_{syn} and $n_3 \approx n_9$, $n_4 \approx n_{10}$, $n_6 \approx n_{11}$ and $n_8 \approx n_{12}$ in G_{sem} and the result is a configuration where the description derived in G_{sem} is the description φ_2 shown in Fig. 5.34. The elementary configuration of the quantifying phrase *chaque* in the relative clause must be added in such a way that the derivation descriptions are the elementary descriptions for *adore*, the anchor of the relative clause. Therefore the quantifier must be "inserted" between n_{13} and n_{14} in φ_2. The resulting

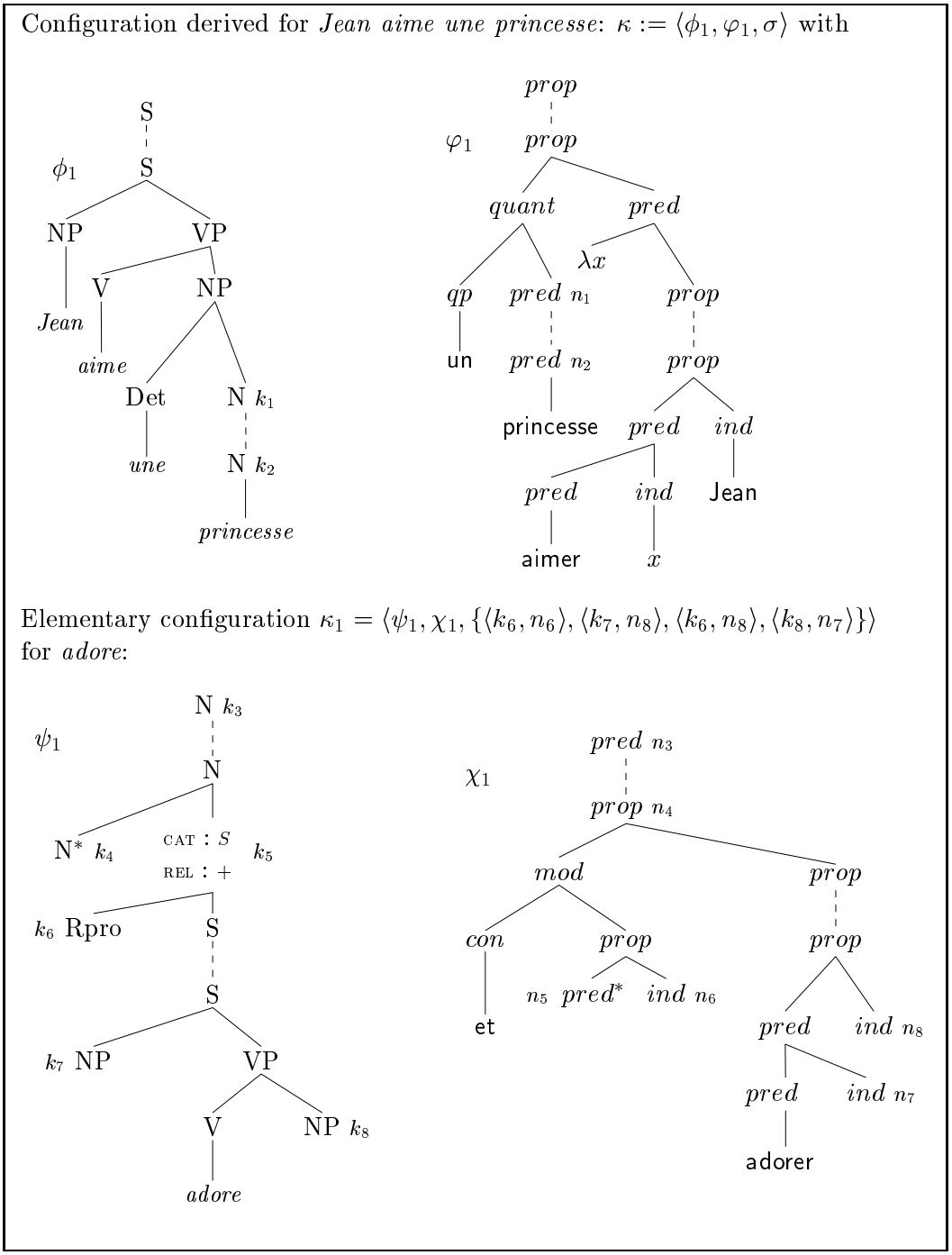


FIGURE 5.33: CONFIGURATION DERIVED FOR *Jean aime une princesse* AND ELEMENTARY CONFIGURATIONS FOR *adore* AS ANCHOR OF A RELATIVE CLAUSE

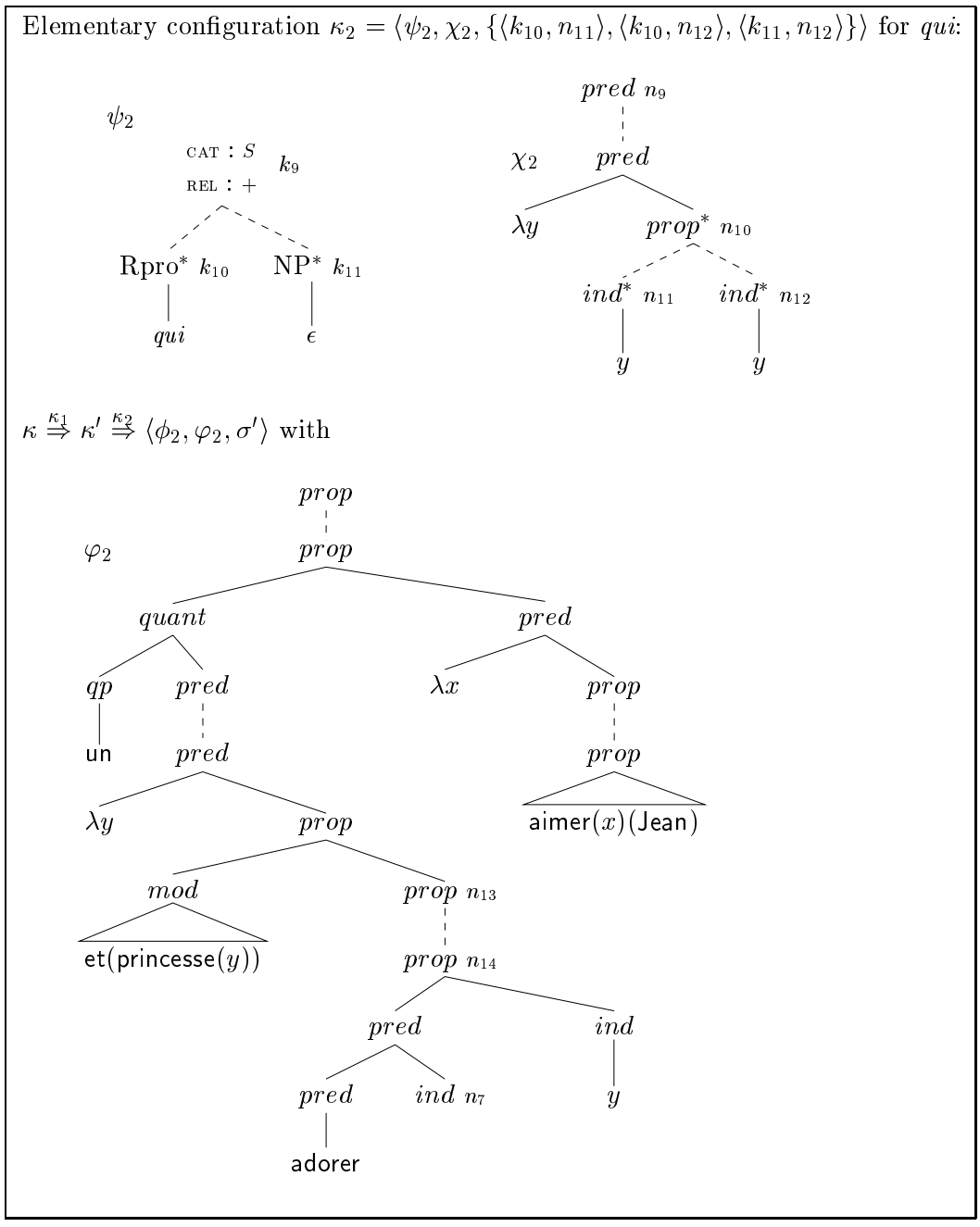


FIGURE 5.34: CONFIGURATION DERIVED FOR *Jean aime une princesse qui adore*

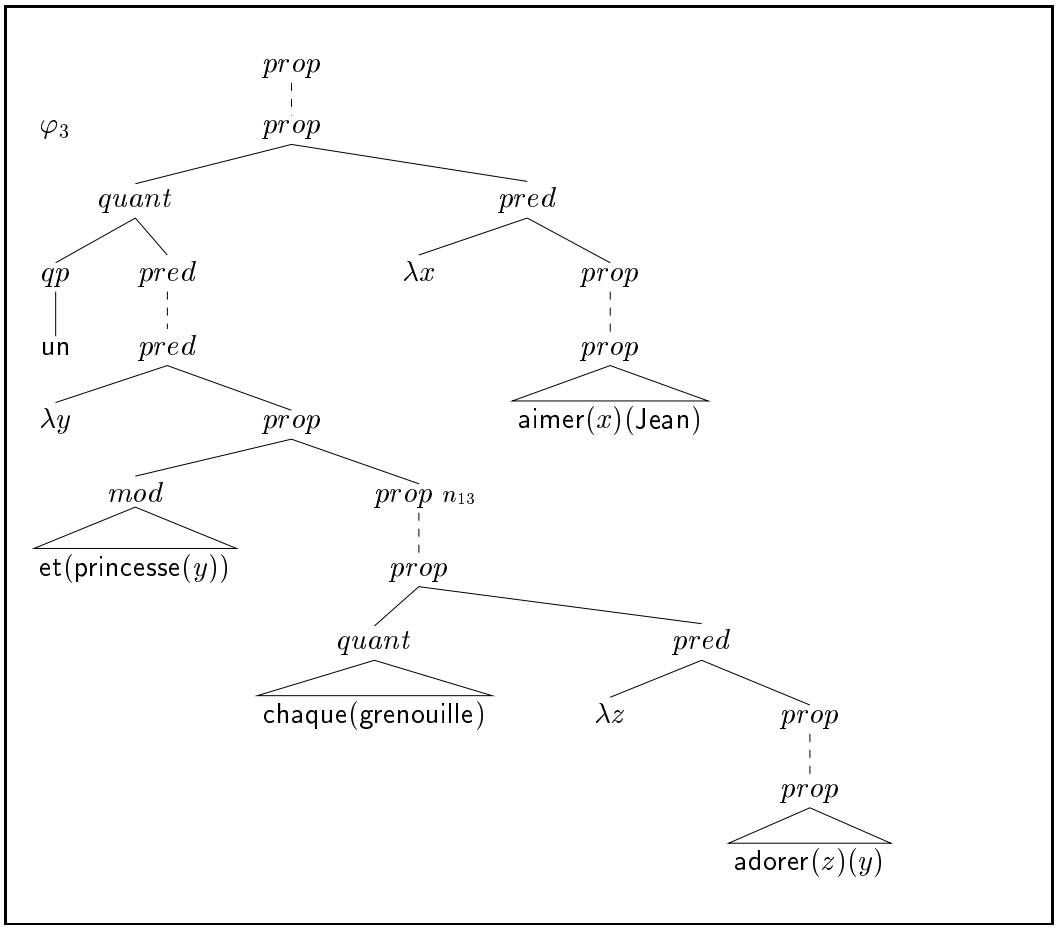


FIGURE 5.35: DESCRIPTION DERIVED IN G_{sem} FOR *Jean aime une princesse qui adore chaque grenouille*

description derived for (36) in G_{sem} is φ_3 in Fig. 5.35. φ_3 has only one minmal tree yielding the following term:

$$\mathsf{un}(\lambda y(\mathsf{et}(\mathsf{princesse}(y))(\mathsf{un}(\mathsf{grenouille})(\lambda z((\mathsf{adorer}(z))(y))))))(\lambda x((\mathsf{aimer}(x))(\mathsf{Jean})))$$

In other words, only the reading with narrow scope of *chaque grenouille* is allowed, i.e. the relative clause constraint is respected. The reason for this restriction is the fact that the elementary configuration for the anchor of the relative clause is added to the elementary descriptions of the modified noun. The locality restriction for the derivation in synchronous local TDGs then signifies that a quantifier occurring in the relative clause cannot raise out of the predicate denoted by the modified noun, i.e. this predicate is an island for quantifier scope.

As a further example where the relative clause constraint and the logical rule both exclude some of the combinatorially possible scope orders, consider (37):

(37) chaque élève lit un livre qu'un professeur recommande
each student reads a book that a professor recommends

In (37), it should not be possible for *un professeur* to outscope *un livre*. This is excluded by the relative clause constraint. Furthermore, because of the logical rule, either *chaque élève* has wide scope or *chaque élève* is in the scope of *un livre* and *un professeur*.

For this example, I will not mention the single derivation steps. The description derived in G_{sem} for (37) is shown in Fig. 5.36. This description has two minimal trees, and the terms yielded by these trees are the following:

1. $\mathsf{un}(\lambda z(\mathsf{et}(\mathsf{livre}(z))(\mathsf{un}(\mathsf{professeur})(\lambda v(\mathsf{recommander}(z)(v))))))$
$(\lambda x(\mathsf{chaque}(\mathsf{eleve})(\lambda y(\mathsf{lire}(y)(x)))))$

2. $\mathsf{chaque}(\mathsf{eleve})$
$(\lambda y(\mathsf{un}(\lambda z(\mathsf{et}(\mathsf{livre}(z))(\mathsf{un}(\mathsf{professeur})(\lambda v(\mathsf{recommander}(z)(v))))))$
$(\lambda x(\mathsf{lire}(y)(x)))))$

Only two of the $3! = 6$ combinatorially possible readings are allowed. There are two restrictions that apply in this case: firstly, the quantifier *un professeur* occurring in the relative clause modifying the predicate *livre* may not outscope *un livre*. This corresponds to the relative clause constraint. By this restriction, 3 of the 6 readings are excluded. Secondly, the term $(\mathsf{lire}(x))(y)$ must be part of the body of the quantifying phrase *chaque* and it must be part of the body of the quantifying phrase *un* in *un livre*. Therefore, the quantifying phrase *chaque* cannot be in the restriction of *un* in *un livre*. Consequently, if *chaque élève* is in the scope of *un livre*, then it must be in the body of the quantifying phrase of *un livre* and therefore cannot outscope *un professeur*. This last restriction corresponds to the logical rule. It follows from the syntax of the logic $TY(2)$, namely from the fact the restriction of a quantifying phrase is left of the body of a quantifying phrase and consequently a quantifier cannot occur both in the restriction and the body of another quantifier.

As this example has shown, the logical rule holds in the framework presented here. Since elementary descriptions in G_{sem} are supposed to describe the syntactic structure of semantic

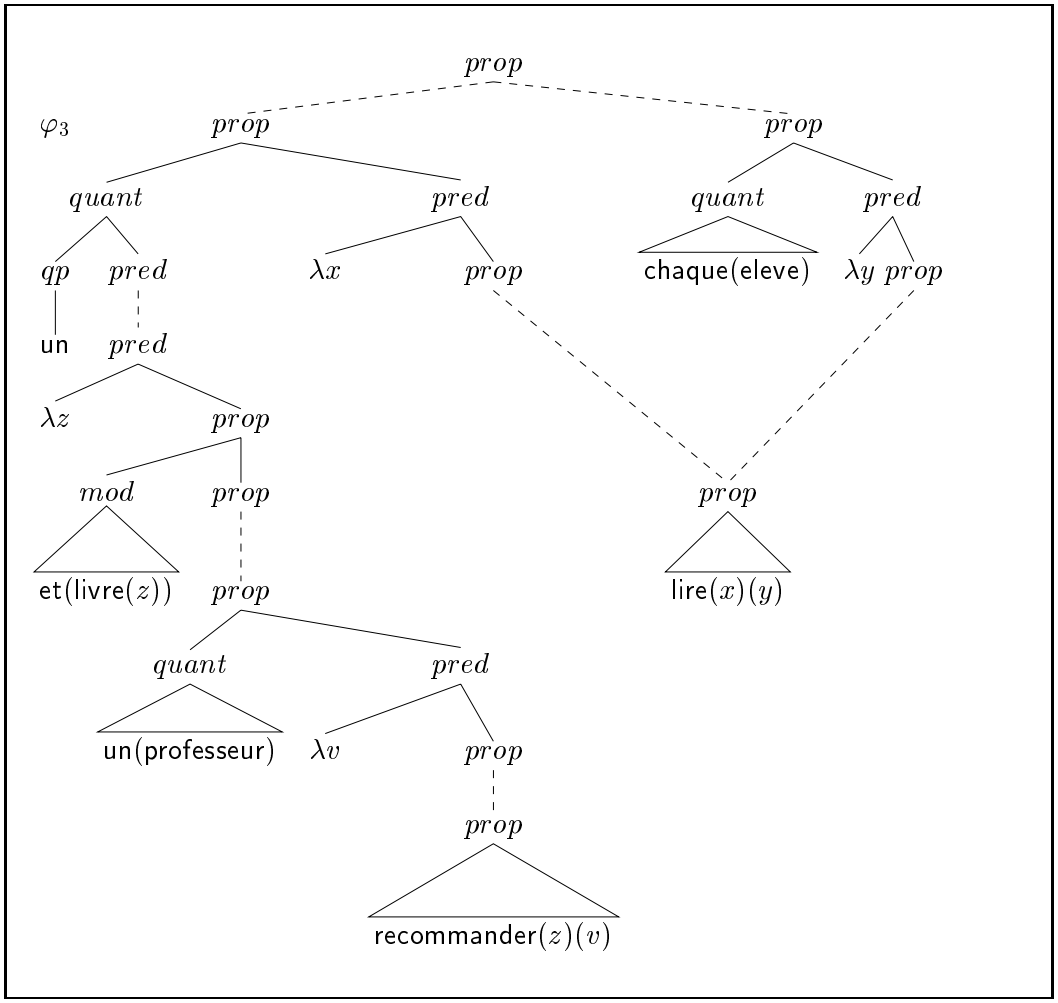


FIGURE 5.36: DESCRIPTION DERIVED IN G_{sem} FOR *chaque élève lit un livre qu'un professeur recommande*

expressions, it is not surprising that the elementary descriptions of G_{sem} do not allow scope orders that do not satisfy the logical rule.

The configurations derived for (35) and (36) have shown that island constraints hold as a consequence of the locality of the derivation in synchronous local TDGs. This is one of the advantages of local TDGs. As in the case of TAGs, the elementary objects in the grammar allow the definition of larger local domains. This gives a way of including island constraints without explicitely stating them.[3] Island constraints arise as a consequence of the specific form of the elementary configurations which is motivated by the principles observed for the construction of elementary trees in TAGs for natural language, in particular the predicate-argument cooccurrence principle. In frameworks that do not offer this extended domain of

[3]For similar reasons, island constraints for wh-movement are respected within a TAG-analysis (see Kroch 1987).

locality such constraints must be explicitely stated, they are not given by the lexical entries themselves.

5.5.3 Comparison to related work

Among recent theories of underspecified semantics, there are two approaches that both capture scope ambiguities using tree descriptions as underspecified representations and that are therefore similar to the approach presented in the course of this chapter. These are first the Ambiguous Logical Forms proposed by Muskens (1995) and second the context constraints over finite trees proposed in Niehren et al. 1997a,b (see also Pinkal 1995).

Ambiguous Logical Form

Muskens (1995) proposes to represent scope ambiguities by partial tree descriptions that are very similar to those generated with local TDGs. These tree descriptions consist of node names $l_1, l_2, \ldots$ together with dominance ($\lhd^*$) and parent ($\lhd$) relations. The node names are labels of Compositional DRT (CDRT, see Muskens 1996) formulas. For simplicity, in this presentation I will use a first order logic with $\forall$ and $\exists$ instead of CDRT. A node name l_i is label of the formula p_i.

(38) every boy adores a girl

For the quantifier scope ambiguity in (38), the following underspecified representation is generated:

$$
\begin{array}{ll}
l_1 \lhd^* k_{14} & \\
l_1 \lhd^* k_{16} & \\
l_{14} \lhd l_{15} & p_{14} = \forall u_1 : \mathsf{boy}(u_1) \rightarrow p_{15} \\
l_{16} \lhd l_{17} & p_{16} = \exists u_2 : \mathsf{girl}(u_2) \wedge p_{17} \\
l_{17} \lhd^* l_2 & p_2 = \mathsf{adores}(u_2)(u_1) \\
l_{15} \lhd^* l_2 &
\end{array}
$$

Disambiguation leads to the two desired readings: either $p_1 = p_{14}$, $p_{15} = p_{16}$ and $p_{17} = p_2$ which corresponds to wide scope of the universal quantifier, or $p_1 = p_{16}$, $p_{17} = p_{14}$ and $p_{15} = p_2$ which corresponds to wide scope of the existential quantifier.

The similarity to the underspecified representations derived with local TDGs is obvious. However, there are some crucial differences concerning the way underspecified representations are generated.

Muskens' Ambiguous Logical Forms are generated in the following way: a phrase structure grammar is used to describe the syntactic structure. Each phrase structure rule is connected to an l-description and an s-description. l-descriptions are used to build a tree description representing scope and s-descriptions are used to generate formulas that are connected to the node-names of the tree description.

Since each phrase structure rule is connected to l- and s-descriptions, the relation between syntactic structure and logical form is closer than with synchronous local TDGs. Therefore it is necessary to introduce island constraints in an explicit way. For each node name, an island must be defined in the l-descriptions that are related to the phrase structure rules. The rule

$RC \rightarrow RPRO\ S$ for example (relative clause to relative pronoun and sentence) is connected to the l-description $l_M \lhd l_{D2}, I(l_{D2}) = l_{D2}$. "$M$" stands for "mother" and "$D2$" for "second daughter". This l-description signifies that the node name of the second daughter (i.e. the label of the interpretation of the relative clause) is an island.

The result is that island constraints are respected but they are not predicted by more general rules. Whereas with synchronous local TDGs, island constraints are a consequence of general principles such as locality and the predicate-argument cooccurrence principle that were introduced for reasons independent from island constraints.

A second limitation of Muskens' approach is that underspecification concerns only the logical form and not the syntactic structure. The phrase structure rules used for the generation of syntactic trees do not allow underspecification of the dominance relation. However, we have seen that this is desirable to account for structural ambiguities such as the nominal modifier ambiguities considered in (33) on page 211. With synchronous local TDGs, underspecification of syntax and semantics is allowed in the same way.

Context constraints over finite trees

Niehren, Pinkal and Ruhrberg (1997b) propose to describe underspecified semantic information by constraints interpreted over finite trees. These constraints are called *context constraints*. The language of context constraints is powerful enough to describe equality, subtree and "equality-up-to" relations over trees.

Context constraints contain variables $X_1, X_2, \ldots$ for trees and variables $C_1, C_2, \ldots$ for a tree with a *hole*, i.e. a slot where a subtree must be inserted.

(39) two languages are spoken by many linguists

With these context constraints, underspecified representations can be described. For the ambiguity in (39), the following context constraint is derived:

$$X_1 = C_1(((\mathsf{two})(\mathsf{language}))(\lambda x(C_3(X_2))))$$
$$\wedge\, X_1 = C_2(((\mathsf{many})(\mathsf{linguist}))(\lambda y(C_4(X_2))))$$
$$\wedge\, X_2 = (\mathsf{spoken\text{-}by})(y)(x)$$

If first order variables are viewed as node names and second order variables as strong dominances, then this context constraint corresponds to the underspecified representations obtained with local TDGs.

Context unification leads to a disjunction of the following solved constraints:

1. $X_1 = C_1(((\mathsf{two})(\mathsf{language}))(\lambda x(C_5(((\mathsf{many})(\mathsf{linguist}))(\lambda y(C_4((\mathsf{spoken\text{-}by})(y)(x))$
2. $X_S = C_2(((\mathsf{many})(\mathsf{linguist}))(\lambda y(C_6(((\mathsf{two})(\mathsf{language}))(\lambda x(C_3((\mathsf{spoken\text{-}by})(y)(x))$

To obtain the minimal solutions, a closure operation is applied that consists of identifying the remaining free variables $C_1, C_2, \ldots$ with the identity context.

The context constraints are obtained from syntax trees. For each node, conjuncts are added depending on the node and its daughters.

Niehren et al. (1997b) show that context constraints can even capture parallelism between trees since the equality up-to relation over trees can be expressed by them. This can be used to capture parallelism between scope orders of quantifiers in elliptic clauses.

However, with context constraints, there arise the same problems as in the case of Muskens' Ambiguous Logical Form: firstly, the formalism of context constraints does not offer an extended domain of locality and therefore islands must be explicitely specified. Secondly, underspecification is limited to the semantic representation.

A further problem is that, according to Niehren et al. (1997a), the decidability of context unification is an open problem. In contrast to this, the disambiguation process for local TDGs, i.e. the construction of all minimal trees, is decidable, as I have shown in the proof of Prop. 3.9, page 96.

5.6 Conclusion

In this chapter, I have presented a fragment of a syntax-semantics interface for French, based on the formalism of synchronous local TDGs introduced in Section 4.4 in Chapter 4. The architecture of the syntax-semantics interface is such that one TDG G_{syn} is related to a second local TDG G_{sem} by a synchronization relation Σ such that $\langle G_{syn}, G_{sem}, \Sigma \rangle$ form a pair of synchronous local TDGs. The first TDG, G_{syn}, describes the syntactic structure of French whereas the second TDG, G_{sem} describes the syntactic structure of expressions in a truth conditional logic.

The use of synchronous local TDGs has several advantages:

- Tree descriptions as underspecified representations: since the grammars consist of tree descriptions, i.e. of representations that are underspecified by nature, we do not need any extra mechanisms in order to obtain underspecified representations for scope ambiguities. No metavariables and constraints on these variables must be introduced as it is the case in Bos 1995, Richter and Sailer 1997 or Reyle 1993. Therefore local TDGs enable us to generate underspecified representations in a very elegant and natural way.

- Extended domain of locality: syntax and semantics depend on each other in a compositional way but this dependence is less close than in more traditional Montagovian approaches where each syntactic phrase structure rule is connected to a semantic operation. This is a consequence of the larger domain of locality provided by TAG-like formalisms such as local TDGs, in other words it follows from the possibility to have larger elementary descriptions. Jacobs (1995) for example also argues that there is a systematic compositional relation between syntax and semantics but that this connection is less close than most traditional theories assumed. Both, syntax and semantics, have their own and independent structural principles.

  As a consequence of this weakening of the relation between syntax and semantics, a quantifying-in mechanism is not needed. Furthermore, it is not necessary to define different types for a lexical item depending on the different syntactic constructions it occurs in (except in cases of real lexical ambiguities, of course). In general, types of constants may be less complex. The way they are combined with other constants

depends on the syntactic context and is specified by the form of the correponding elementary descriptions.

Another consequence is that trees in the tree language of G_{sem} can be simply seen as syntactic trees of logical expressions. There is no need to define specific semantic operations for internal nodes.

- Locality of the derivation: in the preceding section we have seen that as a consequence of the locality of the derivation island constraints for quantifier scope are respected. This means that island constraints are a consequence of the more general locality principle together with the principles that hold for the elementary descriptions of a local TDG, in particular the predicate-argument cooccurrence principle.

Summarizing one can say that the architecture of synchronous local TDGs combines the advantages of TAG-like formalisms, in particular the extended domain of locality, with the idea of partial representations for scope ambiguities. Therefore the use of this formalism for the description of a syntax-semantics interface offers an attractive alternative to theories in the tradition of Montague that assume a close relation between syntax and semantics.

Chapter 6

Conclusion

In this thesis I have presented a formal framework for the generation of tree descriptions. This framework, called *(local) Tree Description Grammar (TDG)* was motivated by the following two goals:

- the formalism was supposed to be an extension of Tree Adjoining Grammars in the following sense: its generative capacity should be greater than the generative capacity of TAGs, and TDGs should generate tree descriptions from elementary descriptions in such a way that an extended domain of locality was allowed, as it is also the case in TAGs.
- the formalism should allow the generation of underspecified representations for scope ambiguities. In particular an underspecification of the dominance relation between the nodes in a tree should be possible, since this is necessary to account for structural ambiguities.

In a first approach, a general definition of TDGs has been presented. This comprises a definition of the tree logic and the specific descriptions allowed in TDGs, and a definition of the derivation mode. Motivated by the desire to show a limitation of the generative power of the formalism, TDGs have been further restricted. This has lead to the central formalism of the thesis, *local Tree Description Grammars.*

I have shown that a consequence of the locality restriction is that the derivation process becomes context-free. Therefore, semilinearity of the string languages can be shown via letter equivalence to context-free languages. This implies that local Tree Description Languages are of constant growth, a property generally ascribed to natural languages.

The requirements mentioned above are satisfied by local TDGs:

Local TDGs are an extension of TAGs, since I have shown that local TDGs are more powerful than set-local MC-TAGs. The extra power in comparison to set-local MC-TAGs arises from the fact that the locality restriction concerns only the derivation of descriptions, whereas the definition of a minimal tree does not contain any locality restriction.

As the elementary objects of local TDGs are tree descriptions, an extended domain of locality as in TAGs is provided by this formalism. Consequently, local TDGs can be lexicalized, and local dependencies such as filler gap dependencies can be expressed in elementary descriptions. For lexicalized local TDGs, the word recognition problem is decidable.

The generation of underspecified representations is possible in the local TDG framework. Even underspecification of the dominance relation is allowed and therefore structural ambiguities can be analyzed by one single tree description.

Local TDGs of different rank can be distinguished depending on the form of derivation steps that are possible in these grammars. I have introduced a hierarchy of local TDGs. For the string languages generated by local TDGs of a certain rank, a pumping lemma has been shown. With this pumping lemma, it is possible to prove that local TDGs of rank n can generate a language $L_i := \{a_1^k \cdots a_i^k \mid k \geq 0\}$ iff $i \leq 2n$ holds.

In order to describe the relation between two languages, I have introduced synchronous local TDGs. For this formalism, I have shown that the synchronization with a second local TDG does not increase the generative power of the grammar in the sense that each language generated by a local TDG that is part of a synchronous pair of local TDGs, also can be generated by a single local TDG. However, it might be the case that the rank of the local TDG is increased.

As a linguistic application, this formalism of synchronous local TDGs has been used to describe a syntax semantics interface for a fragment of French. This illustrates the derivation of underspecified representations for scope ambiguities with local TDGs. I have shown that, as a consequence of the extended domain of locality together with the locality restriction for derivations, island constraints for quantifier scope hold in this framework without being explicitely stated.

Besides several points concerning details of the formalism, there are mainly two issues that have been left aside and that are obviously of interest: firstly, parsing has not been considered, and therefore no result concerning parsing complexity has been shown. Secondly, I have not treated the question whether there is a description-based formalization of local TDGs that is not derivation-based and how such a formalization could look like. The consideration of these two issues awaits future research.

Appendix A

Additional Proofs

In this appendix, those proofs are listed that are more technical and not relevant for the intuition of the reader and that were therefore left aside in the chapters of the thesis.

Proofs from Chapter 3

First the proof of Lemma 3.3, page 85 is shown.

Lemma 3.3 *If $\phi_1 \overset{\psi}{\Rightarrow} \phi_2$ is a derivation step in a TDG, then*

(LP) *for all k_1, k_2: $\phi_2 \vdash k_1 \prec k_2$ iff there are k_1', k_2' with $\phi_2 \vdash k_1' \lhd^* k_1 \wedge k_2' \lhd^* k_2$ and $\phi_1 \wedge \psi \vdash k_1' \prec k_2'$.*

Proof Let $\phi_1 \overset{\psi}{\Rightarrow} \phi_2$ be a derivation step in a TDG.
Suppose that $\phi_2 \vdash k_1 \prec k_2$ for some k_1, k_2 such that there are no k_1', k_2' with $\phi_2 \vdash k_1' \lhd^* k_1 \wedge k_2' \lhd^* k_2$ and $\phi_1 \wedge \psi \vdash k_1' \prec k_2'$.
$\Rightarrow$ either $k_1 \prec k_2$ is a conjunct in ϕ_2 or it can be derived by inference rules $tr_{\mathcal{L}}$ or T_6.
If there are node names k with $\phi_2 \vdash k_1 \prec k \wedge k \prec k_2$, then there are k_l, k_l' such that: $\phi_2 \vdash k_1 \prec k_l \wedge k_l \prec k_l' \wedge k_l' \prec k_2$ and there is no k_l'' with $\phi_2 \vdash k_l \prec k_l'' \wedge k_l'' \prec k_l'$, and there are no k_d, k_d' with $\phi_2 \vdash k_d \lhd^* k_l \wedge k_d' \lhd^* k_l'$ and $\phi_1 \wedge \psi \vdash k_d \prec k_d'$.
$\Rightarrow$ either $k_1 \prec k_2$ is a conjunct in ϕ_2
or there are k_l, k_l' with $\phi_2 \vdash k_1 \prec k_l \vee k_1 \approx k_l$ and $\phi_2 \vdash k_l' \approx k_2 \vee k_l' \prec k_2$, such that

- $\phi_2 \vdash k_l \prec k_l'$ and there is no k_l'' with $\phi_2 \vdash k_l \prec k_l'' \wedge k_l'' \prec k_l'$,
- there are no k_d, k_d' with $\phi_2 \vdash k_d \lhd^* k_l \wedge k_d' \lhd^* k_l'$ and $\phi_1 \wedge \psi \vdash k_d \prec k_d'$,
- and either $k_l \prec k_l'$ is a conjunct in ϕ_2 or there are k_d^1, k_d^2 with $\phi_2 \vdash k_d^1 \lhd^* k_l \wedge k_d^2 \lhd^* k_l' \wedge k_d^1 \prec k_d^2$ and $\phi_1 \wedge \psi \nvdash k_d^1 \prec k_d^2$, and there are no $k_d^{1'} \neq k_d^1, k_d^{2'} \neq k_d^2$ with $\phi_2 \vdash k_d^{1'} \lhd^* k_d^1 \wedge k_d^{2'} \lhd^* k_d^2 \wedge k_d^{1'} \prec k_d^{2'}$. (In this case $k_d^1 \prec k_d^2$ is a conjunct in ϕ_2.)

$\Rightarrow$ there are k_1^l, k_2^l such that ϕ_2 contains $k_1^l \prec k_2^l$ as conjunct, and if ϕ_2' is obtained from ϕ_2 by removing this conjunct, then $\phi_2' \nvdash k_1^l \prec k_2^l$ holds.

$\Rightarrow \phi_2'$ is a description (since $\phi_1 \wedge \psi \not\vdash k_1^l \prec k_2^l$ and because of *(A3)*, there is no k with $\phi_2' \vdash k \lhd k_1^l \wedge k \lhd k_2^l$), and ϕ_2' also satisfies *(A1)* to *(A4)*, but $\phi_2 \vdash \phi_2'$ and $\phi_2' \not\vdash \phi_2$ holds. Contradiction to *(A5)*.

□

The following proof is the proof of Lemma 3.4 on page 86.

Lemma 3.4 *If $\phi_1 \overset{\psi}{\Rightarrow} \phi_2$ is a derivation step in a TDG, and $\phi_\approx := \bigwedge\{k_\phi \approx k_\psi \,|\, k_\phi \in node(\phi_1), k_\psi \in node(\psi), \phi_2 \vdash k_\phi \approx k_\psi\}$, then*

(D) *for all k_1, k_2 with $\phi_2 \vdash_s k_1 \lhd^* k_2$:*

(a) either $\phi_1 \wedge \psi \wedge \phi_\approx \vdash k_1 \lhd^* k_2$

(b) or there are $k_m \in M_\psi$, $k \in node(\phi_1)$ and $k' \in node(\phi_2)$ such that

- *$\phi_2 \vdash k_m \approx k$,*
- *either $\psi \vdash_s k_m \lhd^* k_2$ and $\phi_1 \vdash k \lhd k'$, or $\phi_1 \vdash_s k \lhd^* k_2$ and $\psi \vdash k_m \lhd k'$,*
- *and $\phi_2 \vdash k' \lhd^* k_1$.*

Proof Let $\phi_1 \overset{\psi}{\Rightarrow} \phi_2$ be a derivation step in a TDG with $\phi_\approx$ as in Lemma 3.4.
To show: there is a $D = \{k_{11} \lhd^* k_{21}, \cdots, k_{1n} \lhd^* k_{2n}\}$ such that

(a) $\phi_2 \vdash k_{11} \lhd^* k_{21} \wedge \cdots \wedge k_{1n} \lhd^* k_{2n}$,

(b) for all $i, 1 \leq i \leq n$, $\phi_1 \wedge \psi \wedge \phi_\approx \not\vdash k_{1i} \lhd^* k_{2i}$

(c) for all $k_\psi \in node(\psi)$: there is exactly one $k_d \in node(\phi) \cup node(\psi)$ with $k_d \lhd^* k_\psi \in D$ iff

1. there is a $k_m \in M_\psi$ with $\phi_1 \vdash_s k_m \lhd^* k_\psi$,
2. there are $k, k' \in node(\phi_1)$ with $\phi_\approx \vdash k \approx k_m$ and $\phi_1 \vdash k \lhd k'$,
3. and there is no $k_m' \in K_\psi$ with $\psi \vdash k_\psi \lhd^* k_m'$.

(Then for one k_d such that $\phi_1 \vdash k \lhd^* k_d$ and there is no k_p with $\phi_1 \vdash k_d \lhd k_p$, the formula $k_d \lhd^* k_\psi$ is in D.)

(d) for all $k_\phi \in node(\phi_1)$: there is exactly one $k_d \in node(\phi) \cup node(\psi)$ with $k_d \lhd^* k_\phi \in D$ iff

1. there is a k_s with $\phi_1 \vdash_s k_s \lhd^* k_\phi$,
2. there is a $k_m \in M_\psi$ with $\phi_\approx \vdash k_m \approx k_s$ such that there is a k with $\psi \vdash k_m \lhd k$.
3. there are no $k_\phi' \in node(\phi_1)$ and $k_m \in K_\psi$ with $\psi \vdash k_1 \lhd^* k_m$, $\phi_\approx \vdash k_\phi' \approx k_m$ and $\phi_1 \vdash k_\phi \lhd^* k_\phi'$.

(Then for one $k_d \in node(\psi)$ with $\psi \vdash k_m \lhd^* k_d$ such that there is no k_p with $\psi \vdash k_d \lhd k_p$, the formula $k_d \lhd^* k_\phi$ is in D.)

Suppose that

- either there is a $k_\psi \in node(\psi)$ such that conditions 1. to 3. in (c) hold for k_ψ and there is no k_d with $\phi_2 \vdash k_d \lhd^* k_\psi$.
  $\Rightarrow$ there are k_1', k_2' with $\phi_2 \vdash k \lhd^* k_1' \wedge k_1' \lhd k_2'$ and $\phi_2 \vdash_s k_1' \lhd^* k_\psi$.
- or there is a $k_\phi \in node(\phi_1)$ such that conditions 1. to 3. in (d) hold for k_ϕ and there is no k_d with $\phi_2 \vdash k_d \lhd^* k_\phi$.
  $\Rightarrow$ there are k_1', k_2' with $\phi_2 \vdash k_m lhd^* k_1' \wedge k_1' \lhd k_2'$ and $\phi_2 \vdash_s k_1' \lhd^* k_\phi$.

$\Rightarrow$ (because of condition *4.* in Def. 3.10) ϕ_2 is no description. Contradiction.

$\phi_2' := \phi_1 \wedge \psi \wedge \phi_{\approx} \wedge k_{11} \lhd^* k_{21} \wedge \cdots \wedge k_{1n} \lhd^* k_{2n}$
ϕ_2' is a description satisfying *(A1)* to *(A4)* because:

It is easy to see that conditions 1. to 3. and the 5. condition in Def. 3.10 hold because ϕ_1 and ψ are descriptions and for the minimal name in ψ, there is an equivalent name in ϕ_1.

Condition 4. in Def. 3.10 holds because of the addition of the dominances in D.

(A1) to *(A4)* also hold.

Since $\phi_2 \vdash \phi_2'$ holds and ϕ_2 satisfies *(A5)*, also $\phi_2' \vdash \phi_2$ must hold, and therefore **(D)** holds for ϕ_2.

□

List of Figures

Bibliography

Abeillé, A.: 1988a, A lexicalized tree adjoining grammar for French: The general framework, *Technical Report MS-CIS-88-64*, University of Pennsylvania.

Abeillé, A.: 1988b, Parsing French with tree adjoining grammar: some linguistic accounts, *Proceedings of COLING*, Budapest, pp. 7–12.

Abeillé, A.: 1990, Lexical and syntactic rules in a Tree Adjoining Grammar, *Proceedings of ACL*, pp. 292–298.

Abeillé, A.: 1992, Synchronous TAGs and French pronominal clitics, *Proceedings of COLING*, Nantes, pp. 60–66.

Abeillé, A.: 1993, *Les Nouvelles Syntaxes. Grammaires d'unification et analyse du français*, Armand Colin, Paris.

Abeillé, A.: 1994, Syntax or semantics? Handling nonlocal dependencies with MCTAGs or synchronous TAGs, *Computational Intelligence* **10**(4), 471–485.

Abeillé, A., Schabes, Y. and Joshi, A. K.: 1990, Using lexicalized TAGs for machine translation, *Proceedings of COLING*, Helsinki.

Alshawi, H.: 1990, Resolving quasi logical form, *Computational Linguistics* **16**(3), 133–144.

Alshawi, H. and Crouch, R.: 1992, Monotonic Semantic Interpretation, *Proceedings of ACL*.

Alshawi, H. (ed.): 1992, *The Core Language Engine*, MIT Press.

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

Barwise, J. and Cooper, R.: 1981, Generalized quantifiers and natural language, *Linguistics and Philosophy* **4**, 159–219.

Becker, T., Joshi, A. K. and Rambow, O.: 1991, Long-distance scrambling and tree adjoining grammars, *Proceedings of ACL-Europe*.

Becker, T., Rambow, O. and Niv, M.: 1992, The Derivationel Generative Power of Formal Systems or Scrambling is Beyond LCFRS, *Technical Report IRCS-92-38*, Institute for Research in Cognitive Science, University of Pennsylvania.

Bos, J.: 1995, Predicate logic unplugged, *in* P. Dekker and M. Stokhof (eds), *Proceedings of the 10th Amsterdam Colloquium*, pp. 133–142.

Bresnan, J., Kaplan, R. M., Peters, S. and Zaenen, A.: 1982, Cross-serial dependencies in Dutch, *Linguistic Inquiry* **13**(4), 613–635. Reprinted in Savitch et al. (1987).

Carpenter, B.: 1997, *Type-Logical Semantics*, MIT Press.

Chierchia, G. and McConnell-Ginet, S.: 1990, *Meaning and Grammar. An Introduction to Semantics*, MIT Press.

Chomsky, N.: 1957, *Syntactic Structures*, Mouton & Co, The Hague.

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

Cooper, R.: 1983, *Quantification and Syntactic Theory*, Synthese Language Library, Reidel, Dordrecht.

Dawar, A. and Vijay-Shanker, K.: 1990, An Interpretation of Negation in Feature Structure Descriptions, *Computational Linguistics* **16**(1), 11–21.

Dowty, D. R., Wall, R. E. and Peters, S.: 1981, *Introduction to Montague Semantics*, Studies in Linguistics and Philosophy, Kluwer, Dordrecht.

Ebbinghaus, H.-D., Flum, J. and Thomas, W.: 1992, *Einführung in die mathematische Logik*, BI-Wissenschaftsverlag, Mannheim, Leipzig, Wien, Zürich.

Fauconnier, G.: 1976, *Etude de certains aspects logiques et grammaticaux de la quantification et de l'anaphore en français et en anglais*, PhD thesis, Université de Paris.

Friedrichsdorf, U.: 1992, *Einführung in die klassische und intensionale Logik*, Vieweg, Braunschweig/Wiesbaden.

Gallin, D.: 1975, *Intensional and Higher-Order Modal Logic with Applications to Montague Semantics*, North Holland mathematics studies 19, North-Holland Publ. Co., Amsterdam.

Grätzer, G.: 1978, *General lattice theory*, Mathematische Reihe, Birkhäuser, Basel, Stuttgart.

Groenink, A. V.: 1996, Mild context-sensitivity and tuple-based generalizations of context-free grammar, *Report CS-R9634*, Centrum voor Wiskunde en Informatica, Amsterdam.

Hobbs, J. R. and Shieber, S. M.: 1987, An algorithm for generating quantifier scopings, *Computational Linguistics* **13**, 47–63.

Hopcroft, J. E. and Ullman, J. D.: 1979, *Introduction to automata theory, languages and computation*, Addison Wesley. German translation in Hopcroft and Ullman 1994.

Hopcroft, J. E. and Ullman, J. D.: 1994, *Einführung in die Automatentheorie, Formale Sprachen und Komplexitätstheorie*, 3. edn, Addison Wesley.

Hümpel-Schökel, C.: 1993, *Über echte Teilklassen von indizierten Sprachen und ihre Einbettung in die Chomsky-Hierarchie*, PhD thesis, Universität Hannover.

Jacobs, J.: 1995, Wieviel Syntax braucht die Semantik? Möglichkeiten und Grenzen einer sparsamen Theorie der Bedeutungskomposition, *Arbeitspapiere des SFB 282 "Theorie des Lexikons" 73*, Heinrich Heine Universität Düsseldorf.

Johnson, M.: 1990, Expressing disjunctive and negative feature constraints with classical first-order logic, *Proceedings of ACL*, pp. 173–179.

Joshi, A. K.: 1985, Tree adjoining grammars: How much contextsensitivity is required ro provide reasonable structural descriptions?, *in* D. Dowty, L. Karttunen and A. Zwicky (eds), *Narural Language Parsing*, Cambridge University Press, pp. 206–250.

Joshi, A. K.: 1987, An introduction to Tree Adjoining Grammars, *in* A. Manaster-Ramer (ed.), *Mathematics of Language*, John Benjamins, Amsterdam, pp. 87–114.

Joshi, A. K., Levy, L. S. and Takahashi, M.: 1975, Tree Adjunct Grammars, *Journal of Computer and System Science* **10**, 136–163.

Kallmeyer, L.: 1996a, Tree Description Grammars, *in* D. Gibbon (ed.), *Natural Language Processing and Speech Technology. Results of the 3rd KONVENS Conference*, Mouton de Gruyter, Berlin, pp. 332–341.

Kallmeyer, L.: 1996b, Underspecification in Tree Description Grammars, *Arbeitspapiere des SFB 340 81*, Universität Tübingen.

Kallmeyer, L.: 1997a, Local Tree Description Grammars, *Proceedings of the Fifth Meeting on Mathematics of Language, DFKI Research Report*, pp. 77–84.

Kallmeyer, L.: 1997b, A syntax-semantics interface with Synchronous Tree Description Grammars, *in* G.-J. M. Kruijff, G. V. Morrill and R. T. Oehrle (eds), *Formal Grammar 1997. Linguistic Aspects of Logical and Computational Perspectives on Language*, Aix-en-Provence, pp. 112–124.

Kallmeyer, L.: 1998, A hierarchy of local TDGs, *Fourth International Workshop on Tree Adjoining Grammars and Related Frameworks, IRCS Report 98–12*, University of Pennsylvania, Philadelphia, pp. 76–79.

Kallmeyer, L.: 1999a, Synchronous Local TDGs and Scope Ambiguities, *in* G. Bouma, E. W. Hinrichs, G.-J. Kruijff and R. T. Oehrle (eds), *Constraints and Resources in Natural Language Syntax and Semantics*, CSLI. To appear.

Kallmeyer, L.: 1999b, Underspecification in Tree Description Grammars, *in* H. P. Kolb and U. Mönnich (eds), *The Mathematics of Syntactic Structure: Trees and their Logics*, Mouton de Gruyter. To appear.

Kamp, H. and Reyle, U.: 1993, *From Discourse to Logic*, Studies in Linguistics and Philosophy, Kluwer, Dordrecht, Boston, London.

Kasper, R. T. and Rounds, W. C.: 1990, The logic of unification in grammar, *Linguistics and Philosophy* **13**, 35–58.

King, P.: 1994, An expanded logical formalism for Head-Driven-Phrase-Structure Grammar, *Arbeitspapiere des SFB 340 59*, Universität Tübingen.

King, P. J. and Vaillette, N.: 1997, The expressive power of a Formalism for Head-Driven Phrase Structure Grammar. Unpublished manuscript.

König, E. and Reyle, U.: 1996a, A general reasoning scheme for underspecified representations, *in* M. Butt and T. H. King (eds), *Proceedings of the First LFG conference*, RANK Xerox, Grenoble.

König, E. and Reyle, U.: 1996b, A general reasoning scheme for underspecified representations, *in* H.-J. Ohlbach and U. Reyle (eds), *Logic and its Applications.Festschrift for Dov Gabbay. Part I*, Kluwer.

Kripke, S.: 1972, *Naming and Necessity*, Blackwell, Oxford UK and Cambridge USA.

Kroch, A. S.: 1987, Unbounded dependencies and subjacency in a Tree Adjoining Grammar, *in* A. Manaster-Ramer (ed.), *Mathematics of Language*, John Benjamins, Amsterdam, pp. 143–172.

Kroch, A. S. and Joshi, A. K.: 1987, Analyzing extraposition in a tree adjoining grammar, *in* G. J. Huck and A. E. Ojeda (eds), *Syntax and Semantics: Discontinuous Constituency*, Academic Press, Inc., pp. 107–149.

Kroch, T. and Rambow, O.: 1994, The linguistic relevance of quasi-trees, *3e Colloque International sur les Grammaires d'Arbres Adjoints (TAG+3)*, Paris.

Landman, F.: 1991, *Structures for Semantics*, Studies in Linguistics and Philosophy, Kluwer, Dordrecht.

Lehmann, C.: 1993, Relativsätze, *in* J. Jacobs, A. von Stechow, W. Sternefeld and T. Vennemann (eds), *Syntax. Ein internationales Handbuch zeitgenössicher Forschung. 1. Halbband*, Walter de Gruyter, pp. 1199–1216.

Marcus, M. P., Hindle, D. and Fleck, M. M.: 1983, D-theory: Talking about talking about trees, *Proceedings of ACL*.

Meyer Viol, W. and Kempson, R.: 1997, Parsing as Tree Construction in LDS, *Proceedings of Formal Grammar*, Aix-en-Provence, pp. 147–157.

Michaelis, J. and Kracht, M.: 1996, Semilinearity as a Syntactic Invariant, *Logical Aspects of Computational Linguistics*, Nancy.

Montague, R.: 1974, The Proper Treatment of Quantification in Ordinary English, *in* R. H. Thomason (ed.), *Formal Philosophy: Selected Papers of Richard Montague*, Yale University Press, New Haven, pp. 247–270.

Morrill, G. V.: 1994, *Type Logical Grammar. Categorial Logic of Signs*, Kluwer, Dordrecht.

Muskens, R.: 1995, Order-independence and underspecification, *in* J. Groenendijk (ed.), *Ellipsis, Underspecification, Events and More in Dynamic Semantics*, DYANA Report R2.2.C.

Muskens, R.: 1996, Combining Montague Semantics and Discourse Representation, *Linguistics and Philosophy* **19**, 143–186.

Niehren, J., Pinkal, M. and Ruhrberg, P.: 1997a, On Equality Up-to Constraints over Finite Trees, Context Unification, and One-Step Rewriting, *Proceedings of the International Conference on Automated Deduction*, Lecture Notes in Computer Science.

Niehren, J., Pinkal, M. and Ruhrberg, P.: 1997b, A Uniform Approach to Underspecification and Parallelism, *Proceedings of ACL*, Madrid.

Pafel, J.: 1991, Zum relativen Quantorenskopus im Deutschen, *Arbeitspapiere des SFB 340 5*, Universität Tübingen.

Pafel, J.: 1993, Scope and Word Order, *in* J. Jacobs, A. von Stechow, W. Sternefeld and T. Vennemann (eds), *Syntax. Ein internationeles Handbuch zeitgenössischer Forschung*, Vol. 1, Walter de Gruyter, Berlin New York, pp. 867–880.

Parikh, R.: 1966, On context-free languages, *Jounal of the ACM* **13**, 570–581.

Pinkal, M.: 1995, Radical underspecification, *in* P. Dekker and M. Stokhof (eds), *Proceedings of the 10th Amsterdam Colloquium*, pp. 587–606.

Poesio, M.: 1996, Semantic ambiguity and perceived ambiguity, *in* K. van Deemter and S. Peters (eds), *Semantic Ambiguity and Underspecification*, CLSI Publications Stanford, pp. 159–201.

Pollard, C. and Sag, I. A.: 1994, *Head-Driven Phrase Structure Grammar*, Studies in Contemporary Linguistics, The University of Chicago Press, Chicago, London.

Prigent, G.: 1994, Synchronous TAGs and Machine Translation, *3e Colloque International sur les Grammaires d'Arbres Adjoints (TAG+3)*, Paris.

Prior, A. N.: 1968, *Papers on Time and Tense*, Oxford University Press.

Pulman, S. G.: 1994, A Computational Theory of Context Dependence, *in* H. Bunt, R. Muskens and G. Rentier (eds), *Proceedings of the International Workshop on Computational Semantics*, Tilburg.

Quine, W. V. O.: 1960, *Word and Object*, MIT Press.

Rambow, O.: 1994a, *Formal and Computational Aspects of Natural Language Syntax*, PhD thesis, University of Pennsylvania.

Rambow, O.: 1994b, Multiset-Valued Linear Index Grammars: Imposing dominance constraints on derivations, *Proceedings of ACL*.

Rambow, O. and Lee, Y.-S.: 1994, Word order variation and Tree-Adjoining Grammars, *Computational Intelligence* **10**(4), 386–400.

Rambow, O. and Satta, G.: 1996, Synchronous models of language, *Proceedings of ACL*.

Rambow, O., Vijay-Shanker, K. and Weir, D.: 1995, D-Tree Grammars, *Proceedings of ACL*.

Resnik, P.: 1992, Probabilistic Tree Adjoining Grammars as a framework for statistical natural languae processing, *Proceedings of COLING*.

Reyle, U.: 1993, Dealing with ambiguities by underspecification: Construction, representation and deduction, *Journal of Semantics* **10**, 123–179.

Reyle, U.: 1995, On reasoning with ambiguities, *Proceedings of EACL*.

Richter, F. and Sailer, M.: 1996, Syntax für eine unterspezifizierte Semantik: PP-Anbindung in einem deutschen HPSG-Fragment, *in* S. Mehl, A. Mertens and M. Schlulz (eds), *KONVENS-Workshop: Präpositionalsemantik und PP-Anbindung*, Schriftenreihe Informatik SI-16, Gerhard-Mercator-Universität-GH Duisburg, Insitut für Informatik, pp. 39–47.

Richter, F. and Sailer, M.: 1997, Underspecified Semantics in HPSG, *Second International Workshop on Computational Semantics*, Tilburg.

Rodman, R.: 1976, Scope phenomena, "movement transformations", and relative clauses, *in* B. H. Partee (ed.), *Montague Grammar*, Academic Press, pp. 165–176.

Rogers, J.: 1994, *Studies in the Logic of Trees with Applications to Grammar Formalisms*, PhD thesis, University of Delaware.

Rogers, J.: 1996, On Descriptive Complexity, Language Complexity and GB, *in* P. Blackburn and M. de Rijke (eds), *Specifying Syntactic Structures*, CSLI Publications.

Rogers, J. and Vijay-Shanker, K.: 1992, Reasoning with descriptions of trees, *Proceedings of ACL*.

Rogers, J. and Vijay-Shanker, K.: 1994, Obtaining trees from their descriptions: an application to Tree-Adjoining Grammars, *Computational Intelligence* **10**(4), 401–421.

Ross, J. R.: 1968, Constraints on variables in syntax, *Linguistics Club: Aufsatzsammlung*, Indiana University.

Savitch, W. J., Bach, E., Marxh, W. and Safran-Naveh, G. (eds): 1987, *The Formal Complexity of Natural Language*, Studies in Linguistics and Philosophy, Reidel, Dordrecht, Holland.

Schabes, Y.: 1990, *Mathematical and Computational Aspects of Lexicalized Grammars*, PhD thesis, University of Pennsylvania.

Schilder, F.: 1997a, *Temporal Relations in English and German Narrative Discourse*, PhD thesis, University of Edinburgh.

Schilder, F.: 1997b, Towards a Theory of Discourse Processing. Flashback Sequences Described by D-Trees, *Proceedings of Formal Grammar*, Aix-en-Provence, pp. 189–198.

Shieber, S. M.: 1985, Evidence against the context-freeness of natural language, *Linguistics and Philosophy* **8**, 333–343. Reprinted in Savitch et al. (1987).

Shieber, S. M.: 1986, *An introduction to unification-based approaches to grammar*, CSLI.

Shieber, S. M.: 1992, *Constraint-Based Grammar Formalisms*, MIT Press.

Shieber, S. M.: 1994, Restricting the weak-generative capacity of synchronous Tree-Adjoining Grammars, *Computational Intelligence* **10**(4), 271–385.

Shieber, S. M. and Schabes, Y.: 1990, Synchronous Tree-Adjoining Grammars, *Proceedings of COLING*, pp. 253–258.

Sueur, J.-P.: 1977, Quantificateurs et modalités, *Langages* **48**, 84–99.

Vijay-Shanker, K.: 1987, *A Study of Tree Adjoining Grammars*, PhD thesis, University of Pennsylvania.

Vijay-Shanker, K.: 1992, Using descriptions of trees in a tree adjoining grammar, *Computational Linguistics* **18**(4), 481–517.

Vijay-Shanker, K. and Joshi, A. K.: 1985, Some computational properties of Tree Adjoining Grammars, *Proceedings of ACL*.

Vijay-Shanker, K. and Joshi, A. K.: 1988, Feature structures based tree adjoining grammar, *Proceedings of COLING*, Budapest.

Vijay-Shanker, K. and Schabes, Y.: 1992, Structure sharing in lexicalized tree adjoining grammars, *Proceedings of COLING*, Nantes.

Vijay-Shanker, K., Weir, D. J. and Joshi, A. K.: 1987, Characterizing structural descriptions produced by various grammatical formalisms, *Proceedings of ACL*, Stanford.

Vijay-Shanker, K., Weir, D. and Rambow, O.: 1995, Parsing D-Tree-Grammars, *Fourth International Workshop on Parsing Technologies*, Prague.

von Stechow, A. and Nohl, C.: 1995, Interpretation syntaktischer Stukturen. Eine Semantikeinführung anhand des Deutschen, *SfS-Report 07-95*, Seminar für Sprachwissenschaft, Universität Tübingen, Tübingen.

Weir, D. J.: 1988, *Characterizing mildly context-sensitive grammar formalisms*, PhD thesis, University of Pennsylvania.

Table of notation

The following table lists symbols that are used in the thesis without being explicitely defined. Symbols that are explicitely introduced figure in the index.

Symbol	Explanation
$\square$	end of a proof
$\in$	membership sign
$\{x \mid Y\}$	set of all x such that Y holds
$\subseteq$	inclusion
$\subset$	proper inclusion
$\cup$, $\bigcup$	union
$\cap$, $\bigcap$	intersection
$\setminus$	set difference
$A \times B$	set of all pairs $\langle a, b \rangle$ with $a \in A$ and $b \in B$
$Pow(A)$	powerset of A, i.e. set of all subsets of A
A^B	set of all mappings from B to A
$\mathbb{N}$	set of non-negative integers
A^n	($n \in \mathbb{N}$) set of all n-tuples $\langle a_1, \ldots, a_n \rangle$ with $a_i \in A$ for $1 \leq i \leq n$
S_n	($n \in \mathbb{N} \setminus \{0\}$) group of permutations of degree n (i.e. bijections of $\{1, \ldots, n\}$)

Index