Remarks on Linearization

Reflections on the Treatment of LP-Rules in HPSG in a Typed Feature Logic

Magisterarbeit

Universität Tübingen

Frank Richter
Bei der Kirche 2
72074 Tübingen

Manfred Sailer
Martin-Crusius-Str. 10
72076 Tübingen

January 2, 1995

Von Frank Richter wurden die folgenden Teile der Arbeit geschrieben:

Kapitel 1	
Kapitel 2	
Kapitel 3	
Abschnitt 4.3	Außer den Beispielnummern: (4.16), (4.18), (4.22), (4.23), (4.19), (4.21), (4.24), (4.25), (4.26), (4.27), (4.28), (4.30), (4.31), (4.32), (4.33), (4.34), (4.35)

Von Manfred Sailer wurden die folgenden Teile der Arbeit geschrieben:

Abschnitt 4.1	
Abschnitt 4.2	
Abschnitt 4.4	
Abschnitt 4.3.	Nur die Beispielnummern: (4.16), (4.18), (4.22), (4.23), (4.19), (4.21), (4.24), (4.25), (4.26), (4.27), (4.28), (4.30), (4.31), (4.32), (4.33), (4.34), (4.35)
Abschnitt 4.5	
Abschnitt 4.6	
Abschnitt 4.7	
Kapitel 5	
Kapitel 6	

Hiermit versichere ich, Frank Richter, die oben unter meinem Namen angegebenen Teile der Magisterarbeit selbstständig verfaßt und keine anderen Hilfsmittel als die angegebenen Quellen benutzt zu haben.

Hiermit versichere ich, Manfred Sailer, die oben unter meinem Namen angegebenen Teile der Magisterarbeit selbstständig verfaßt und keine anderen Hilfsmittel als die angegebenen Quellen benutzt zu haben.

Contents

1 Introduction

The goal of this thesis is to present a logical architecture for treating word order phenomena in natural languages in the grammatical formalism of Head-Driven Phrase Structure Grammar (HPSG). The existence of a component of the grammar that deals with word order is crucial for the construction of a realistic fragment of a natural language. Therefore the lack of a sufficiently general treatment of word order in HPSG is a substantial problem for linguists working within this grammatical framework. In order to change this situation, two questions have to be addressed. First, it has to be investigated how a component handling word order phenomena can be included into the current theory of HPSG as presented in (Pollard and Sag 1994). And second, once this has been determined the sort hierarchy has to be amended in a suitable way and some tools have to be provided that enable the linguist to easily write linear precedence rules for word order in HPSG. The grammar component for linear order should be flexible enough to allow the formalization of any kind of linear precedence rule a linguist might want to express.

Word order phenomena were subject to extensive research in both empirical work and theoretical work on grammar formalisms such as Generalized Phrase Structure Grammar (GPSG) and the line of research that is usually referred to as Government and Binding theory (GB). This has led to a considerable number of statements or rules about the linear order of words or constituents in natural languages, which we will generally call linear precedence rules (LP rules) for the moment. Ideally, it should be possible to implement generalizations about constituent order that were formulated for other grammatical frameworks in HPSG. One can only hope to achieve this goal by providing a setup that allows to proceed from an informal rule to a rigorous formalization in a fairly straightforward manner. On the other hand, the LP component has to be flexible enough to cope with various sorts of syntactic, semantic and pragmatic information that influence word order.

It is one of our major concerns in the present thesis to show how a setup for LP rules in HPSG can integrate a thorough formalization and the possibility to extend the empirical scope of the LP rules beyond the range of data we are going to cover. A flexible setup that can be augmented in a modular fashion gains special importance for us because for reasons of space we can not go into an empirical discussion but strictly confine ourselves to an examination of the technical questions of an LP component of HPSG.

Thus the notion of modularity is essential for our approach to word order. We believe that not only should it be possible to formulate the LP rules themselves in a theory-independent way that makes them applicable in diverse frameworks of formal grammar. Moreover, their very formalization in HPSG should vary with the specific linguistic analysis as little as, for example, the Head Feature Principle. This is to say that a change in the analysis of the German mittelfeld by substituting a binary branching structure for a flat structure should not entail a reformalization of the LP rules that govern the German mittelfeld. Such an approach is especially desirable for the development of large fragments of grammar for some language where the linguist should be free to focus on the specific analytical problems at hand without having to worry about the proper functioning of other parts of the grammar. In order to guarantee the independence of the formalized LP rules, it is necessary to define the information that has to be provided for them as input. These interfaces are treated in separate parts of the grammar, and they are subject to modification if a change in the syntactic or semantic analysis affects the information that has to be provided as input for the LP rules.

In general, our theory of word order in HPSG consists of three parts. One of them is concerned with the definition of domains to which specific LP rules are applied. For expository purposes we choose the German mittelfeld as the domain for the LP rules we are going to formalize. As we will see, the construction of the word order domain can be subject to change if the syntactic analysis is altered. The second module of the theory of LP consists of a formalization of the LP rules, which we call LP principles since their formal shape exactly resembles the shape of every other grammatical principle in HPSG. The LP principles are usually not subject to change if parts of the syntactic analysis are modified. This is made sure by the third module, which consists of a component of grammar which constructs the information which is necessary for the interfaces to the LP principles but is not directly provided from outside of the LP component of the grammar.

In addition to these three components we assume a universal constraint on the construction of the phonological string of a phrase in natural languages. It says that the phonological string of a phrase is always some permutation of the phonological strings of that phrase's daughters. It is important to note that this universal constraint does not claim that the phonological string of a phrase is some permutation of the *immediate* daughters of that phrase. Hence every daughter of a certain phrase within some suitably defined domain can be affected. What is excluded is the appearance of phonological material in a phrase that does not come from any one of its daughters.

It is well-known that some phenomena of linear order in natural languages are very strict whereas others exhibit degrees of grammaticality. To be more specific, prepositions always have to precede the NP they belong to and inflected adjectives always come after the determiner but before the head noun in German. The situation is different for adverbials, nominal phrases and prepositional phrases in the mittelfeld. The notion of mittelfeld refers to the part of a sentence between the verb second position and the verbal complex (c.f. (Höhle 1986)). Constituents in the mittelfeld can be scrambled, which means they do not show a rigidly fixed constituent order. In addition, different orderings may exhibit varying degrees of grammaticality. Some orderings are completely acceptable, some are less acceptable and others are impossible. A great number of factors play a role in scrambling, and they also influence the degree of of grammaticality of a sentence in a given context. The judgment on a sentence usually varies across speakers and sometimes even the judgments of one and the same speaker may change. This state of things poses of course a vexing problem for linguists. The weakness of these data and their contrast to strict ordering requirements in other empirical domains should be taken into account in a formulation of a theory of word order. However, a discussion of the ontological implications of degrees of grammaticality for the logical formalism underlying HPSG and attempts on their solution go far beyond the scope of this thesis. Therefore we will ignore this topic throughout and leave it to further research. Our own treatment of word order starts from the simplistic assumption that sentences are either grammatical or ungrammatical.

In our treatment of word order the attribute PHON, which is defined for signs, plays an important role. This should not lead to the assumption that we have anything to say about the phonological component of the grammar. For the sake of simplicity we abstract away from an actual phonological analysis. As a consequence, the value of PHON is simply a list of words in an orthographical representation. Of course, this leaves open the possibility to add further information, e.g. about the boundaries of constituents, in a more realistic approach. We assume that our simplifying treatment of the phonological component does not in any way impede a modular augmentation of the grammar with a well-founded phonological theory.

The treatment of LP rules that we are going to propose does not introduce formal devices that are not part of HPSG in the version of (Pollard and Sag 1994). All parts of our theory of LP outlined above will be formalized by just extending the sort hierarchy, introducing new appropriateness conditions and writing implicational constraints. Throughout this thesis we presuppose some familiarity with (Pollard and Sag 1994) and do not present the sort hierarchy that Pollard and Sag establish. Assumptions on the analysis of German that go beyond the sort hierarchy and the constraints of (Pollard and Sag 1994) will be discussed as it becomes necessary.

1.1 Motivation

As we have already mentioned in the beginning, there is no generally accepted way of treating word order phenomena in the current HPSG literature. This might partly be due to the fact that issues of linear order have been of minor concern to Pollard and Sag in their standard volumes on HPSG (Pollard and Sag 1987, Pollard and Sag 1994). Although (Pollard and Sag 1987, pp.169-190) contains an informal chapter on principles of constituent order, the authors do not elaborate on their sketchy remarks in (Pollard and Sag 1994) but ignore this topic completely.

Since an analysis of linear order is a prerequisite for the practical implementation of natural language fragments, it was a necessity for computational linguists to adopt a working solution for natural language parsing and generation in HPSG. Thus systems like ALE (Carpenter 1993) and Troll (Gerdemann 1993) introduce a phrase-structural backbone to HPSG that allows ID/LP parsing as it was proposed first by Shieber (1984) and subsequently refined for efficiency and the treatment of unification-based ID/LP grammars.[1] One further reason for adopting this solution was the availability of efficient parsing algorithms for ID/LP parsing, while no efficient techniques for a purely constraint based grammar were known.

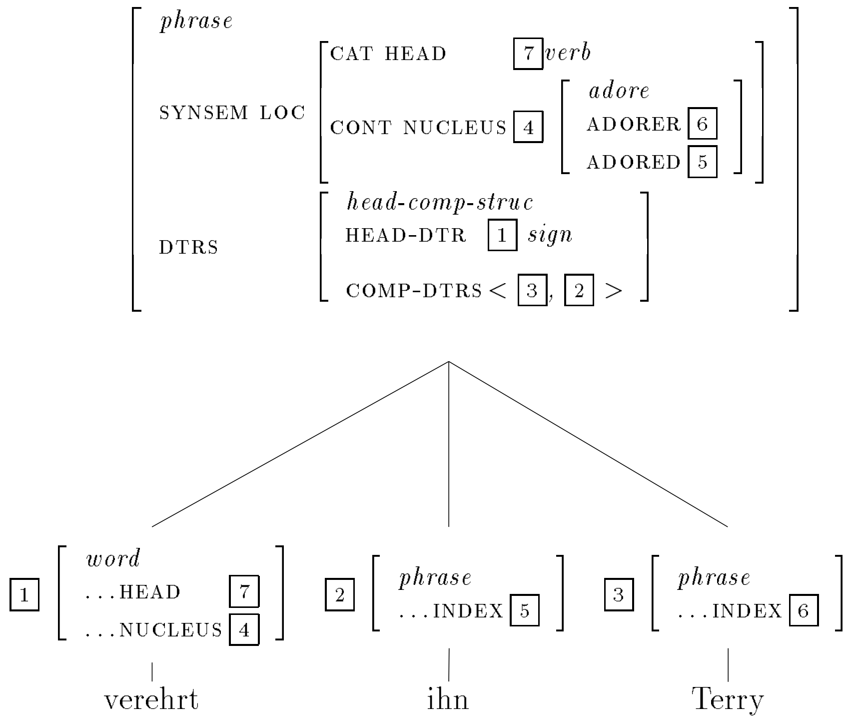
For an HPSG formalism with a phrase-structure backbone it is necessary to reinterpret a suitable set of constraints as ID rules because the concept of phrase structure rules does not belong to the classical HPSG formalisms that are envisioned in (Pollard and Sag 1987, Pollard and Sag 1994). Although the ontologies of the formalism in these two books are radically distinct in other respects neither of them comprises the notion of phrase structure rules. It is clear from the outset that the addition of phrase structure rules means a departure from the traditional logical setup and introduces new technical devices such as the definition of a typed unification grammar including a set of terminals, nonterminals and unification based phrase structure rules.[2] For obtaining phrase structure rules or ID rules, the ID principles are interpreted as ID rules with additional constraints on the nodes of the tree. As a result, the values of the DTRS attribute of phrases are translated into the daughter nodes in phrase structure trees, thus preserving the intuitive meaning of that part of the descriptions. The following tree illustrates this interpretation of

[1] See (Morawietz 1994, pp.9-14) and (Meurers and Morawietz 1994, pp.5-9) for an overview over the developments in ID/LP parsing. ID means immediate dominance (of nodes in a tree) as opposed to the linear order of nodes of equal depth.

[2] For a complete definition of a typed unification grammar see (Morawietz 1994, p.27). The notion of type as used in (Morawietz 1994) is equivalent to the notion of sort in the present thesis.

the DTRS attribute.

(1.1) *HPSG with a phrase structure backbone*



Most crucially, the reinterpretation of HPSG as a phrase structure grammar involves a different view on how the phonological string of words of a sentence is constructed. Pollard and Sag (1987) define an attribute PHON for signs that contains its phonological string in chronological order. Although they do not formalize principles which constrain the value of PHON, they informally explain that the phonological string associated with each phrase is somehow constructed out of the PHON values of the signs under its DTRS attribute. Sentences of a phrase structure grammar, on the other hand, are defined as the elements of the set of strings (usually called the language of the grammar) that can be derived by rewriting a nonterminal start symbol into a string of terminal symbols of the grammar by applying phrase structure rules. This setup does not necessarily include a PHON attribute for signs. Instead, phonological representations of the words form the terminal symbols of the phrase structure grammar. The phonological representation of the sentence is the result of rewriting the start symbol of the grammar.[3] The fundamental difference between

[3]It should be noted that variants of this setup are possible in which the terminal symbols are feature graphs (of sort word) which actually contain a PHON attribute. As far as we can see, this has no influence on the choices a grammar writer has at her disposal for writing LP statements in the ID/LP format as we will describe it below.

the two approaches lies in the different formal status of the phonological string of a sentence. As a consequence, the means for restricting word order diverge widely.

Before we enter into the discussion of the various forms of LP rules that occur in the literature, let us briefly clarify our terminology. In the following paragraphs we use the term *LP rules* for linear precedence rules in general, independent of the formalism they are written in. With *LP principles* or *LP constraints* we mean such linear precedence rules that are written as implicational constraints in a logical formalism for HPSG as envisioned in (Pollard and Sag 1994). *LP statements* refers to linear precedence rules that are written in the format of LP rules as defined for GPSG style ID/LP grammars. Consequently, the term LP statement used in connection with HPSG means an LP rule for an HPSG that is defined as an ID/LP grammar. LP statements for HPSG are applied to the phrase structure backbone of the grammar.

Since the ID/LP paradigm is widely used for computational work on HPSG, it gained significant influence on the thinking about word order in HPSG. This is so despite the fact that it is not claimed that the ID/LP interpretation of the formalism is necessarily adequate (c.f. (Engelkamp, Erbach and Uszkoreit 1992, p.201) and (Morawietz 1994, p.3)) The important role this interpretation played in the literature nevertheless justifies a closer look at the meaning of ID/LP and the specific assumptions it purports about generalizations concerning word order.

First of all it should be noted, that the notion of ID/LP format refers to the very specific notion of separation of hierarchical and linear information in a syntactic structure as it is employed in Generalized Phrase Structure Grammar (Gazdar and Pullum 1982, Gazdar, Klein, Pullum and Sag 1985). In principle, the X-bar theory of GB-theory in combination with the postulation of language specific head parameters which place the syntactic head of a projection at a certain linear position and the formulation of rules that govern the ordering of constituents within the projection domain of a syntactic head introduces a similar distinction between hierachical structure and the ordering of words in a phonetic string. What is most frequently meant with the notion of ID/LP, though, is not the general concept of separating generalizations on hierarchical structure and the linear order of the phonology but the specific formal explication of this idea in GPSG.

In GPSG ID rules were introduced as a variant of phrase structure rules that do not by themselves determine the eventual order of the terminal or nonterminal symbols on the right-hand side of the rule. Instead, the order of the sisters in a local tree is determined by the interaction of ID rules and LP statements. For example, if no LP statement concerning any pair of symbols a, b and c is imposed, the ID rule $S \rightarrow a,\ b,\ c$ admits local trees with six possible orderings of these three symbols, namely *abc*, *acb*, *bac*, *bca*, *cab* and *cba*. The classical phrase structure rule $S \rightarrow a\ b\ c$ on the other hand only licenses the single ordering *abc*. In the ID/LP format, constraints on order are imposed by linear precedence statements of the form $a \prec b$, which says that if a local tree is consistent with an ID rule having a and b on its right-hand side, it is only admissible if a precedes b. In actual LP statements a and b are represented by bundles of features, and the LP statement is interpreted as demanding that the sisters in a local tree which contain the respective features be in the required order.[4] The ID rule above together with the LP statement $a \prec b$ only admits three strings of symbols instead of six, namely *abc*, *acb* and *cab*.

This paradigm for treating word order generalizations exhibits three particular properties that are of special interest for us, since they carry over to an ID/LP interpretation of HPSG but not to the HPSG formalism itself as outlined in (Pollard and Sag 1994). Even more importantly, there is a significant difference in what kind of linear precedence rules can be expressed in the two formalisms.

Linear precedence statements in GPSG can only apply in flat structures, i.e., to sisters in a local tree. This restriction is of great impact for the description of the German mittelfeld, since it implies that it has to exhibit a flat syntactic structure. If this was not the case, it would not be possible to capture the linear order of NPs in the mittelfeld directly by LP statements. The tacit assumption that HPSG can or should be treated as a variant of grammars in ID/LP format has led some authors to believe that LP rules for HPSG are subject to the same restrictions as in GPSG. In (Pollard 1990) this assumption is produced as indirect support for Pollard's suggestion to assign a flat structure to an HPSG analysis of the German mittelfeld. As we will show, nothing in the HPSG formalism prevents the formalization of LP principles that do not exclusively refer to the value of the DTRS attribute of a phrase and thus to its closest equivalent of a local tree in phrase structure grammars. The decision whether the mittelfeld has to be assigned a flat or some other structure is left to empirical investigation and not imposed by the fact that LP principles and the ID principle are independent in HPSG.

Another important attribute of LP statements in GPSG is the ECPO property (exhaustive constant partial ordering) which demands that an LP rule be applied in every local tree. This means, whenever the features of an LP rule match the right-hand side of an ID rule the resulting structure has to obey the LP rule, no matter in which context the ID

[4] For a discussion of differing options to define a semantics for LP rules see (Uszkoreit 1986a, pp.14-21). For a detailed explication of the ID/LP format, see (Gazdar et al. 1985, pp.44-50).

rule applies. Hinrichs and Nakazawa (1989, p.143) point out that this condition is weakened by the possibility to explain away apparent violations of ECPO by adding arbitrary distinguishing features to the grammar and making the LP rules sensitive to them. From a theoretical point of view, it is, of course, unsatisfactory to invent new features only to avoid overgeneralizations of LP rules. In HPSG, however, the situation is somewhat different, because an independently motivated sort hierarchy can be used besides attributes to distinguish between syntactic structures. As we will show, implicational constraints for LP principles make use of sorts in the sort hierarchy just like any other principle. The sorts as antecedents in the constraints provide an additional means to restrict the LP rules of HPSG to certain domains, while at the same time they allow for generalizations by the way the sort hierarchy is constructed.

The last point to be addressed here concerns the kind of LP statements that can be expressed in the ID/LP format and is mainly a question of the explanatory adequacy of the formalism. By the way LP statements are defined they exclusively refer to features and feature values in a node. They can not take into account the function a constituent fulfills unless it is encoded in a feature. Now the question is if this restriction fits the needs of an elegant linguistic theory of word order or if there are reasons to abandon it.

Since its appearance the work by Lenerz (1977) has been particularly important for the theory of constituent order in the German mittelfeld. Lenerz himself does not use features for his LP rules since he does not use any formal framework at all. But most of the informal rules he established for the order of nominal arguments of verbs in the mittelfeld have since been described by formulating LP statements with respect to certain features. Among them are features for definiteness, case, pronouns and nonpronouns, topic, focus and thematic roles. Subsequent work by Uszkoreit (1986b, 1986a) in GPSG demonstrated how this approach could be made formally precise. In addition, Uszkoreit even explored the possibilities of including an explicit analysis of degrees of grammaticality.

Höhle (1982) and Reis (1987), however, raise interesting questions about the adequacy of Lenerz's descriptive categories. For our purposes, it is noteworthy that Reis (1987, p.156) mentions the existence of different classes of verbs which impose differing ordering constraints on their nominal arguments in the mittelfeld, the values of the features mentioned above being held equal. Höhle (1982, p.106) shows that the focus potential of a sentence depends on both the internal structure and properties of the nominal arguments and the type of the verb in the sentence. The focus potential is also sensitive to varying linear order of the arguments of a verb in a sentence. Thus constituent order and focus potential interact for a given verb, and together they constrain the set of contexts the sentence can felicitously occur in.

These observations lead to the expectation that in the general case the possible orders of constituents in the mittelfeld do not only depend on properties that are inherent to these constituents. Secondly, only after the verb in a given sentence structure is instantiated by a specific lexical item, the possible orders of the constituents can be determined. Although it is conceivable that such factors can be accounted for by adding appropriate features to the affected nominal constituents, we would prefer a formalism that can handle them in a more direct manner. That is to say, we are aiming at a formalism in which word order regularities can be expressed that do not exclusively refer to inherent properties of the elements that are to be linearized. The ID/LP format of GPSG does not provide a suitable foundation for this strategy.

Engelkamp et al. (1992) go one step beyond a pure ID/LP interpretation of HPSG with respect to LP rules. Instead of writing simple LP statements they formalize two LP principles in the description language of HPSG in a variant of (Pollard and Sag 1987). At the same time, they stick to the assumption of a phrase structure backbone to be able to apply efficient processing algorithms in a computer implementation of the grammar. The main idea of (Engelkamp et al. 1992) is to furnish lexical entries with information as to which properties other constituents to their left or to their right in a syntactic structure must not have. The LP principles then take care of the propagation and application of that information in a sentence. The domain in which the information percolates is limited to so-called head domains, i.e., the projection domain of the syntactic head of the construction. Thus constituents or words that are contained in the projection domain of the same syntactic head can be ordered with respect to each other.[5] As a result Engelkamp et al. (1992) can dispense with the necessity to adopt a flat mittelfeld in German, because all NPs in the mittelfeld are in the head domain of the finite verb. On the negative side, all information about word order has to be encoded in features and must be contained in the lexical entries. As we have argued above, this is a consequence we want to avoid in our own approach to principles of word order.

Besides approaches which treat word order by reinterpreting HPSG as an ID/LP grammar in the sense of GPSG, there is one line of research that takes a different stance. Following up ideas of Reape (1989, 1992), Pollard, Levine and Kasper (1994) outline an architecture for HPSG in which word order can be captured with implicational constraints only. This means, the authors do not employ a phrase structure backbone and there is no need for them to abandon the formal assumptions underlying

[5] For a short summary and discussion of the approach of (Engelkamp et al. 1992) in the context of ID/LP parsing see (Morawietz 1994, pp.6-8).

(Pollard and Sag 1994).

Interesting as a comparison of the ideas of (Pollard et al. 1994) with our conception would be, it can not be done in a fruitful way before both approaches present concrete analyses of a sufficiently broad range of pertinent data. Even more importantly, since (Pollard et al. 1994) is only a research proposal and not a fully developed theory, it is not yet possible to see how they want to formalize their intuitions. At this point, we just want to point out two major differences between our conception and the one sketched by Pollard et al. (1994) that highlight the differences in the two strategies.

Whereas it is our goal to stick as closely as possible to the sort hierarchy and appropriateness conditions of (Pollard and Sag 1994) and to augment them in a modular fashion with an additional LP component, Pollard et al. (1994) propose a significant change of the appropriateness conditions for signs. They eliminate both the SYNSEM and the PHON attribute and replace them by the new attribute NODE, whose value *node* is specified for PHON and SYN.

(1.2) *The structure of signs in (Pollard et al. 1994)*

$$\begin{bmatrix} \textit{sign} & \\ \textsc{node} & \begin{bmatrix} \textit{node} & \\ \textsc{phon} & \textit{list-of-phons} \\ \textsc{syn} & \textit{syn} \end{bmatrix} \\ \textsc{dom} & \textit{list-of-nodes} \\ \textsc{dtrs} & \textit{const-struc} \end{bmatrix}$$

As a consequence, the principles of (Pollard and Sag 1994), none of which has anything to do with word order, have to be rewritten to fit the new architecture. During further development of the theory it might again become necessary to adjust one part of the theory where some change in the other is desired. In contrast, it is our goal to keep the two components as independent as possible and to define clear interfaces.

The approach to formalizing the LP principles differs even more remarkably. Pollard et al. (1994), following (Reape 1992), make use of the symbol $\prec$, which is supposed to ensure a certain precedence relation of two elements on a list, namely the DOM list. In case they intend this to be more than a notational abbreviation for some description, they have to expand the underlying logic by a syntax and presumably rather complex semantics of that symbol.[6] We take a more conservative position

[6] This is the approach taken by Reape (1989), who formalizes the symbol $\preceq$ in his logic (Reape 1989, pp.104-106). Since Pollard et al. (1994) closely follow Reape in their assumptions on the formalization of LP principles, it is likely that they have something very similar in mind.

and instead of changing the logic to handle constraints that are specific for word order, we use the well-established logic of (King 1989) and its extension in (King 1994) to formalize both the old constraints of (Pollard and Sag 1994) and new principles of the LP component of the grammar.

1.2 The Logic

Pollard and Sag (1994) argue that a linguistic theory such as HPSG should be formalized. However, in the course of their discussion they do not commit themselves as to which particular formalization of HPSG they assume. Instead they describe some properties of the formalism they envision. Therefore there exists no standard formalization of (Pollard and Sag 1994) and the reader is left to choose a formalism she prefers. Since we consider it a major advantage of HPSG that precise formalisms are available for the linguist, we put special emphasis on an exact formalization of our theory of word order. Only a thoroughly formalized variant of HPSG provides a sufficiently explicit formal language which can be used to prove the correctness of the proposed constraints mathematically. This is in turn a prerequisite for the establishment of a truly modularly constructed grammar, which has to rely on precisely defined interfaces between its components.

For the definition and proof of the LP principles we use a variant of the logic of the K-formalism of (King 1994). The logic which is part of the K-formalism is (modulo some terminological changes) identical with SRL (speciate re-entrant logic), the logic of the classical formal language for HPSG developed in (King 1989). King (1994) basically uses SRL to define the notions of type, token, their relationship, and what it is for a theory to admit a type. The definitions of type and token are meant to capture the intuitions that guide the informal discussion of the type-token distinction of Pollard and Sag (1994, p.14). Roughly speaking, a linguistic token is a concrete utterance of a sentence at a specific place and time, while a linguistic type is an abstract linguistic entity that corresponds to a concrete utterance of a sentence and is subject to theoretical interest. King (1994) then proceeds to build a mathematical model for the K-formalism and shows that the mathematical entities of his model correspond to the feature structures of (Pollard and Sag 1994). Thus we can be certain that the K-formalism meets all fundamental requirements Pollard and Sag (1994) envisage for a thorough formalization of HPSG.

Since we are not concerned with the formalism itself, we will not discuss the K-formalism and its relation to (Pollard and Sag 1994). Readers interested in technical details of formalization are referred to the mathematical literature (King 1989, King 1992, King 1994, Kepser 1994). Meurers (1994, pp.50-55) provides an overview over alternative logical setups and discusses some of the consequences of different ontological assumptions. What we will need, though, is the logic of the K-formalism that can guarantee us the logical soundness of our principles. Hence we will repeat the definitions of (King 1994) with slight modifications, introduce some shorthand notation that will be useful for better readability of the proofs and comment on some terminological idiosyncrasies of the logic. In our presentation we basically follow (King 1994, pp.8-10). We differ from (King 1994) only in the additional sort hierarchy we build into the signature and the interpretation.

From the linguist's point of view, an HPSG consists of a signature comprising a sort hierarchy and appropriateness conditions, sometimes called partitions and feature declarations, and of a set of principles, which are written as implicational constraints and conjunctively make up the theory of a natural language fragment. It should be noted at this point that the lexicon is expressed by an implicational constraint which is known as the Word Principle (c.f. (Höhle 1994, p.2) and (Pollard 1993, p.3)). More precisely, the Word Principle states that every word has to be described by one of the lexical entries LE_i in the constraint $word \rightarrow (\mathrm{LE}_1 \vee \ldots \vee \mathrm{LE}_n)$, $1 \leq i \leq n$, where the descriptions LE_1 to LE_n make up what is traditionally called the lexicon.

The linguist wants to use these tools to describe objects, namely well-formed objects of the linguistic universe. Furthermore, she wants to know if some given description is true or false of an object and if an object in the linguistic universe is described by a description or not. For this purpose a logic of description is needed which provides the means for describing objects and the objects that are described. The signature supplies the syntactic means for the construction of descriptions.

1 Definition. *Σ is a signature iff*

Σ is a quintuple $\langle \mathfrak{G}, \sqsubseteq, \mathfrak{S}, \mathfrak{A}, \mathfrak{F} \rangle$,

$\langle \mathfrak{G}, \sqsubseteq \rangle$ is a finite partial order,

$$\mathfrak{S} = \left\{ \sigma \in \mathfrak{G} \,\middle|\, \begin{array}{l} \textit{for each } \sigma' \in \mathfrak{G}, \\ \textit{if } \sigma' \sqsubseteq \sigma \textit{ then } \sigma = \sigma' \end{array} \right\},$$

$\mathfrak{A}$ is a set, and

$\mathfrak{F}$ is a partial function from the Cartesian product of $\mathfrak{G}$ and $\mathfrak{A}$ to $\mathfrak{G}$, and

for each $\sigma_1 \in \mathfrak{G}$, for each $\sigma_2 \in \mathfrak{G}$ and for each $\alpha \in \mathfrak{A}$,

if $\mathfrak{F}\langle \sigma_1, \alpha \rangle$ is defined and $\sigma_2 \sqsubseteq \sigma_1$
then $\mathfrak{F}\langle \sigma_2, \alpha \rangle$ is defined and $\mathfrak{F}\langle \sigma_2, \alpha \rangle \sqsubseteq \mathfrak{F}\langle \sigma_1, \alpha \rangle$,

for each $\sigma \in \mathfrak{S}$ *and for each* $\alpha \in \mathfrak{A}$,

if for each $\sigma' \sqsubseteq \sigma$, $\mathfrak{F}\langle\sigma', \alpha\rangle$ *is defined,*
then $\mathfrak{F}\langle\sigma, \alpha\rangle$ *is defined.*

We call each member of $\mathfrak{S}$ a sort, each member of the subset $\mathfrak{S}$ of $\mathfrak{S}$ a species, each member of $\mathfrak{A}$ an attribute and $\mathfrak{F}$ an appropriateness function. Sorts are sometimes called types and species are occasionally called varieties in the literature. The symbols :, $\sim$, $\approx$, $\neg$, [,], $\wedge$, $\vee$ and $\rightarrow$ are reserved for descriptions, and we assume that none of them is a sort, a species or an attribute.

Clearly, $\mathfrak{F}$ provides what is known to the linguist as the appropriateness conditions (or feature declarations) whereas the partial order $\langle\mathfrak{S}, \sqsubseteq\rangle$ provides the sort hierarchy (or partition). In accordance with Pollard and Sag (1994, p.395) we make two assumptions about the sort hierarchy in our definition of the signature. It is a finite partial order and a sort is maximal (a species) iff it has no proper subsorts. Furthermore, we can safely assume that a sort is root iff the sort is supersort of each sort and the sort partial order has a root sort. With the first condition on the appropriateness functions we meet the requirement by Pollard and Sag (1994) that a sort inherit the attributes of its supersorts.[7] The second condition we introduce is in fact of no consequence for the logical setup. It says that if an attribute is defined for each subsort of a certain sort, then it is also defined for that sort. It forces the linguist to make all attributes a sort may have explicit in the appropriateness conditions, which we believe improves readability.

2 Definition. *I is an interpretation iff*

I *is a triple* $\langle U, S, A\rangle$,

U *is a set of objects,*

S *is a total function from* U *to* $\mathfrak{S}$ *called species assignment function,*

A *is a total function from* $\mathfrak{A}$ *to the set of partial functions from* U *to* U *called attribute interpretation function, and*

for each $\alpha \in \mathfrak{A}$ *and each* $u \in U$,

if $A(\alpha)(u)$ *is defined*
then $\mathfrak{F}\langle S(u), \alpha\rangle$ *is defined, and* $S(A(\alpha)(u)) \sqsubseteq \mathfrak{F}\langle S(u), \alpha\rangle$, *and*

[7]This condition corresponds to Carpenter's upward closure condition on appropriateness specifications (Carpenter 1992, p.86). It should be noted, though, that our sort hierarchy is reversed. Species are at the bottom of the sort hierarchy, whereas Carpenter's most specific types are at the top. This is of no consequence for an HPSG grammar.

for each $\alpha \in \mathfrak{A}$ *and each* $u \in U$,

if $\mathfrak{F}\langle S(u), \alpha\rangle$ *is defined then* $A(\alpha)(u)$ *is defined.*

U is the set of objects in the linguistic universe. The species assignment function S makes sure that each object in the universe is labeled by a species. As $\langle \mathfrak{G}, \sqsubseteq\rangle$ is a finite partial order, the set of species is finite. Therefore the objects of the universe are exhaustively partitioned into a finite number of categories. With the definition of S we adopt a closed world interpretation, that is, every object in the denotation of a non-maximal sort is described by a species, and every object is of exactly one species.

The attribute interpretation function A gives each attribute a meaning by assigning it a partial function from U to U. The two conditions at the end of the definition enforce the correlation between species and attributes defined by the signature in the interpretation. The first condition requires that if an attribute is defined on an object u then the object u' that is related to u by attribute α must be of a species as defined by the signature Σ, i.e., $S(u') \sqsubseteq \mathfrak{F}\langle S(u), \alpha\rangle$. Conversely, the second condition says that if an object u is of a species an attribute α is appropriate for then the partial function modeling α in the interpretation I must actually be defined on u. In short, each attribute names a unary partial function from objects to objects that is defined on all or no objects in the denotation of each maximal sort.

The definition of an interpretation meets two more conditions that Pollard and Sag (1994, p.395) establish. The root sort denotes the set of all linguistic entities in the interpretation and each sort denotes a set of objects in the interpretation.

King (1989, pp.47-48) points out that a signature and interpretation comprising a partial order on sorts with the conditions we introduce can readily be borne out by the simpler definition of a signature and interpretation that SRL employs. Hence all formal properties of the logic in (King 1994) apply to the logic we use. There are no formal reasons for the modifications we made. They simply serve to provide a formal notation that is as close as possible to the standard setup of HPSG.

For the notation of objects in the linguistic universe we follow some abbreviatory conventions. We write $<>$ for those $u \in U$ such that S(u) = *elist*. For any sort $\sigma \in \mathfrak{G}$ and $u \in U$ such that S(u) = *nelist-of-*σ*s* of length n we sometimes write $< u_1, \ldots, u_n >$, where S(u_i) $\sqsubseteq \sigma$, $1 \leq i \leq n$, and u_i is the i-th element on list u. This notation for objects in the interpretation should not be confused with the syntactically similar list notation that is used in AVMs. The latter belongs to the description language.

Occasionally we will employ a partial function `length` which is also implicitly meant when we informally speak of the length of a list in the interpretation. `length` is a partial function from U to the natural numbers including 0 which indicates the length of lists. Therefore we define $\texttt{length}(<>) = 0$ and $\texttt{length}(< u_1, \ldots, u_{n-1}, u_n >) = \texttt{length}(< u_1, \ldots, u_{n-1} >) + 1$, with $\mathrm{S}(u_i) \sqsubseteq \sigma$, $1 \leq i \leq n$, for some sort $\sigma \in \mathfrak{S}$. For any other $u \in U$ `length` is undefined.

3 Definition. *τ is a term iff τ is a member of the smallest set $\mathfrak{T}$ such that*

$: \in \mathfrak{T}$, and

for each $\alpha \in \mathfrak{A}$ and each $\tau \in \mathfrak{T}$, $\tau\alpha \in \mathfrak{T}$.

A term consists of the symbol : designating the root followed by a (possibly empty) string of attributes. The notation is defined from left to right. Thus it corresponds to the notation which is ordinarily used in attribute value matrices (AVMs). This deviation from the standard logical notation enhances the readability for linguists.

4 Definition. *T is a term interpretation function over I iff*

I is an interpretation $\langle U, S, A\rangle$,

T is a total function from $\mathfrak{T}$ to the set of partial functions from U to U,

$T(:)$ is the identity function from U to U, and

for each $\alpha \in \mathfrak{A}$ and each $\tau \in \mathfrak{T}$,

$$T(\tau\alpha) \text{ is the functional composition of } T(\tau) \text{ and } A(\alpha).$$

According to the definition, terms are interpreted as partial functions. Since terms consist of a rooted sequence of attributes which are interpreted as partial functions, it is natural to interpret terms as a composition of the partial functions the attributes name. For each interpretation I there is a unique term interpretation function T_I over I. Since we are working with a fixed interpretation throughout the rest of the thesis, we will ignore this in later chapters and simply write T for the term interpretation function. Furthermore, we introduce some abbreviatory conventions to enhance the readability of the term interpretation function in proofs. For $T(\tau)(u)$ we simply write $[\tau]u$, and for convenience we also omit the root symbol : in the term notation of the proofs.

5 Definition. *δ is a description iff δ is a member of the smallest set $\mathfrak{D}$ such that* *for each* $\sigma \in \mathfrak{S}$ *and each* $\tau \in \mathfrak{T}$, $\tau{\sim}\sigma \in \mathfrak{D}$,

for each $\tau_1 \in \mathfrak{T}$ *and each* $\tau_2 \in \mathfrak{T}$, $\tau_1{\approx}\tau_2 \in \mathfrak{D}$,

for each $\delta \in \mathfrak{D}$, $\neg\delta \in \mathfrak{D}$,

for each $\delta_1 \in \mathfrak{D}$ *and each* $\delta_2 \in \mathfrak{D}$, $[\delta_1 \wedge \delta_2] \in \mathfrak{D}$,

for each $\delta_1 \in \mathfrak{D}$ *and each* $\delta_2 \in \mathfrak{D}$, $[\delta_1 \vee \delta_2] \in \mathfrak{D}$, *and*

for each $\delta_1 \in \mathfrak{D}$ *and each* $\delta_2 \in \mathfrak{D}$, $[\delta_1 \rightarrow \delta_2] \in \mathfrak{D}$.

$\mathfrak{D}$ is called the set of descriptions. Descriptions are that part of the logic that is probably most familiar to linguists, since they correspond to the AVM notation. In fact, King (1989, pp.116-118) proposes a translation from AVMs to term notation.

The linguist uses descriptions to write constraints like the Head Feature Principle, the ID Principle, the Semantics Principle and the lexical entries. As the definition shows, two basic statements are provided. With $\tau{\sim}\sigma$ it can be said that a certain path receives a specific sort, and the term equation $\tau_1{\approx}\tau_2$ corresponds to the notion of structure sharing. In addition, descriptions are supplied with the standard connectives of logic. The ID principle of Pollard and Sag (1994, p.38), for example, is a disjunction of six ID schemata[8] and the principles mentioned above are connected by conjunction.

All principles that make up the LP component of the grammar will likewise be written as descriptions.

6 Definition. *D is a description denotation function over I iff*

I is an interpretation $\langle U, S, A\rangle$,

D is a total function from $\mathfrak{D}$ *to the powerset of* U,

for each $\sigma \in \mathfrak{S}$ *and each* $\tau \in \mathfrak{T}$,

$$D(\tau{\sim}\sigma) = \left\{ u \in U \,\middle|\, \begin{array}{l} T_I(\tau)(u) \textit{ is defined, and} \\ S(T_I(\tau)(u)) \sqsubseteq \sigma \end{array} \right\},$$

for each $\tau_1 \in \mathfrak{T}$ *and each* $\tau_2 \in \mathfrak{T}$,

$$D(\tau_1{\approx}\tau_2) = \left\{ u \in U \,\middle|\, \begin{array}{l} T_I(\tau_1)(u) \textit{ is defined,} \\ T_I(\tau_2)(u) \textit{ is defined, and} \\ T_I(\tau_1)(u) = T_I(\tau_2)(u) \end{array} \right\},$$

[8] In the appendix of the book Pollard and Sag (1994, p.399) make a slightly different assumption which we adopt later. We will shortly comment on the difference in section 3.1.

for each $\delta \in \mathfrak{D}$, $D(\neg\delta) = U \setminus D(\delta)$,

for each $\delta_1 \in \mathfrak{D}$ *and each* $\delta_2 \in \mathfrak{D}$, $D([\delta_1 \wedge \delta_2]) = D(\delta_1) \cap D(\delta_2)$,

for each $\delta_1 \in \mathfrak{D}$ *and each* $\delta_2 \in \mathfrak{D}$, $D([\delta_1 \vee \delta_2]) = D(\delta_1) \cup D(\delta_2)$, *and*

for each $\delta_1 \in \mathfrak{D}$ *and each* $\delta_2 \in \mathfrak{D}$, $D([\delta_1 \rightarrow \delta_2]) = (U \setminus D(\delta_1)) \cup D(\delta_2)$.

As for the term denotation function, there is a unique description denotation function D_I over each interpretation I. Then for some interpretation $I = \langle U, S, A \rangle$, $D_I(\delta)$ is the set of entities in I of which δ is true. Hence $D_I(\tau{\sim}\sigma)$ denotes the set of those objects u in I, such that the term interpretation function $T_I(\tau)$ is defined on u, $T_I(\tau)(u) = u'$, and u' is of a species σ' with $\sigma' \sqsubseteq \sigma$. In case σ is already a species, we obtain $\sigma' = \sigma$. $D_I(\tau_1{\approx}\tau_2)$ denotes the set of those objects, such that the term interpretation function $T_I(\tau_1)$ is defined on u, $T_I(\tau_1)(u) = u_1$, $T_I(\tau_2)$ is defined on u, $T_I(\tau_2)(u) = u_2$, and u_1 and u_2 are identical. $D_I(\neg\delta)$ is the set of objects of which δ is false, which is the set complement of $D_I(\delta)$ against U. $D_I([\delta_1 \wedge \delta_2])$ is the set of those objects of which both δ_1 and δ_2 are true. $D_I([\delta_1 \vee \delta_2])$ is the set of objects of which δ_1 is true or δ_2 is true. And finally, $D([\delta_1 \rightarrow \delta_2])$ is the set of objects of which δ_1 is false or δ_2 is true.

With the definitions we have discussed so far we have provided everything we need for the formalization of LP principles. For the sake of completeness, we add two more definitions which say what a theory is and what it means.

7 Definition. *θ is a theory iff $\theta \subseteq \mathfrak{D}$.*

A theory is simply a set of descriptions which are interpreted conjunctively. An HPSG theory consists therefore of the conjunction of its principles.

8 Definition. *Θ is a theory denotation function over I iff*

I is an interpretation,

Θ is a total function from the powerset of $\mathfrak{D}$ to the powerset of U, and

for each $\theta \subseteq \mathfrak{D}$, $\Theta(\theta) = \bigcap\{D_I(\delta) \mid \delta \in \theta\}$.

There exists a unique theory denotation function Θ_I for each interpretation I. The denotation of a theory θ over I is the set of those entities of which each description in that theory is true.

There are some properties of the logic linguists should be aware of when they use it to write an HPSG theory. Most importantly, the logic alone is not sufficient to enforce the well-formedness of linguistic objects like sentences. The reason for this is that sentences, which are objects of sort *phrase*, contain further phrases. When linguists write principles for phrases they want to express that every phrase in a sentence and not only the sentences themselves meet those principles, and they do not express this explicitly in the description of sentences. But the descriptions of the logic do not say anything about objects that are embedded in the described object. Thus for the recursive application of the principles of grammar to each object within a larger object some extra measures have to be taken. In (King 1994, p.12) the proper treatment of these cases is ensured by defining type admission recursively in the K-formalism. Therefore only the entire K-formalism guarantees that all principles exert the restrictions the linguist aims at and, for example, every phrase within a sentence meets the Head Feature Principle.

A related issue is the question of grammaticality. Kepser (1994) proves that King's logic is decidable. This does not entail, though, that given a theory it can be determined by a computable function whether an arbitrary description of a sentence is grammatical according to the theory.[9] The latter question is a problem of the decidability of grammaticality. Decidability of the logic means only that there exists a computable function over the set of descriptions $\mathfrak{D}$ that determines for each $\delta \in \mathfrak{D}$ whether the empty set entails δ or not. Hence it can be checked if a theory is consistent or, to put it differently, if there is an interpretation such that the theory denotes a nonempty set in that interpretation. The linguistically even more relevant question if a sentence is grammatical is formally different. To decide if a sentence is grammatical, it has to be checked if there exists an interpretation such that the theory denotes all objects in the universe and the description of the sentence denotes a nonempty set in that interpretation. This condition is clearly stronger than the former.

1.3 Recursive Sort Definitions

The task of ordering the phonological string of a sentence is so obvious and crucial for a linguistic theory that at first glance it comes as a surprise that HPSG does not offer a solution for it from the start. Since the topic of constituent order and word order has been important for GB theory and was one of the central concerns in the development of GPSG, which influenced the molding of HPSG, the lack of an explicit theory of linear order is even more remarkable.

[9] The notion of grammaticality of a description as used here is defined in (Kepser 1994, p.9).

Considering the technical devices that are needed to construct a theory of word order with a descriptive formalism as presented in 1.2 it becomes clear that the real issue is the format of the implicational constraints which have to be used. The task to be solved is the ordering of phonological strings on a list of arbitrary length. But in order to treat lists of arbitrary length, simple structure sharing or, in the terms of our logic, path equations will not do, since the length of the attribute terms that have to be equated is not known in advance. Examples for principles in which the paths always remain constant and the problem does not occur are the Head Feature Principle, the Marking Principle and the Spec Principle.

(1.3) *The Head Feature Principle*

$$\left[\begin{array}{l} \textit{phrase} \\ \textsc{dtrs}\ \textit{headed-struc} \end{array}\right] \implies$$

$$\left[\begin{array}{ll} \textit{phrase} & \\ \textsc{synsem loc cat head} & \boxed{1} \\ \textsc{dtrs head-dtr synsem loc cat head} & \boxed{1} \end{array}\right]$$

The constraint (1.3) can be applied to each phrase for which the antecedent is true no matter what the rest of the description looks like and will always yield the desired result. A simple case in which mere structure sharing is not sufficient to achieve the intended effect, though, is the Subcategorization Principle. In the version of (Pollard and Sag 1994, p.399) it requires that in a phrase whose DTRS value is of sort *headed-struc* the list value of the head daughter's SUBCAT list be the concatenation of the SUBCAT list of the phrase with the list consisting of the synsem objects of the elements of the list value of the DTRS COMP-DTRS list. Obviously, this can not be written in the same way as (1.3), because we do not know how long the DTRS COMP-DTRS list and the phrase's SUBCAT list are and, consequently, where we have to look for the synsem objects that have to be collected. Thus we can formalize the antecedent of the Subcategorization Principle, but we do not know what the consequent looks like in the general case.

(1.4) *The Subcategorization Principle*

$$\left[\begin{array}{l} \textit{phrase} \\ \textsc{dtrs}\ \textit{headed-struc} \end{array}\right] \implies$$

$$\begin{bmatrix} \textit{phrase} & \\ \text{SYNSEM} & \begin{bmatrix}\text{LOC CAT SUBCAT} < \boxed{1}, \ldots, \boxed{i-1} > \end{bmatrix} \\ \text{DTRS} & \begin{bmatrix} \text{HEAD-DTR} & \begin{bmatrix} \textit{sign} \\ \text{SYNSEM LOC CAT SUBCAT} < \boxed{1}, \ldots, \boxed{i}, \ldots, \boxed{n} > \end{bmatrix} \\ \text{COMP-DTRS} & \left\langle \begin{bmatrix} \textit{sign} \\ \text{SYNSEM} \boxed{i} \end{bmatrix}, \ldots, \begin{bmatrix} \textit{sign} \\ \text{SYNSEM} \boxed{n} \end{bmatrix} \right\rangle \end{bmatrix} \end{bmatrix}$$

Example (1.4) illustrates the problem that occurs with the Subcategorization Principle. Neither the value of i nor the value of n, both tags interpreted as indices designating the linear position of the respective elements on the subcat list of the head daughter, are known, hence the precise paths can not be stated. Therefore an accurate formalization has to employ some additional mechanism.

In fact, a short examination of HPSG principles reveals that a considerable number of them can not be stated as easily as the Head Feature Principle, including the Nonlocal Feature Principle, the Relative Uniqueness Principle, the Quantifier Binding Principle, the Semantics Principle and, of course, the Binding Theory. All of them face the problem that they involve statements on recursively built paths in the description language. Pollard and Sag (1994) do not explicitly address this problem, and it is largely hidden by the fact that they do not go beyond a verbal description of the formal principles they have in mind. Their strategy of ignoring the issue can be successful as long as the number of lists of varying length in one principle is low and an informal description still understandable. This is the case if the variables comprise simple concepts like the concatenation of lists or the demand that there be exactly one element with certain properties on a list. The situation becomes more problematic in increasingly complex structures, and its precariousness is reflected in the formulation of the Binding Theory, which employs a number of terminological definitions which serve as abbreviations for the description of certain recursive structures.[10]

A formalization of a theory of word order in HPSG is immediately confronted with the issue of recursive structures in descriptions, and the number of variables is considerably big from the beginning. Not only is there a phonological string of elements of undetermined length to be ordered, but it is to be done for sentences with various, recursively defined, syntactic structures. Each principle deals with an unknown number of words which are to be found in lists of arbitrary length and have to be classified according to their properties. Moreover, as we have argued in section 1.1, constituent order in the German mittelfeld is not only sensitive to properties of the constituents that have to be ordered, but

[10] c.f. the notion of being more oblique than, (local) o-command, (local) o-freedom and its negation, and (local) o-binding in (Pollard and Sag 1994, p.401).

also to some structural conditions and to properties of the main verb of the sentence. Thus, the lack of a formal treatment of constraints on recursive structures in (Pollard and Sag 1994) seems to be at least partly responsible for the omission of a theory of word order.

Since the problem of a precise formalization of principles such as the Subcategorization Principle or the Binding Theory and of LP principles is formally closely related, we conclude that a theoretically adequate attempt at constructing LP principles should reflect this relationship. A satisfying solution for LP principles should ideally avail itself of the same general mechanism that can be used for the former principles as well. From this point of view it is for example not apposite to augment the formalism of HPSG with a phrase structure backbone in order to express word order generalizations with LP statements, since phrase structure rules do not contribute to a better understanding of technical solutions for the Semantics Principle. It is noteworthy, though, that phrase structure rules supply a well understood recursive mechanism which allows to express LP statements elegantly. The recursive nature of phrase structure rules can be taken as yet another unequivocal indication of what issue is at stake.

Given a logic such as the one defined in section 1.2, there are several approaches conceivable. The first one we like to discuss concerns extensions of the syntax and semantics of the logic by adding non-standard logical connectives. It is the route taken by Reape (1989, 1994), whose conception in many aspects provides the background for the considerations of Pollard et al. (1994). The idea is to introduce logical operators for functions that are useful for linguistic descriptions, such as $\bullet$ for list concatenation and $\preceq$ for linear precedence descriptions.[11] More precisely, $\bullet$ is defined as performing a sequence concatenation of two lists of arbitrary length. $\preceq$ is an operator that allows for determining a precedence relation of two elements on a list.

Additional operators in the logic clearly provide some part of the functionality that is needed for formalizing grammatical principles. One of the nice aspects of operators such as $\bullet$ and $\preceq$ is their sort independence. They apply to lists or elements of lists regardless of their species. On the other hand, they quickly inflate the logical formalism, lead to a rather complex and unattractive semantics and are specifically designed to perform certain tasks in linguistic descriptions. At the same time, the logic does not change substantially, since, as we will show, the tasks the new operators perform can be expressed in the logic without them. The line between underlying formalism and linguistic theory becomes blurred by shifting some content of the actual linguistic theory to the formalism.

[11] See (Reape 1989, pp.104-106) for a logic endowed with the syntax and semantics of these operators.

One would rather expect that the formalism supplies the standard logical connectives and leaves the description of specific constraints entirely to the description language and the linguist using it rather than to the logician.

The last point is amplified by the observation that sort unspecific list concatenation already fails to be sufficient in as simple cases as the Subcategorization Principle in example (1.4). Here the special procedure of extracting some information from a larger structure, namely synsem objects from lists of signs, and appending the list of extracted elements to another list (of synsems) is asked for. It seems to make little sense to expect this kind of operators from a general logical formalism.

We conclude that the approach to extend the underlying logic by task-specific logical operators is methodically questionable and refrain from persuing this course any further.

The second approach to recursive mechanisms for a description language of HPSG we want to discuss originates from the theory of programming languages and computer implementations of natural language fragments which are specified in HPSG. The construction in question has received various names depending on the choice of the authors and is known as definite relations, definite clause specifications or definite clause relations. For our expository purposes we can simply assume that definite relations are relations that are written in a syntax closely related to the one used in the programming language Prolog. The variables of these relations range over sorted feature structure descriptions. In effect, definite relations allow for the formalization of a function such as `append` for the concatenation of two finite lists of the same sort and `prec` for a precedence relation between two elements on a finite list in some formal language. Using definite relations as an extension of our description language, the Subcategorization Principle looks as follows:

(1.5) *The Subcategorization Principle with definite relations*

$$\begin{bmatrix} \textit{phrase} \\ \text{DTRS}\ \textit{headed-struc} \end{bmatrix} \Longrightarrow$$

$$\begin{bmatrix} \textit{phrase} & \\ \text{SYNSEM} & \left[\text{LOC CAT SUBCAT}\ \boxed{1}\ \textit{list-of-synsems}\right] \\ \text{DTRS} & \begin{bmatrix} \text{HEAD-DTR} & \begin{bmatrix} \textit{sign} \\ \text{SYNSEM LOC CAT SUBCAT}\ \textit{append}(\boxed{1}, \textit{extract-synsems}(\boxed{2})) \end{bmatrix} \\ \text{COMP-DTRS} & \boxed{2}\ \textit{list-of-signs} \end{bmatrix} \end{bmatrix}$$

`extract-synsems` and `append` are not to be interpreted as sorts here but as definite relations. `extract-synsems` is defined in such a way that it produces a list of the synsem elements on the DTRS COMP-DTRS list.

`append` then appends this list to the SYNSEM LOC CAT SUBCAT list. As a result, we arrive at the desired list as value for the subcat list of the head daughter.

The formal concept underlying the idea of definite relations is presented in detail in (Höhfeld and Smolka 1988). Höhfeld and Smolka show how a constraint language which is not necessarily a sublanguage of predicate logic can be extended to a constraint language providing general definite clause specifications. This technique, which was developed against the background of logic programming, has been exploited by systems for the implementation of HPSG grammars. Examples for this strategy are CUF by Dörre and Dorna (1993), the ALE system by Bob Carpenter and Gerald Penn and the Troll system of Dale Gerdemann and Thilo Götz (c.f. Carpenter 1993, Gerdemann 1993). Consequently, definite clause specifications are utilized in computer implementations of natural language fragments as for example in the implementation of the analysis of partial-VP extraction and split-NP topicalization of Hinrichs and Nakazawa (1994) in a grammar written for ALE and Troll by Meurers and Richter (1994). One should keep in mind, though, that the programming environment for implementations deviates rather strongly from the HPSG formalism set out in (Pollard and Sag 1994). In the case of Troll, a variant of the logic of (King 1994) is only one part in a formalism which also provides a phrase structure backbone that is vividly used in the implementation of grammars.[12] The existence and use of definite clause specifications in computational systems by no means implies that a descriptive formalism for HPSG which includes definite relations actually exists.

Pollard and Sag (1994, p.8) seem to refer to the literature on logic programming when they say that the logic they envisage should be augmented with definite relations. Later in the text they mention 'functional and relational symbols such as append'[13], which are to be part of the description language, i.e., of the AVM notation of Pollard and Sag. Allusions to the intended extension of the description language can be traced back as far as (Pollard and Sag 1987, p.48, 104, 109, 111), where several examples of definite relations (in this case called functionally dependent values) are given and explained as denoting recursive functions.[14] What

[12]For the relationship between an implemented grammar and its HPSG specification c.f. (Meurers 1994). Meurers shows that the grammar of Meurers and Richter (1994) is basically a traditional phrase structure grammar which is extended by a signature and uses certain notational conventions to give it the flavor of HPSG.

[13](Pollard and Sag 1994, p.21)

[14]c.f. (Pollard and Sag 1987, p.48 and 104). For more information about the notion of functionally dependent value they refer to section 2.3 of the book, but that section neither mentions functionally dependent values nor anything related to this concept. Since section 2.3 consists of a short summary of the technical literature concerned with

is missing, though, is any formal specification of the logical constructions they have in mind. This is a rather strong drawback of this approach, since definite relations as Pollard and Sag use them are neither attributes of any kind nor do they belong to the sort hierarchy. In short, they are not part of the syntax of the descriptive formalism. Hence they are not defined in the formalism.

Despite their still missing formalization within a formalism for (Pollard and Sag 1994), definite relations are an attractive and very flexible option for bringing recursive mechanisms into the principles of grammar. To put them to use it would be necessary to extend SRL in the sense of (Höhfeld and Smolka 1988). One should be aware of the fact, though, that such an extension leads to a change in the ontology of the formalism by adding relations, and it remains to be seen yet if the resulting setup could be satisfactorily close to the intuitive notions that are put forward in (Pollard and Sag 1994). For the time being, definite relations in HPSG remain a desideratum, and it is not clear to us if they can provide any notational or formal advantages compared to the current logical setup. At this stage, it seems that every linguistically relevant function that can be formalized with definite relations can be translated into the more conservative technique we present below and vice versa. This situation suggests that it is not necessary to go beyond the logic presented in section 1.2 to formalize properly whatever linguists wish to express with definite relations.

The approach we are going to adopt is based on an idea of Ait-Kaci (1984), who exemplifies his method with a formalization of `append`. The idea is taken up by Carpenter (1992, pp.238-242) in a discussion of definite constraints and by King (1992, p.34), who attributes it to Alex Franz and formalizes the so-called functionally dependent values of the Semantics Principle of (Pollard and Sag 1987) as well as `append` functions for lists of several sorts in SRL. In order to illustrate the method, we are going to formalize a constraint that can be added to the Subcategorization Principle in example (1.4) to yield a formally precise definition entirely within the logic of section 1.2. The constraint will exactly correspond to the definite relations sketched in example (1.5). Following (Meurers 1994) we call the kind of constraints that introduce recursivity into the description language recursive sort definitions.

A recursive sort definition consists of two parts, an appropriateness condition and an implicational constraint. In our presentation, a recursive sort definition begins with the statement of the appropriateness conditions for a new sort. With the appropriateness conditions the sort hierarchy is enlarged by a new sort and its subsorts. At the same time the

feature structures, the logic of partial information and computational implementations of unification grammars, one might deduce that they refer to implementational work.

attributes are declared for which objects of the new species are defined. In contrast to (Pollard and Sag 1994) we do not split the declarations into one part for the sort hierarchy and a second part for feature declarations, but define both parts in one step.

The appropriateness condition starts with the supersort of the new section of the sort hierarchy. Together with the supersort the attributes for which every subsort of the supersort is defined are given. Subsorts of a sort come after their supersorts and are indented. If a subsort is a species then all attributes it is defined for occur together with it, otherwise all attributes its subsorts have in common are printed. The sort the appropriateness function $\mathfrak{F}$ assigns to a species and an attribute which is defined for that species stands on the righthand side of each attribute. Example (1.6) illustrates this notation with the appropriateness conditions for objects of sort *sub-append*.

(1.6) *Appropriateness Conditions for the sort sub-append*

sub-append	LEFT	*list-of-synsems*
	RIGHT	*list-of-signs*
	RES	*list-of-synsems*
unfinished-append	LEFT	*nelist-of-synsems*
	RIGHT	*list-of-signs*
	RES	*list-of-synsems*
	REC	*sub-append*
part-fin-append	LEFT	*elist*
	RIGHT	*nelist-of-signs*
	RES	*list-of-synsems*
	REC	*sub-append*
finished-append	LEFT	*elist*
	RIGHT	*elist*
	RES	*elist*

By our notational conventions the appropriateness conditions in (1.6) introduce a sort *sub-append*. Objects of sort *sub-append* come in three species: *unfinished-append*, *part-fin-append* and *finished-append*. (1.7) shows the resulting partial order of sorts with the root sort at the root of the tree and the maximal sorts as leaves.

(1.7) *Sort hierarchy for sub-append*

subappend

unfinished-append *part-fin-append* *finished-append*

In our logic this means that objects of sort *sub-append* fall into one of three categories of maximal sorts. All of them are defined for the attributes LEFT, RIGHT and RES (for result). Objects of species *unfinished-append* and *part-fin-append* are defined for the additional attribute REC (for recursion), and the interpretation of REC with respect to objects of these two species yields an object of sort *sub-append*. This puts us into a position to write a recursive theory of *sub-append*.

Before we present the theory of *sub-append*, let us insert some remarks on the encoding of lists in HPSG. Like the programming language Prolog, HPSG encodes lists of arbitrary length by splitting the description of the list recursively into the first element on the list and the rest of the list. Since lists in HPSG are sorted we arrive at the following schema for encoding a list of elements of sort σ.

(1.8) *Appropriateness Conditions for the sort list-of-σs*

list-of-σs			
	elist		
	nelist-of-σs	FIRST	σ
		REST	*list-of-σs*

By substituting some sort of the sort hierarchy for σ we obtain the appropriateness conditions for lists of that sort. Throughout the rest of the thesis we tacitly assume that lists of the sorts we use have been defined in this manner. In AVMs we use the common abbreviations. $<>$ simply stands for *elist*. For a list with elements of sort σ the abbreviation $< \sigma \mid$ *list-of-σs* $>$ stands for the more explicit AVM in example (1.9):

(1.9) *Lists in the description language*

$$\begin{bmatrix} \textit{nelist-of-}\sigma\textit{s} \\ \text{FIRST } \sigma \\ \text{REST } \textit{list-of-}\sigma\textit{s} \end{bmatrix}$$

Although this method to encode lists is standardly used in the literature it is perhaps worth mentioning that not all logical setups for HPSG support this encoding. King and Götz (1993, pp.19-22) point out that the feature introduction condition which Carpenter (1992, p.86) establishes for the treatment of type inference[15] is not compatible with multiple sorts for lists using the schema (1.8). Since our logic does not impose a feature introduction condition, there is no need to avoid the schematic sort hierarchy (1.8) for lists.

[15]The feature introduction condition requires that for each attribute α there be a least specific sort that deserves α.

Having explained our treatment of lists, we can now return to the discussion of recursive sort definitions.

The second part of a recursive sort definition consists of an implicational constraint which states, how the objects of the new sort are structured. The sort hierarchy and the values of the appropriateness function are utilized to do this recursively. In later chapters of this thesis we provide the implicational constraints both in term notation for mathematical precision and in AVM notation for better readability for the linguist. At this point we simply state the AVMs for the theory of *sub-append*:

(1.10) *Theory of sub-append*

$$\begin{bmatrix} \textit{unfinished-append} \end{bmatrix} \Longrightarrow \begin{bmatrix} \textit{unfinished-append} & \\ \textsc{left} & < \boxed{1} \mid \boxed{2} > \\ \textsc{right} & \boxed{3}\ \textit{list-of-signs} \\ \textsc{res} & < \boxed{1} \mid \boxed{4} > \\ \textsc{rec} & \begin{bmatrix} \textit{sub-append} & \\ \textsc{left} & \boxed{2} \\ \textsc{right} & \boxed{3} \\ \textsc{res} & \boxed{4} \end{bmatrix} \end{bmatrix}$$

$$\begin{bmatrix} \textit{part-fin-append} \end{bmatrix} \Longrightarrow \begin{bmatrix} \textit{part-fin-append} & \\ \textsc{left} & \textit{elist} \\ \textsc{right} & \left\langle \begin{bmatrix} \textit{sign} & \\ \textsc{synsem} & \boxed{1} \end{bmatrix} \mid \boxed{2} \right\rangle \\ \textsc{res} & < \boxed{1} \mid \boxed{3} > \\ \textsc{rec} & \begin{bmatrix} \textit{sub-append} & \\ \textsc{left} & \textit{elist} \\ \textsc{right} & \boxed{2} \\ \textsc{res} & \boxed{3} \end{bmatrix} \end{bmatrix}$$

In effect, the recursive sort definition for *sub-append* says that objects of sort *sub-append* are well-formed iff the value of their RES attribute is the concatenation of the finite list of synsems in their LEFT attribute with the synsem objects that are (in order of their appearance on the list) extracted from the finite list of signs of their RIGHT attribute. This statement is of course a proposition. Since all components involved are rigorously formalized it can be proved correct (or false). For every recursive sort definition in our theory of constituent order we will provide a proposition and the proof of its correctness.

Example (1.11) illustrates the properties and working of the sort *sub-append* for a LEFT list of length one and a RIGHT list of length two.

(1.11) *An example for sub-append*

$$
\left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{unfinished-append}} \\
\text{LEFT} & < \boxed{1}\ \textit{synsem} > \\
\text{RIGHT} & \left\langle \boxed{2} \left[\begin{array}{l} \textit{sign} \\ \text{SYNSEM}\ \boxed{3}\ \textit{synsem} \end{array}\right], \boxed{4} \left[\begin{array}{l} \textit{sign} \\ \text{SYNSEM}\ \boxed{5}\ \textit{synsem} \end{array}\right] \right\rangle \\
\text{RES} & < \boxed{1}\ \textit{synsem},\ \boxed{3}\ \textit{synsem},\ \boxed{5}\ \textit{synsem} > \\
\text{REC} & \left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{part-fin-append}} \\
\text{LEFT} & \textit{elist} \\
\text{RIGHT} & \left\langle \boxed{2} \left[\begin{array}{l} \textit{sign} \\ \text{SYNSEM}\ \boxed{3} \end{array}\right], \boxed{4} \left[\begin{array}{l} \textit{sign} \\ \text{SYNSEM}\ \boxed{5} \end{array}\right] \right\rangle \\
\text{RES} & < \boxed{3}\ \textit{synsem},\ \boxed{5}\ \textit{synsem} > \\
\text{REC} & \left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{part-fin-append}} \\
\text{LEFT} & \textit{elist} \\
\text{RIGHT} & \left\langle \boxed{4} \left[\begin{array}{l} \textit{sign} \\ \text{SYNSEM}\ \boxed{5} \end{array}\right] \right\rangle \\
\text{RES} & < \boxed{5}\ \textit{synsem} > \\
\text{REC} & \left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{finished-append}} \\
\text{LEFT} & \textit{elist} \\
\text{RIGHT} & \textit{elist} \\
\text{RES} & \textit{elist}
\end{array}\right]
\end{array}\right]
\end{array}\right]
\end{array}\right]
$$

The overall object is of species *unfinished-append*, since the LEFT list is nonempty. Objects of species *part-fin-append* have to have an empty LEFT list, and for objects of species *finished-append* the LEFT list and the RIGHT list have to be empty. Because we are dealing with an object of sort *unfinished-append*, the first constraint of (1.10) applies. Therefore the single element on the LEFT list is the first element on RES, and the interpretation of REC maps the *unfinished-append* object to another object of sort *sub-append*. More specifically, it is of sort *part-fin-append*. This is so, because by the applied constraint the first element of LEFT and RES are missing on REC LEFT and REC RES respectively and only the tails of the first two lists are passed on. In addition, the RIGHT list equals REC RIGHT. But that means that the LEFT list of the *part-fin-append* object is empty whereas RIGHT is still nonempty. Hence the object can indeed only be of species *part-fin-append* and not of species *unfinished-append* or *finished-append*.

For objects of sort *part-fin-append* the second constraint of (1.10) must hold true. This means, that the SYNSEM value of the first element on its RIGHT list is also the first element on the RES list. The tail of RIGHT and RES are again passed on to an object to which the *part-fin-append* object we are currently looking at is mapped by the interpretation of its REC attribute. This object is again of species *part-fin-append*, because according to the constraint on the former object its LEFT attribute is of value *elist* and its RIGHT value is equal to the tail of the RIGHT list of the former object. Also, its RES list, which equals the SYNSEM value of the one remaining element on the RIGHT list by the second constraint of (1.10), is equal to the tail of the RES list of the former object. As both the RIGHT and the REST list are of length one, their tails are of value *elist*. Hence the interpretation of REC maps the object to an object with empty RIGHT, RES and LEFT list. Thus it must be of species *finished-append* and the recursion ends.

The sort *sub-append* can be used to make the Subcat Principle of example (1.4) taken from (Pollard and Sag 1994) precise. All we have to do is to amend the appropriateness function $\mathfrak{F}$ at a suitable point. Since the Subcategorization Principle is valid for phrases whose DTRS attribute is of sort *headed-struc*, we might insert an attribute APPEND for headed structures and rewrite the Subcategorization Principle as follows:

(1.12) *The Subcategorization Principle with recursive sort definitions*

$$\begin{bmatrix} \textit{phrase} \\ \text{DTRS } \textit{headed-struc} \end{bmatrix} \Longrightarrow$$

$$\begin{bmatrix} \textit{phrase} & \\ \text{SYNSEM} & \left[\text{LOC CAT SUBCAT } \boxed{1}\ \textit{list-of-synsems}\right] \\ \text{DTRS} & \begin{bmatrix} \textit{headed-struc} & \\ \text{HEAD-DTR} & \begin{bmatrix} \textit{sign} \\ \text{SYNSEM LOC CAT SUBCAT } \boxed{3}\ \textit{list-of-synsems}\end{bmatrix} \\ \text{COMP-DTRS} & \boxed{2}\ \textit{list-of-signs} \\ \text{APPEND} & \begin{bmatrix} \textit{sub-append} & \\ \text{LEFT} & \boxed{1}\ \textit{list-of-synsems} \\ \text{RIGHT} & \boxed{2}\ \textit{list-of-signs} \\ \text{RES} & \boxed{3}\ \textit{list-of-synsems}\end{bmatrix}\end{bmatrix}\end{bmatrix}$$

Since the sort *sub-append* is fully formalized, it is now even possible to prove that (1.12) delivers precisely what Pollard and Sag (1994, p.399) wish to express with the Subcategorization Principle.

The theory of *sub-append* and its application in (1.12) unfold the structure and use of recursive sort definitions. As we have seen, a suitably constructed sort hierarchy serves as mechanism to distinguish several cases of the application of a recursive function. The attributes for which the species are defined establish the connection to the description outside of the new sort and handle the recursion. Usually their values come as lists whose first-rest structure can be exploited for recursive constraints. In procedural terms, some of those lists can be seen as input to the recursive sort and some as its output. Objects of some of its species are defined for an attribute whose interpretation yields the supersort and therefore allows for a constraint that refers to the object of the constraint and an object of the same sort that is in some sense embedded into the first. In writing these constraints, then, techniques from specifying recursive functions as they are known from logic programming can be applied.

Recursive sort definitions reveal the properties we want for a mechanism in which the LP component of the grammar can be formalized. They are mathematically explicit and can be captured in the already existing formalism of (King 1994), which sticks closely to the theory of (Pollard and Sag 1994). Thus there is no need to define a particular syntax and semantics for LP rules. Moreover, as we have demonstrated for the Subcategorization Principle, the same technique that we will use to build an LP component of the grammar can be used to make necessary amendments for traditional principles of HPSG. As a result, one simple formalism is sufficient to capture every principle of grammar.

Recursive sort definitions also turn out to be a very flexible tool that allows for modular extensions of the grammar. An example for this property provides the Subcategorization Principle in (1.12), which differs from the informal account in (1.4) only by an additional attribute for *headed-struc* and the recursive sort definition, whereas there are no changes in the traditional appropriateness conditions.

In the remainder of the thesis we sketch the architecture of an LP component that complies with the general requirements for LP rules we have discussed. The emphasis of our present investigation is put on the formal concepts that are needed, and the syntactic analyses we choose for the examples are not meant to be optimal. Quite to the contrary, the analysis of the German mittelfeld adopted in the following chapters in many respects represents the worst possible case for the LP component, because it makes it difficult to access the information which is necessary to formulate LP rules. This will give us ample opportunity to explore the possibilities and flexibility of our approach.

The ultimate goal of a theory of LP is to develop a library of sorts that can be used as a toolbox for writing LP constraints for natural languages. A parsimonious theory should use as few recursive sort definitions as possible, and it would be an interesting task to find out which basic recursive sort definitions are needed. In the case of the Subcategorization Principle in (1.12) above, a general sort *append* for the concatenation of lists of synsem objects could be combined with another sort that extracts a list of synsem objects from a list of signs. Both sorts are needed at other places in the grammar as well, whereas the specific sort *sub-append* can be dispensed with.

2 A General Restriction on the PHON-List

In the introduction it was claimed that natural languages obey a universal constraint which requires that the phonological string of a phrase be always some permutation of the phonological strings of that phrase's daughters. Loosely speaking, a phrase's daughters are the signs contained in the value of the DTRS attribute of the phrase plus recursively all signs that are contained in the DTRS attributes of deeper embedded phrases. Since we assume the permutation constraint to be universal, it is independent of language specific linearization principles and is therefore not part of the LP component proper. The task of the permutation constraint is to make sure that only permutations of the phonology of the words that occur within a phrase can occur in the phonology of the phrase. The LP principles then select the grammatical phonologies from these permutations.

Basically the same idea of a general constraint on the value of the attribute PHON is presented by Pollard and Sag (1987, chapter 7), who propose a Constituent Order Principle in form of a definite relation:[16]

(2.1) *(=320)* *Constituent Order Principle*

$$\textit{phrasal-sign}\begin{bmatrix}\ \end{bmatrix} \Longrightarrow \begin{bmatrix}\text{PHON} & \texttt{order-constituents}(\boxed{1}) \\ \text{DTRS} & \boxed{1}\end{bmatrix}$$

Pollard and Sag (1987) do not provide a definition of `order-constituents`, but they try to give an intuition about what it means:

> [...] In general, however, we assume that order-constituents always gives as its value a permutation [...] of the PHONOLOGY values of the sign's daughters. (Pollard and Sag 1987, p.169)

It is clear from the context that with this definite relation Pollard and Sag only intend to impose a constraint that relates the phonologies of the immediate daughters of a phrase and not of the daughters in general to the PHON value of that phrase. Therefore the definite relation `order-constituents` can be seen as a specialized instance of a more liberal permutation constraint. The structure-sharing in (2.1) is supposed to reflect the fact that the actual order of PHON values of the daughters on the phrase's PHON list depends not just on the PHON values of the daughters but potentially on the entire syntactic and semantic structure of the daughters. For example, the order in which complements

[16] The type *phrasal-sign* in (2.1) corresponds to the sort *phrase* in (Pollard and Sag 1994).

appear on the SUBCAT list of their syntactic head could constrain their linear order in a phrase. In short, further restrictions on the output of `order-constituents` which encode language specific ordering principles can make use of the full range of information that is available within the value of DTRS.

At the end of chapter 7, the authors observe that this may not be all a linguist needs for the description of natural languages. According to their intuitive interpretation of (2.1) it is only possible to construct permutations of the complete PHON values of daughters, but it is not allowed to interleave the phonological lists of two daughters. Therefore it is not clear how to describe languages which exhibit absolutely free word order or it might be problematic to cope with scrambling phenomena in some languages. To remedy this issue, they suggest a second definite relation `interleave-constituents`, which is supposed to allow the interleaving of the PHON values of daughters in a syntactically higher phrase, but they fail to give even the slightest hint at how they want to make this idea more precise (Pollard and Sag 1987, p.189). It is however evident that the second definite relation is supposed to allow the permutation of the phonology of every word in a phrase.

In this chapter we will define an attribute for objects of species *phrase* that yields objects of sort *build-phon* as value. *build-phon* supplies the permutations of the phonological strings for a list of signs. The list of signs may be a list of the immediate daughters of a phrase or may contain non-immediate daughters. The actual content of the list of signs for which the theory of *build-phon* delivers the phonological permutations will be constructed by other constraints in a different part of the grammar which will be discussed in chapter 3. The interaction of these two components will yield a precise formalization of the definite relations Pollard and Sag describe.

2.1 Introducing *build-phon*

The internal structure of objects of species *build-phon* embodies the core of a permutation constraint on the PHON list of phrases. Although its structure is still easy to survey, it displays a complex interaction of several recursive sort definitions. Thus the presentation of *build-phon* will convey a feeling for how a complex constraint consisting of several recursive sort definitions is built and how appropriateness conditions and theories for sorts relate to each other. Familiarity with the techniques introduced here is essential for an understanding of the working of the principles in the chapter on the LP component.

Objects of species *build-phon* are the value of the attribute BUILD-PHON applied to objects of sort *phrase*. BUILD-PHON is only defined for phrases, since the concept of a permutation of the phonology of daughters is only pertinent to phrases and not to words. The AVM in (2.2) shows the structure of phrases in accordance with the appropriateness conditions of (Pollard and Sag 1994), extended by attributes and sorts that play a role in linearization. Formal definitions of these extensions will be given later.

(2.2) *The structure of phrases*

$$
\begin{bmatrix}
\textit{phrase} & \\
\text{PHON} & \boxed{1} \\
\text{SYNSEM} & \textit{synsem} \\
\text{BUILD-PHON} & \begin{bmatrix} \textit{build-phon} & \\ \text{PERM} & \begin{bmatrix} \textit{perm} & \\ \text{ORIGIN} & \boxed{2}\ \textit{list-of-signs} \\ \text{PERMUTED} & \boxed{3}\ \textit{list-of-signs} \end{bmatrix} \\ \text{SIGNPHON} & \begin{bmatrix} \textit{signphon} & \\ \text{SIGNS} & \boxed{3}\ \textit{list-of-signs} \\ \text{PHONS} & \boxed{1}\ \textit{list-of-phons} \end{bmatrix} \end{bmatrix} \\
\text{LP} & \textit{lp} \\
\text{DTRS} & \begin{bmatrix} \textit{const-struc} \\ \dots \\ \text{ALL-DTRS}\ \boxed{2} \end{bmatrix}
\end{bmatrix}
$$

Besides the attribute BUILD-PHON and the new attributes that occur within *build-phon* objects on which we comment presently there are two new attributes in (2.2).

The attribute LP, defined for phrases and denoting a function from phrases to objects of sort *lp*, connects the module of the grammar which contains the LP principles to phrases. The sort *lp* will be discussed in chapter 4.

The second new attribute, ALL-DTRS, is defined on objects of sort *const-struc* and relates them to lists of signs. In the LP principles that we formalize we only address head complement structures, i.e., the species *head-comp-struc* of the supersort *const-struc*. Head complement structures have a head daughter and a (possibly empty) list of complement daughters. The idea is that the ALL-DTRS list contains a list of signs comprising the head daughter as first element on the list and the complement daughters. This list serves as an interface to *build-phon* and supplies the list of signs for which the permutations of the phonological strings are provided. In the case of free word order languages, though, ALL-DTRS contains not only a list of the immediate daughters of a constituent structure, but of all daughters of a given phrase. It is also conceivable to differentiate further as to which signs occur on the ALL-DTRS list.

The structure sharing indicated in (2.2), which has to be defined in constraints on phrases, connects the ALL-DTRS list with *build-phon* objects. The purpose of objects of sort *build-phon* is to take this list as input, indicated by tag $\boxed{2}$, and specify every permutation of the PHON lists of the signs on the ALL-DTRS list as admissible for BUILD-PHON SIGNPHON PHONS and thus, by tag $\boxed{1}$, for the entire phrase. Therefore, it depends on the theory that specifies the members of the ALL-DTRS list (which in turn might be specified differently for different constituent structures) whether we arrive at an instance of the relation `order-constituents` or at an instance of `interleave-constituents` of (Pollard and Sag 1987). Of course, in some constituent structures a variant of `interleave-constituents` might be appropriate, while in others a variant of the more rigid definite relation `order-constituents` might be the solution of choice.

In any case, BUILD-PHON PERM ORIGIN takes the list of signs in ALL-DTRS as input and BUILD-PHON PERM PERMUTED admits every permutation of that list. These permutations of signs serve as input to BUILD-PHON SIGNPHON SIGNS, and in BUILD-PHON SIGNPHON PHONS the corresponding list of PHON values is contained. In effect, we allow exactly every permutation of the PHON values of the list of signs in ALL-DTRS as value of the PHON attribute of the phrase.

Two properties of this architecture should be noted. First, the part of the grammar that provides the permutations of phonological strings of signs which are part of a given phrase is independent of the LP principles. It can thus be viewed as a module of its own. And second, the sort *build-phon* and its functioning are independent of the syntactic analysis of phrases which is encoded within the *synsem* object of phrases. The component which constructs the content of the ALL-DTRS list serves as clearly defined interface to the language specific analysis of constituent structures and has to comply with the particular demands of a language as to the freedom of word order.

We are now ready to state the appropriateness conditions that are needed to formalize the theory of *build-phon*.

(2.3) *Appropriateness Conditions for the sort build-phon*

build-phon	PERM	*perm*
	SIGNPHON	*signphon*

As we have already mentioned, the theory of *perm* will relate an input list to every possible permutation of that list, and the theory of *signphon* is responsible for extracting the phonological information from the permutations of the list of signs provided by the theory of *perm*.

2.2 A sort for the permutation of lists

Before we get to the formalization of *signphon*, we investigate the sort hierarchy and appropriateness conditions of *perm*.

(2.4) *Appropriateness Conditions for the sort perm*

perm	ORIGIN	*list-of-signs*
	PERMUTED	*list-of-signs*

perm0	ORIGIN	*elist*
	PERMUTED	*elist*
perm1	ORIGIN	*nelist-of-signs*
	PERMUTED	*nelist-of-signs*
	TEST	*take-out*
	REC	*perm*

By notational convention for appropriateness conditions, *perm0* and *perm1* are speciate subsorts of *perm*.

perm0 evidently takes care of the base case, in which both lists are empty. Species *perm1* allows to bring recursion into the theory of *perm*, since the appropriateness function $\mathfrak{F}$ has the supersort *perm* as its value for *perm1* and REC. Furthermore, both the ORIGIN and the PERMUTED list must be nonempty for objects of sort *perm1*. The theory of *perm1* we envision must ensure that the PERMUTED list of any *perm1* object be a permutation of its ORIGIN list.

There is one problem remaining for the formalization of a theory of *perm1*. It can only be well-defined once all subparts have been defined. Therefore, we have to supply the appropriateness conditions and the theory of the new sort *take-out* first. We will see that the theory of *take-out* plays a crucial role in the theory of *perm1*.

It may be helpful for understanding the following technical definitions to keep in mind how the permutations of a given list of objects can be obtained. The way it is done here, an arbitrary element of the original list is taken and put into the first position of the new list. This task will be fulfilled by *take-out* and ensures that every element of the original list may end up in the first position of the permuted list. In the next step the recursive structure of *perm* objects comes to bear. We take the original list minus the element which is in the first position of the new list and, if it is nonempty, again pick out an arbitrary element which is put into the first position of the rest of the permuted list. Since this procedure is repeated as long as the reduced original list is nonempty, every permutation of it can be constructed. With the additional condition that the element we choose from the original list be its first occurrence on that list we manage to avoid obtaining several instances of indiscernible permutations that are described by different descriptions of *perm* objects. In other words, we can be sure that every permutation of the original list is described by a unique *perm* object. (2.5) displays the appropriateness conditions for *take-out*.

(2.5) *Appropriateness Conditions for the sort take-out*

take-out	SET	*nelist-of-signs*
	ELE	*sign*
	NSET	*list-of-signs*

take-out0	SET	*nelist-of-signs*
	ELE	*sign*
	NSET	*list-of-signs*
take-out1	SET	*nelist-of-signs*
	ELE	*sign*
	NSET	*list-of-signs*
	REC	*take-out*

Again, a sort which introduces recursion (*take-out1*) and one which does not (*take-out0*) can be distinguished. The basic idea of *take-out* is simple. The list SET which is always nonempty contains the ELE sign, and the NSET list is identical to the SET list except that the ELE sign is missing. In (2.6) this is spelled out in detail.

(2.6) *Theory of take-out* [17]

$$:\sim take\text{-}out0 \quad \rightarrow \quad \begin{array}{l}(:\text{SET FIRST} \approx :\text{ELE} \\ \wedge\ :\text{SET REST} \approx :\text{NSET})\end{array}$$

as AVM:

$$\begin{bmatrix} take\text{-}out0 \end{bmatrix} \quad \Longrightarrow \quad \begin{bmatrix} take\text{-}out0 \\ \text{SET} \ < \boxed{1} \mid \boxed{2} > \\ \text{ELE} \ \boxed{1} \\ \text{NSET} \ \boxed{2} \end{bmatrix}$$

$$:\sim take\text{-}out1 \quad \rightarrow \quad \begin{array}{l}(\neg\ (:\text{SET FIRST} \approx :\text{ELE}) \\ \wedge\ :\text{SET FIRST} \approx :\text{NSET FIRST} \\ \wedge\ :\text{REC SET} \approx :\text{SET REST} \\ \wedge\ :\text{REC ELE} \approx :\text{ELE} \\ \wedge\ :\text{REC NSET} \approx :\text{NSET REST})\end{array}$$

[17]For reasons of readability we take the liberty to write round brackets instead of square brackets in our notation of descriptions. Furthermore, we omit and add brackets to make the structure of the descriptions more transparent for the reader.

as AVM:

$$\begin{bmatrix} \textit{take-out1} \end{bmatrix} \Longrightarrow \begin{bmatrix} \textit{take-out1} & \\ \text{SET} & < \boxed{1} \mid \boxed{2} > \\ \text{ELE} & \boxed{3} \\ \text{NSET} & < \boxed{1} \mid \boxed{4} > \\ \text{REC} & \begin{bmatrix} \text{SET} & \boxed{2} \\ \text{ELE} & \boxed{3} \\ \text{NSET} & \boxed{4} \end{bmatrix} \end{bmatrix}$$

$$\& \ \boxed{1} \neq \boxed{3}$$

take-out0 is the simple case. If an object is of that species, exactly the first element of the SET list is missing on NSET. For *take-out1* objects the first element on the SET list is identical to the first element on the NSET list, and in addition it is different from the ELE sign. This makes sure that indeed the first element on SET that is identical to ELE is missing on NSET, thus excluding spurious ambiguities in the structure of permutations. The rest of both the SET and the NSET list (which has to be nonempty for both lists) as well as the ELE sign are input to the recursion. In other words, the element on SET that is missing on NSET has still to be found.

As for all theories, it will be proved that for every interpretation in which the denotation of a description of objects of the newly introduced sort is nonempty, the objects have the properties we claim.

PROPOSITION
If u is a *take-out* and $[\text{NSET}]u$ is finite then $[\text{ELE}]u$ is in $[\text{SET}]u$ and $[\text{NSET}]u$ is $[\text{SET}]u$ with the first occurrence of $[\text{ELE}]u$ removed.

PROOF
(by induction on the length of the NSET list)

Base:
For each u such that $\mathrm{S}(u) \sqsubseteq$ *take-out*, if $\mathrm{S}([\text{NSET}]u) =$ *elist*, then $\mathrm{S}(u) =$ *take-out0*, since (by definition) in *take-out1* objects the NSET list contains at least one element. It follows by definition that $[\text{SET REST}]u = [\text{NSET}]u$ $= <>$ and $[\text{SET FIRST}]u = [\text{ELE}]u$. Hence $[\text{ELE}]u$ is in $[\text{SET}]u$ and $[\text{NSET}]u$ is $[\text{SET}]u$ with the first occurrence of $[\text{ELE}]u$ removed.

Hypothesis:
Assume that we have shown for each u with $\mathrm{S}(u) \sqsubseteq$ *take-out* that if $[\text{NSET}]u$ is of length n then the proposition holds for u.

Step:
For each u such that $\mathrm{S}(u) \sqsubseteq$ *take-out*, if the length of $[\text{NSET}]u$ is $n + 1$, then either $\mathrm{S}(u) =$ *take-out0* or $\mathrm{S}(u) =$ *take-out1*.

(a) Assume S(u) = *take-out0*.
By definition [SET FIRST]u = [ELE]u and [SET REST]u = [NSET]u, thus the proposition holds for u.

(b) Assume S(u) = *take-out1*.
Then [NSET FIRST]u = [SET FIRST]$u \neq$ [ELE]u by definition, which means that [ELE]u is not the first element on the SET list. Since [SET REST]u = [REC SET]u and [NSET REST]u = [REC NSET]u with S([REC]u) = S(u') $\sqsubseteq$ *take-out* and the element [ELE]u is passed on by [REC ELE]u = [ELE]u, [REC NSET]u = [NSET]u' is of length n and the hypothesis applies to [NSET]u'. Therefore, the proposition holds for u.

The proposition follows by induction. ■

Exploiting the properties of *take-out*, the theory of *perm* for the permutation of lists of signs can now be defined.

(2.7) *Theory of perm*

$$
:\sim perm1 \quad \rightarrow \quad \begin{array}{l}
(\text{:ORIGIN} \approx \text{:TEST SET} \\
\wedge\ \text{:PERMUTED FIRST} \approx \text{:TEST ELE} \\
\wedge\ \text{:REC PERMUTED} \approx \text{:PERMUTED REST} \\
\wedge\ \text{:REC ORIGIN} \approx \text{:TEST NSET})
\end{array}
$$

as AVM:

$$
\begin{bmatrix} \textit{perm1} \end{bmatrix} \quad \Longrightarrow \quad
\begin{bmatrix}
\textit{perm1} & \\
\text{ORIGIN} & \boxed{1} \\
\text{PERMUTED} & < \boxed{2} \mid \boxed{3} > \\
\text{TEST} & \begin{bmatrix} \textit{take-out} & \\ \text{SET} & \boxed{1} \\ \text{ELE} & \boxed{2} \\ \text{NSET} & \boxed{4} \end{bmatrix} \\
\text{REC} & \begin{bmatrix} \textit{perm} & \\ \text{ORIGIN} & \boxed{4} \\ \text{PERMUTED} & \boxed{3} \end{bmatrix}
\end{bmatrix}
$$

We prove that for any *perm* object, PERMUTED may contain every possible permutation of elements of the ORIGIN list.

PROPOSITION
For each u, if u is a *perm* and [ORIGIN]u = l is a finite list of signs and [PERMUTED]u = c, then c is an arbitrary permutation of l.

PROOF
(by induction on the length of [ORIGIN]u)

Base:
For each u with $\mathrm{S}(u) \sqsubseteq$ *perm*, if $\mathrm{S}([\text{ORIGIN}]u) =$ *elist*, then $\mathrm{S}(u) =$ *perm0*. Therefore $\mathrm{S}([\text{PERMUTED}]u) =$ *elist*. As both lists are empty the proposition trivially holds for u.

Hypothesis:
Assume that we have shown for each u with $\mathrm{S}(u) \sqsubseteq$ *perm*, if $[\text{ORIGIN}]u = l$ is of length n and $[\text{PERMUTED}]u = c$ then c is an arbitrary permutation of l.

Step:
For each u such that $\mathrm{S}(u) \sqsubseteq$ *perm*, if $[\text{ORIGIN}]u = l'$ is of length $n+1$, then $\mathrm{S}(u) =$ *perm1*. Hence $[\text{TEST SET}]u = [\text{ORIGIN}]u = l'$ and $[\text{TEST ELE}]u = [\text{PERMUTED FIRST}]u = e$. Since $\mathrm{S}([\text{TEST}]u) \sqsubseteq$ *take-out* it follows from the theory of *take-out* that e is on l' and $[\text{TEST NSET}]u = l$ of length n is identical to $[\text{TEST SET}]u = l'$ except for the missing e, which is (by definition of *take-out*) the first occurrence of an arbitrary element on l'.

We also know that $[\text{REC PERMUTED}]u = [\text{PERMUTED REST}]u = c$, where c consists of all elements of $\mathrm{S}([\text{PERMUTED}]u) = c'$ except for e, and $[\text{TEST NSET}]u = [\text{REC ORIGIN}]u = l$ of length n (by the theory of *take-out*). Since $\mathrm{S}([\text{REC}]u) = \mathrm{S}(u') \sqsubseteq$ *perm* the hypothesis applies to $[\text{REC ORIGIN}]u = [\text{ORIGIN}]u'$, and thus c is an arbitrary permutation of l. As c and l differ from c' and l' respectively only by the presence of e on the latter lists and, as we have shown, the first element on c' is identical to the first occurrence of an arbitrary element on l', c' is an arbitrary permutation of l'. Hence the proposition holds for u.

By induction the proposition follows. ■

With the theories of *take-out* and *perm* we have provided an explicit account for licensing a list which is an arbitrary permutation of a given list of signs. It is obvious that a correct permutation relation for lists of other objects can simply be obtained by turning the objects of sort *list-of-signs* in the theory of *perm* and *take-out* and the value of the ELE attribute in the latter theory into the desired sorts.

2.3 Extracting the phonology from signs

In the preceding section we have shown how to build permutations of a list of signs. Now it remains to be shown how the phonological information can be extracted from a list of signs in such a way that the order of the resulting list of PHON values mirrors the order of the signs. This is going to be done by the sort *signphon*.

(2.8) *Appropriateness Conditions for the sort signphon*

signphon	SIGNS	*list-of-signs*
	PHONS	*list-of-phons*

signphon0	SIGNS	*elist*
	PHONS	*elist*
signphon1	SIGNS	*nelist-of-signs*
	PHONS	*list-of-phons*
	APPEND	*append-of-phons*
	REC	*signphon*

In an object of sort *signphon* the PHONS list is the concatenation of the PHON lists of the elements of the SIGNS list.[18] Since the length of the list of signs is unknown in the general case, the concatenation has to be done recursively by using the auxiliary sort *append-of-phons*. The idea of the theory of *signphon* is to go recursively through the list of signs in SIGNS and to append the list of the current sign's phonology to the phonological list in PHONS. Since *append-of-phons* is a part of the theory of *signphon* it has to be defined first. We use the standard definition of an append function as it is presented in (King 1992).

(2.9) *Appropriateness Conditions for the sort append-of-phons*

append-of-phons	RES	*list-of-phons*
	LEFT	*list-of-phons*
	RIGHT	*list-of-phons*

finished-append-of-phons	RES	*list-of-phons*
	LEFT	*elist*
	RIGHT	*list-of-phons*
unfinished-append-of-phons	RES	*list-of-phons*
	LEFT	*nelist-of-phons*
	RIGHT	*list-of-phons*
	REC	*append-of-phons*

finished-append-of-phons and *unfinished-append-of-phons* are species of the supersort *append-of-phons*. *finished-append-of-phons* is the base case with empty LEFT list, *unfinished-append-of-phons* introduces recursion and presupposes a nonempty LEFT list. The theory of *append-of-phons* is straightforward:

[18] A concatenation of two lists l_1 and l_2, where l_2 is concatenated to l_1, yields a new list l which contains exactly all objects on l_1 and l_2 in the same order in which they occur on l_1 and l_2 respectively, and all objects that are on l_1 precede all objects that are on l_2 on l.

(2.10) *Theory of append-of-phons*

$$:\sim\textit{finished-append-of-phons} \quad \rightarrow \quad (:\text{RES} \approx :\text{RIGHT})$$

as AVM:

$$\begin{bmatrix} \textit{finished-append-of-phons} \end{bmatrix} \Longrightarrow \begin{bmatrix} \textit{finished-append-of-phons} & \\ \text{RES} & \boxed{1} \\ \text{LEFT} & \textit{elist} \\ \text{RIGHT} & \boxed{1} \end{bmatrix}$$

$$\begin{array}{lll} :\sim\textit{unfinished-append-of-phons} & \rightarrow & (:\text{RES FIRST} \approx :\text{LEFT FIRST} \\ & & \wedge\ :\text{REC RES} \approx :\text{RES REST} \\ & & \wedge\ :\text{REC LEFT} \approx :\text{LEFT REST} \\ & & \wedge\ :\text{REC RIGHT} \approx :\text{RIGHT}) \end{array}$$

as AVM:

$$\begin{bmatrix} \textit{unfinished-append-of-phons} \end{bmatrix} \Longrightarrow$$

$$\begin{bmatrix} \textit{unfinished-append-of-phons} & \\ \text{RES} & < \boxed{1} \mid \boxed{2} > \\ \text{LEFT} & < \boxed{1} \mid \boxed{3} > \\ \text{RIGHT} & \boxed{4} \\ \text{REC} & \begin{bmatrix} \textit{append-of-phons} & \\ \text{RES} & \boxed{2} \\ \text{LEFT} & \boxed{3} \\ \text{RIGHT} & \boxed{4} \end{bmatrix} \end{bmatrix}$$

If the LEFT list is empty, then the result of the concatenation of LEFT and RIGHT is simply the RIGHT list. In case LEFT is nonempty, then the first element on LEFT is the first element on the resulting list RES. The rest of the resulting list can be constructed by passing the rest of LEFT and the complete RIGHT list on to the respective lists in a recursively embedded *append-of-phons* object. We prove that as a result RES is a concatenation of LEFT and RIGHT.

PROPOSITION
For each u such that $\mathrm{S}(u) \sqsubseteq$ *append-of-phons* and $[\text{LEFT}]u$ is finite, $[\text{RES}]u$ is a concatenation of $[\text{LEFT}]u$ and $[\text{RIGHT}]u$ in that order. We write $[\text{RES}]u = [\text{LEFT}]u\hat{\ }[\text{RIGHT}]u$.

PROOF
(by induction on the length of $[\text{LEFT}]u$)

Base:
For each u such that $\mathrm{S}(u) \sqsubseteq$ *append-of-phons*, if $\mathrm{S}([\text{LEFT}]u) = \textit{elist}$, then

it follows that $\mathrm{S}(u) =$ *finished-append-of-phons*. Then $[\text{RES}]u = [\text{RIGHT}]u$ by definition (2.10). Since $[\text{LEFT}]u$ is the empty list, it follows that $[\text{RES}]u = [\text{LEFT}]u\,\hat{}\,[\text{RIGHT}]u$.

Hypothesis:
Assume that we have shown for each u with $\mathrm{S}(u) \sqsubseteq$ *append-of-phons*, if $[\text{LEFT}]u = \mathit{list}$ and *list* is of length n, then $[\text{RES}]u = [\text{LEFT}]u\,\hat{}\,[\text{RIGHT}]u$.

Step:
For each u such that $\mathrm{S}(u) \sqsubseteq$ *append-of-phons*, if $[\text{LEFT}]u = \mathit{list}'$ is of length $n+1$, then $\mathrm{S}(u) =$ *unfinished-append-of-phons*. It follows by definition that $[\text{RES FIRST}]u = [\text{LEFT FIRST}]u = e$, thus the first element e of $[\text{LEFT}]u$ precedes every other element on the resulting list. Furthermore $[\text{RIGHT}]u = [\text{REC RIGHT}]u$ and $[\text{LEFT REST}]u = [\text{REC LEFT}]u$ of length n. Since $\mathrm{S}([\text{REC}]u) = \mathrm{S}(u') \sqsubseteq$ *append-of-phons*, we know that $[\text{REC LEFT}]u = [\text{LEFT}]u' = \mathit{list}$ (of length n) and consequently $[\text{REC RES}]u = [\text{REC LEFT}]u\,\hat{}\,[\text{REC RIGHT}]u = [\text{LEFT REST}]u\,\hat{}\,[\text{RIGHT}]u$ by hypothesis. Since according to (2.10) $[\text{RES REST}]u = [\text{REC RES}]u$ and, by definition, $[\text{LEFT FIRST}]u = [\text{RES FIRST}]u = e$, it follows that $[\text{RES}]u = [\text{LEFT}]u\,\hat{}\,[\text{RIGHT}]u$ and the proposition holds for u.

The proposition follows by induction. ■

The theory of *append-of-phons* delivers a function for the concatenation of phonological lists. Every sort employed in the appropriateness conditions of *signphon* is now defined and *append-of-phons* can be used in the theory of *signphon*.

(2.11) *Theory of signphon*

$$
\begin{array}{lll}
:\sim \mathit{signphon1} & \rightarrow & (:\text{APPEND RES} \approx :\text{PHONS} \\
 & & \wedge\ \text{APPEND LEFT} \approx :\text{SIGNS FIRST PHON} \\
 & & \wedge\ \text{APPEND RIGHT} \approx :\text{REC PHONS} \\
 & & \wedge\ \text{REC SIGNS} \approx :\text{SIGNS REST})
\end{array}
$$

as AVM:

$$
\begin{bmatrix} \mathit{signphon1} \end{bmatrix}
\Longrightarrow
\begin{bmatrix}
\mathit{signphon1} & \\
\text{SIGNS} & < \begin{bmatrix}\text{PHON}\ \boxed{1}\end{bmatrix} \mid \boxed{2} > \\
\text{PHONS} & \boxed{3} \\
\text{APPEND} & \begin{bmatrix} \mathit{append\text{-}of\text{-}phons} & \\ \text{RES} & \boxed{3} \\ \text{LEFT} & \boxed{1} \\ \text{RIGHT} & \boxed{4} \end{bmatrix} \\
\text{REC} & \begin{bmatrix} \mathit{signphon} & \\ \text{SIGNS} & \boxed{2} \\ \text{PHONS} & \boxed{4} \end{bmatrix}
\end{bmatrix}
$$

In a *signphon1* the phonology of the first sign on SIGNS comes at the beginning of the PHONS list. The rest of the PHONS list is constructed in the recursion, using the rest of the SIGNS list. *append-of-phons* ensures that the phonologies of the embedded *signphon* object succeeds the phonology of the current *signphon*. Thus the order of the phonologies of the signs on PHONS mirrors the order of the signs on SIGNS.

PROPOSITION
For each u with $\mathrm{S}(u) \sqsubseteq$ *signphon* and each list of signs $[\text{SIGNS}]u = < u_1, \ldots, u_m >$ the list of corresponding elements on the PHONS list is $[\text{PHONS}]u = <>$ for $m = 0$ and $[\text{PHONS}]u = [\text{PHON}]u_1\hat{}\ldots\hat{}[\text{PHON}]u_m\hat{}<>$ for $m \geq 1$.

PROOF
(by induction on the length of the $[\text{SIGNS}]u$ list)

Base:
(a) For each u such that $\mathrm{S}(u) =$ *signphon*, if $[\text{SIGNS}]u = <>$, then $\mathrm{S}(u)$ = *signphon0*. Therefore $\mathrm{S}([\text{PHONS}]u) =$ *elist* and the proposition holds for u.

(b) For each u such that $\mathrm{S}(u) =$ *signphon*, if $[\text{SIGNS}]u = < u_1 >$ is of length $m = 1$, then $[\text{APPEND LEFT}]u = [\text{SIGNS FIRST PHON}]u = [\text{PHON}]u_1$ by definition. Since `length`($[\text{SIGNS}]u$) = 1 we know that $[\text{SIGNS REST}]u = <> = [\text{REC SIGNS}]u$. With $\mathrm{S}([\text{REC}]u) \sqsubseteq$ *signphon* and (a) it follows that $[\text{REC PHONS}]u = <> = [\text{APPEND RIGHT}]u$. The theory of *append-of-phons* entails $[\text{APPEND RES}]u = [\text{APPEND LEFT}]u\hat{}[\text{APPEND RIGHT}]u = [\text{PHON}]u_1\hat{}<> = [\text{PHONS}]u$. Hence the proposition holds for u.

Hypothesis:
Assume that we have shown for each u with $\mathrm{S}(u) \sqsubseteq$ *signphon*, if signs $[\text{SIGNS}]u = < u_1, \ldots, u_n >$ is of length $m = n$, then $[\text{PHONS}]u = p_1\hat{}\ldots\hat{}p_n\hat{}<>$, where $p_i = [\text{PHON}]u_i$ for each i with $1 \leq i \leq n$.

Step:
For each u such that $\mathrm{S}(u) \sqsubseteq$ *signphon*, if $m = n+1$, then $\mathrm{S}(u) =$ *signphon1* and $[\text{SIGNS}]u = < u_1, \ldots, u_{n+1} >$. We know from the theory of *signphon* that $[\text{APPEND LEFT}]u = [\text{SIGNS FIRST PHON}]u = [\text{PHON}]u_1 = p_1$ and $[\text{APPEND RES}]u = [\text{PHONS}]u$. From the properties of *append-of-phons* we know that $[\text{APPEND RES}]u = [\text{APPEND LEFT}]u\hat{}[\text{APPEND RIGHT}]u$, from which we conclude that $[\text{APPEND RES}]u = p_1\hat{}[\text{APPEND RIGHT}]u$.

Furthermore the theory of *signphon* says $[\text{REC SIGNS}]u = [\text{SIGNS REST}]u$ and we know that $[\text{SIGNS REST}]u = < u_2, \ldots, u_{n+1} >$, thus $[\text{REC SIGNS}]u = < u_2, \ldots, u_{n+1} >$ is of length n, and we know by definition that $[\text{APPEND RIGHT}]u = [\text{REC PHONS}]u$. Since $\mathrm{S}([\text{REC}]u) = \mathrm{S}(u') \sqsubseteq$ *signphon* it follows by hypothesis that $[\text{REC PHONS}]u = [\text{PHONS}]u' = [\text{PHON}]u_2\hat{}\ldots\hat{}[\text{PHON}]u_{n+1}\hat{}<> = p_2\hat{}\ldots\hat{}p_{n+1}\hat{}<>$, thus $[\text{APPEND RIGHT}]u = p_2\hat{}\ldots\hat{}p_{n+1}\hat{}<>$. This entails $[\text{APPEND RES}]u = p_1\hat{}p_2\hat{}\ldots\hat{}p_{n+1}\hat{}<> = [\text{PHONS}]u$, because $[\text{APPEND RES}]u = p_1\hat{}[\text{APPEND RIGHT}]u$ as shown above. With $[\text{APPEND RES}]u = [\text{PHONS}]u$ the proposition follows for u.

The proposition follows by induction. ■

By means of objects of sort *signphon* we are now in a position to extract the phonologies from a list of signs and concatenate them in an order corresponding to the order of the signs. As it is the case with the definition of sort *perm*, the recursive sort definition of *signphon* can be viewed as an instance of a group of recursive sort definitions. With minor adjustments to the sorts of lists within *append-of-phons* and to the values of the attributes SIGNS and PHONS any recursive sort definition can be constructed that provides a concatenation of lists that are part of a list of objects. Again, this might be a function that can be used in other modules of the grammar.

Some HPSG analyses make use of so-called traces and null relativizers, i.e., signs which have an empty phonology and are used for the analysis of unbounded dependencies, verb movement and relative clauses. It is interesting to note that in our approach to word order these phonologically empty signs are not temporally ordered with respect to those signs in a sentence which possess a nonempty phonology or to any sign in general. This is so, because only the phonological string of a phrase is temporally interpreted, and the empty phonology of a trace does not occur in the PHON list of a sentence. Signs or constituents themselves do not have temporal properties. In this respect our approach differs significantly from GPSG as well as from widespread assumptions in GB theory and the psycholinguistic literature which postulate a temporal localization of phonologically empty elements that is derived from a linear order of sisters in a syntactic tree. Example (2.12) illustrates our treatment of traces.

(2.12) *An example for signphon*

$$
\left[\begin{array}{ll}
\textit{signphon1} & \\
\text{SIGNS} & \left\langle \left[\begin{array}{l}\textit{sign}\\ \text{PHON } \langle a, b\rangle\end{array}\right], \left[\begin{array}{l}\textit{sign}\\ \text{PHON } \langle\rangle\end{array}\right], \left[\begin{array}{l}\textit{sign}\\ \text{PHON } \langle c, d\rangle\end{array}\right]\right\rangle \\
\text{PHONS} & \langle a, b, c, d\rangle \\
\text{APPEND} & \left[\begin{array}{ll}
\text{RES} & \langle a, b, c, d\rangle \\
\text{LEFT} & \langle a, b\rangle \\
\text{RIGHT} & \langle c, d\rangle \\
\text{REC} & \left[\begin{array}{ll}
\text{RES} & \langle b, c, d\rangle \\
\text{LEFT} & \langle b\rangle \\
\text{RIGHT} & \langle c, d\rangle \\
\text{REC} & \left[\begin{array}{ll}
\text{RES} & \langle c, d\rangle \\
\text{LEFT} & \textit{elist} \\
\text{RIGHT} & \langle c, d\rangle
\end{array}\right]
\end{array}\right]
\end{array}\right] \\
\text{REC} & \left[\begin{array}{ll}
\textit{signphon1} & \\
\text{SIGNS} & \left\langle \left[\begin{array}{l}\textit{sign}\\ \text{PHON } \langle\rangle\end{array}\right], \left[\begin{array}{l}\textit{sign}\\ \text{PHON } \langle c, d\rangle\end{array}\right]\right\rangle \\
\text{PHONS} & \langle c, d\rangle \\
\text{APPEND} & \left[\begin{array}{ll}
\text{RES} & \langle c, d\rangle \\
\text{LEFT} & \textit{elist} \\
\text{RIGHT} & \langle c, d\rangle
\end{array}\right] \\
\text{REC} & \left[\begin{array}{ll}
\textit{signphon1} & \\
\text{SIGNS} & \left\langle \left[\begin{array}{l}\textit{sign}\\ \text{PHON } \langle c, d\rangle\end{array}\right]\right\rangle \\
\text{PHONS} & \langle c, d\rangle \\
\text{APPEND} & \left[\begin{array}{ll}
\text{RES} & \langle c, d\rangle \\
\text{LEFT} & \langle c, d\rangle \\
\text{RIGHT} & \textit{elist} \\
\text{REC} & \left[\begin{array}{ll}
\text{RES} & \langle d\rangle \\
\text{LEFT} & \langle d\rangle \\
\text{RIGHT} & \textit{elist} \\
\text{REC} & \left[\begin{array}{ll}
\text{RES} & \textit{elist} \\
\text{LEFT} & \textit{elist} \\
\text{RIGHT} & \textit{elist}
\end{array}\right]
\end{array}\right]
\end{array}\right] \\
\text{REC} & \left[\begin{array}{ll}
\textit{signphon0} & \\
\text{SIGNS} & \textit{elist} \\
\text{PHONS} & \textit{elist}
\end{array}\right]
\end{array}\right]
\end{array}\right]
\end{array}\right]
$$

In order to make the example more readable, tags are omitted and instead of phonologies of actual words letters are used as phonological values. The second sign on the SIGNS list of the outer *signphon1* object is phonologically empty and could therefore be a trace in an actual grammar. The phonology of the trace is treated in the first recursion step of the *signphon* objects and leads to an empty list for REC APPEND LEFT. By the properties of *append-of-phons* it follows that REC APPEND RES, which equals REC PHONS, consists only of the elements on REC APPEND RIGHT. As a consequence, traces do not enter into a temporal relationship in a phrase.[19]

[19] The indifference of traces in HPSG with respect to temporal order is shortly discussed in (Pollard and Moshier 1990, pp.291-292).

2.4 The theory of *build-phon*

With all components in place, it is simple to formulate the theory of *build-phon* we were heading for throughout this chapter. The theory of *build-phon* conjoins the recursive sort definitions of *perm* and *signphon* in a new object, thereby delivering a list of concatenated phonologies for every permutation of a given list of signs.

(2.13) *Theory of build-phon*

$$:\sim \textit{build-phon} \quad \rightarrow \quad (:\text{PERM PERMUTED} \approx :\text{SIGNPHON SIGNS})$$

as AVM:

$$\begin{bmatrix} \textit{build-phon} \end{bmatrix} \Longrightarrow \begin{bmatrix} \textit{build-phon} & \\ \text{PERM} & \begin{bmatrix} \textit{perm} & \\ \text{ORIGIN} & \textit{list-of-signs} \\ \text{PERMUTED} & \boxed{1} \end{bmatrix} \\ \text{SIGNPHON} & \begin{bmatrix} \textit{signphon} & \\ \text{SIGNS} & \boxed{1} \\ \text{PHONS} & \textit{list-of-phons} \end{bmatrix} \end{bmatrix}$$

PROPOSITION
If u is a *build-phon* and [PERM ORIGIN]$u = l$ is finite then [SIGNPHON PHONS]u is an arbitrary permutation of the PHON values of the signs on l, if there are any. For an empty list l, the resulting list is empty.

As the theory of *build-phon* is not recursive but only makes use of recursively defined objects which are part of it, the proof of the properties of *build-phon* objects can be obtained by using the previously shown properties of the embedded objects.

PROOF

The theory of *perm* entails that for every u with $S(u) = \textit{build-phon}$ and every finite [PERM ORIGIN]$u = l$, [PERM PERMUTED]$u = c$ is an arbitrary permutation of l, since $S([\text{PERM}]u) \sqsubseteq \textit{perm}$. Because [PERM PERMUTED]u = [SIGNPHON SIGNS]u by definition of *build-phon*, it follows by the theory of *signphon* that [SIGNPHON SIGNS]u contains an arbitrary permutation of the PHON values of l if there are signs on l, otherwise [SIGNPHON SIGNS]$u = <>$.

■

In this chapter a new attribute BUILD-PHON for phrases was proposed that serves to provide a list of permutations of the PHON lists of a given list of signs. It was shown how this concept can be formalized within the description language of HPSG as it is informally suggested in (Pollard and Sag 1994) and rigorously defined in the logic of section 1.2. The theory of *build-phon* illustrated that we do not need a logic more powerful than the one provided by King (1989) to formalize nontrivial principles of grammar that refer to recursive structures. Moreover, the necessary constructions are fairly perspicuous and should be easy to handle for any linguist.

With an appropriate theory of phrases the theory of *build-phon* supplies the means to constrain the PHON list of phrases to a permutation of the PHON lists of its constituents (or even words). As we have discussed in the beginning, this was the underlying idea of the informal proposal by Pollard and Sag (1987), who suggested to use the definite relations `order-constituents` and `interleave-constituents`.

As the theory of *build-phon* imposes a universal constraint on PHON lists of phrases, it remains to be shown how language specific LP rules can be added to the presented setup to further constrain word order, possibly in certain domains. After a brief discussion of our syntactic assumptions about the German mittelfeld and some remarks on the consequences of adopting differing analyses, LP principles will be the topic of the remaining chapters.

3 Constituent Structures

Although in this thesis we are not concerned with the question what an adequate analysis of the German mittelfeld should look like, we are forced to adopt a working hypothesis about its structure in order to supply a foundation for the LP module of the grammar. It depends on the syntactic structure what kind of interfaces are necessary for mediating between the component containing the LP principles which specify the temporal order of phonological items of the sentence and components encoding other properties of that sentence. In section 3.1 we sketch an analysis of German sentence structure that is extremely simplistic and presumably not apposite for a fragment that covers a broader range of data. Its main idea is that all constituents of the mittelfeld and all verbs of the verbal complex are sisters of each other, which results in a completely flat structure.

The reason for choosing that particular analysis is that besides its formal simplicity it provides scarcely any structural information as input to the LP principles, thus requiring an elaborate restructuring of information. This gives us ample opportunity to explore the use of recursive sort definitions for the specification of nontrivial relations. A flat analysis of sentences represents the worst case in terms of the number of auxiliary objects that are needed to restructure functional information about constituents that is needed by the LP principles. In general, we assume that the more information about the properties and grammatical function of signs is reflected in the syntactic structure the less information has to be recovered by objects which relationally connect the syntactic structure to the LP principles. Any analysis which postulates more syntactic structure in the mittelfeld than ours should therefore require less complex recursive sort definitions for the connecting objects.

The purpose of the present chapter is threefold. We introduce the analysis of the German mittelfeld that forms the structural basis for the proper working of the ordering principles of the following chapters. Then we discuss the construction of the ALL-DTRS list. ALL-DTRS was informally introduced in the previous chapter as an attribute of *const-struc* objects. As we have pointed out, the value of ALL-DTRS (which is a list of signs) serves as input for *build-phon* objects. The elements of this list are the signs whose phonologies are in principle subject to permutation in a given phrase. Every word or phrase which occurs on ALL-DTRS is in the permutation domain of the phrase, which at the same time is the domain the LP principles apply to.

Lastly, we present an alternative analysis of German sentence structure which we believe is more adequate and use it to shortly demonstrate the modularity of our approach. We show how the ALL-DTRS list can be constructed for a different sentence structure such that *build-phon* produces exactly the same permutations for the constituents in the mittelfeld as for a flat analysis. This result proves our claim that LP principles in HPSG are more powerful than LP statements in the ID/LP format of GPSG, since ordering regularities on the phonology are not restricted to the phonologies of sisters in a local tree. Some comments on the consequences of the second analysis for the interfaces to the LP principles and the application of the LP principles themselves conclude this chapter.

3.1 Flattening German

The appropriateness conditions we tacitly presuppose for the fragment of German that is covered by our LP principles are taken from the appendix of (Pollard and Sag 1994). We deviate from the appendix version only by adopting the restructuring of *cat* objects which is explored in chapter 9 of (Pollard and Sag 1994). In that variant of the appropriateness conditions the attribute SUBCAT defined on *cat* it is replaced by VAL which denotes a partial function from objects of sort *cat* to objects of sort *val*. For our purposes we then need the following simplified appropriateness conditions for *val*.

(3.1) *Appropriateness Conditions for the sort val*

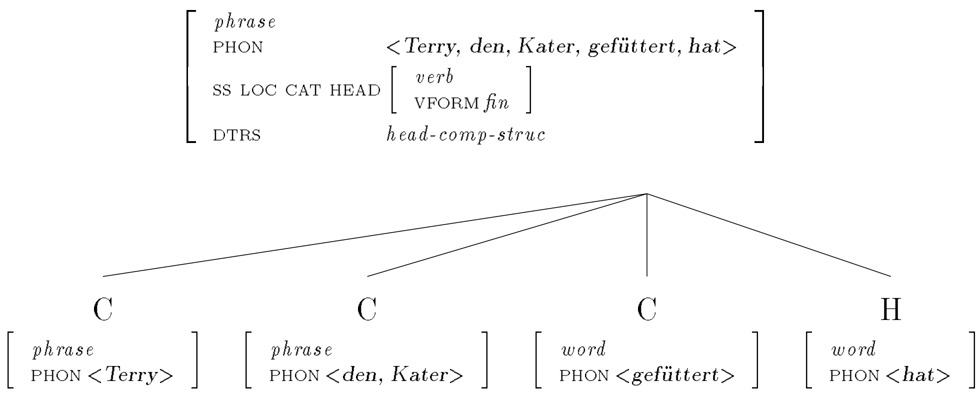
val	SUBJ	*list-of-synsems*
	COMPS	*list-of-synsems*
	APPEND	*append-of-synsems*
	SUCO	*append-of-synsems*

The SUBJ list and the COMPS list replace the traditional SUBCAT list and serve to distinguish subjects from other complements. How this distinction is put to use in HPSG grammars varies. Our own use of the new attributes is explained below in the discussion of coherent constructions. The additional new attributes APPEND and SUCO which introduce recursive sorts will be commented on presently.

The analysis of German sentence structure that is chosen as foundation for the principles of word order postulates that all constituents in the mittelfeld and all verbs of the verbal complex be sisters of each other. Following (Pollard 1990) we further assume that the constituent in the vorfeld in verb second constructions is extracted using the SLASH attribute and the associated principles of (Pollard and Sag 1994). In effect, the constituent in the vorfeld is attached to the rest of the sentence by the head-filler ID schema. Since we are not concerned with vorfeld-phenomena in this thesis we ignore the vorfeld throughout the following discussion. For all examples of the present chapter for which nothing else is said it might be assumed that we are dealing with finite verb final sentences.

As a consequence of the stipulation of a flat mittelfeld, there exists no structural information which could be exploited to determine the grammatical function of the signs in the domain comprising the mittelfeld and the verbal complex. From the structure alone, for example, it is impossible to tell arguments and functors apart. Example (3.2) illustrates this situation for a simple finite sentence.

(3.2) *A flat sentence structure for German*[20]



As it is commonly done in the literature, we depict the DTRS value of a phrase in an extension of the informal AVM notation that utilizes tree-graphics to visualize the daughters structure. The branch labeled H is a notational variant for the attribute HEAD-DTR of *head-comp-struc* and the leaves of the branches labeled C indicate the members of the COMP-DTRS list.

An inspection of the DTRS value in (3.2) is not sufficient to determine that *gefüttert* and *hat* form the verbal complex of the overall phrase and *Terry* and *den Kater* are arguments of *gefüttert*. To find this out, the argument structure of the daughters has to be analyzed. The easiest way to do so is to look at the SUBJ and COMPS list of the verbal daughters and examine the structure sharing of their members with the synsem objects within the nominal daughters of the phrase. Since the potential orderings of the phonologies of the nominal daughters depend in some respects upon the verb they are an argument of, an explicit account of the structure of COMPS lists of verbs is a prerequisite for LP principles to work properly. More specifically, it has to be clarified what the COMPS lists of verbs like *hat* in (3.2) looks like.

We take the standard view of HPSG grammarians that verbs which introduce coherent constructions inherit the arguments of the embedded

[20]For reasons of space we abbreviate the attribute SYNSEM by SS in some AVMs of this chapter.

verbs.[21] Employing a recursive sort definition to formalize argument inheritance, the lexical entry of the verb *haben* looks as presented in (3.3).

(3.3) *The lexical entry of haben*

$$\left[\begin{array}{l}
\textit{word} \\
\text{PHON} <\textbf{\textit{haben}}> \\
\text{SS} \quad \text{LOC CAT} \left[\begin{array}{ll}
\text{HEAD} & \left[\begin{array}{l} \textit{verb} \\ \text{VFORM } \textit{bse} \end{array}\right] \\
\text{VAL} & \left[\begin{array}{ll}
\text{SUBJ} & \boxed{3}\ \textit{list-of-synsems} \\
\text{COMPS} & \boxed{1}\ \textit{list-of-synsems} \\
\text{APPEND} & \left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{append-of-synsems}} \\
\text{RES} & \boxed{1}\ \textit{list-of-synsems} \\
\text{LEFT} & \boxed{2}\ \textit{list-of-synsems} \\
\text{RIGHT} & \left\langle \left[\text{LOC CAT} \left[\begin{array}{ll}
\text{HEAD} & \left[\begin{array}{l} \textit{verb} \\ \text{VFORM } \textit{psp} \end{array}\right] \\
\text{VAL} & \left[\begin{array}{ll} \text{SUBJ} & \boxed{3} \\ \text{COMPS} & \boxed{2} \end{array}\right]
\end{array}\right] \right] \right\rangle
\end{array}\right]
\end{array}\right]
\end{array}\right]
\end{array}\right]$$

Exactly the same idea of argument inheritance is implemented by Nerbonne (1994, p.129), who likewise assumes that verbs like *haben* inherit every argument of the subcategorized verb. Nevertheless (3.3) differs formally from previous renderings of the same idea in the literature. As it is the case for all descriptions in our approach which involve concatenation of lists of arbitrary length, (3.3) utilizes a recursive sort definition for the treatment of argument inheritance. The new recursive sort is called *append-of-synsems* and closely resembles the sort *append-of-phons* that is defined in (2.9) and (2.10) of chapter 2. The only difference consists in the use of the sort *list-of-synsems* where *append-of-phons* employs *list-of-phons*. Since appropriateness conditions and theory of the two sorts are entirely parallel, we will not present *append-of-synsems* but simply suppose that it is defined in the desired way. As a consequence of the use of *append-of-synsems* the appropriateness conditions have to be extended by a new attribute which allows for the accommodation of the recursive sort. This is accomplished by adding the attribute APPEND, which we have already listed in the appropriateness conditions of *val* in (3.1).

Argument inheritance is simply defined by equating the COMPS list of the subcategorized verb whose synsem value is the last element on the COMPS list of *haben* with the APPEND LEFT list of *haben*. As a result,

[21] See (Hinrichs and Nakazawa 1994), (Kiss 1993), (Nerbonne 1994) and (Pollard 1990) for several explications of the concept of argument inheritance in coherent constructions. Some of the verbs in question are often called auxiliaries and modal verbs. As these terms seem to have no explanatory content and their only function is to describe a rather arbitrary collection of verbs with widely differing properties, we refrain from using them.

the COMPS list of *haben* consists of the concatenation of the list of synsem values of the complements of the embedded verb and the singleton list consisting exactly of the synsem value of that verb. Secondly, the entire SUBJ list of the embedded verb and of *haben* are equated. This idea was put forward in (Kiss 1993) and accounts for the fact that *haben* optionally embeds subjectless constructions, i.e., it inherits the subject of the embedded verb in case there is one. Otherwise the empty SUBJ list indicates that there is no subject. Thus subjectless sentences like (3.4) can be captured.

(3.4) *Subjectless construction with haben*

Mich hat ganz schrecklich gefroren.

Finally, care has to be taken of the relationship between the SUBJ list and the COMPS list. For reasons of simplicity no distinction regarding the syntactic argument structure is made between finite and infinite verb forms in contrast to the assumptions made in the majority of the literature.[22] Instead, the *synsem* object on the SUBJ list is uniformly the first element on the COMPS list if a subject exists. The necessary term equations are achieved employing a theory of *val* which refers to the value of the attribute SUCO that was introduced in the appropriateness conditions of *val* in (3.1).

(3.5) *Theory of val*

$$\begin{bmatrix} \textit{val} \end{bmatrix} \Longrightarrow \begin{bmatrix} \textit{val} & \\ \text{SUBJ} & \boxed{1}\ \textit{list-of-synsems} \\ \text{COMPS} & \boxed{2}\ \textit{list-of-synsems} \\ \text{SUCO} & \begin{bmatrix} \textit{append-of-synsems} & \\ \text{RES} & \boxed{2}\ \textit{list-of-synsems} \\ \text{LEFT} & \boxed{1}\ \textit{list-of-synsems} \\ \text{RIGHT} & \textit{list-of-synsems} \end{bmatrix} \end{bmatrix}$$

The theory of *val* guarantees that the *synsem* object on the SUBJ list is always the first element on the COMPS list. Since the relation between the two lists is built into the theory of *val*, it will hold for all objects of species *val* and is not restricted to verbs. It should be noted that this is a difference to argument inheritance which is defined in lexical entries for specific words. This means that although the attribute APPEND is defined on every *val* object, argument inheritance will not occur automatically. For example, we assume that the path SYNSEM LOC CAT VAL APPEND

[22] c.f. (Hinrichs and Nakazawa 1994) for the standard treatment. In this approach the element on the subject list occurs on the COMPS list of finite verbs only.

receives the species *finished-append-of-synsems* in the lexical entry of verbs like *gefüttert*. Hence any kind of argument inheritance is excluded.

In order to describe finite sentences with coherent constructions there are two more grammatical principles needed. These are a revised Subcategorization Principle which respects the new appropriateness conditions and an ID schema that permits phrasal projections for the mittelfeld and the verbal complex as illustrated in (3.2). The new Subcategorization Principle can easily be obtained by slightly modifying the Subcategorization Principle of (Pollard and Sag 1994) that was formalized in example (1.12) above.

(3.6) *The Subcategorization Principle (revised)*

$$\left[\begin{array}{l} \textit{phrase} \\ \text{DTRS}\ \textit{headed-struc} \end{array}\right] \Longrightarrow$$

$$\left[\begin{array}{ll} \textit{phrase} & \\ \text{SYNSEM} & \left[\text{LOC CAT VAL COMPS}\ \boxed{1}\ \textit{list-of-synsems}\right] \\ \text{DTRS} & \left[\begin{array}{ll} \textit{headed-struc} & \\ \text{HEAD-DTR} & \left[\begin{array}{l} \textit{sign} \\ \text{SYNSEM LOC CAT VAL COMPS}\ \boxed{3}\ \textit{list-of-synsems} \end{array}\right] \\ \text{COMP-DTRS} & \boxed{2}\ \textit{list-of-signs} \\ \text{APPEND} & \left[\begin{array}{ll} \textit{sub-append} & \\ \text{LEFT} & \boxed{1}\ \textit{list-of-synsems} \\ \text{RIGHT} & \boxed{2}\ \textit{list-of-signs} \\ \text{RES} & \boxed{3}\ \textit{list-of-synsems} \end{array}\right] \end{array}\right] \end{array}\right]$$

The only difference to the original Subcategorization Principle consists in the replacement of the attribute SUBCAT by the term VAL COMPS in accordance with the appropriateness conditions. The correct working of (3.6) is ensured by structure sharing of the appropriate lists with the attributes of the *sub-append* object whose recursive definition guarantees that the COMPS list of the head daughter is a concatenation of the COMPS list of the entire phrase and the list of synsem values of the list of complement daughters in the order of the appearance of the complement daughters on the COMP-DTRS list.

Besides the Subcategorization Principle the ID Principle governs the constituent structure of sentences. In the appendix of (Pollard and Sag 1994, p.399) the consequent of the ID Principle is a disjunction of six ID schemata whose antecedent is a phrase with DTRS value *headed-struc*.[23] Abbreviating the disjunctive descriptions of the consequent by names, the ID Principle of Pollard and Sag (1994) has the following logical structure.

[23]The explanation of the ID Principle in the text (Pollard and Sag 1994, p.38) is different. There the authors state that the ID Principle itself is a disjunction of ID schemata. In our treatment of the ID schemata we follow the version given in the appendix.

(3.7) *The logical structure of the ID Principle*

$$(:\sim \textit{phrase} \wedge :\text{DTRS} \sim \textit{headed-struc}) \quad \rightarrow \quad \begin{array}{l} (\text{SCHEMA } 1 \\ \vee \ \text{SCHEMA } 2 \\ \vee \ \text{SCHEMA } 3 \\ \vee \ \text{SCHEMA } 4 \\ \vee \ \text{SCHEMA } 5 \\ \vee \ \text{SCHEMA } 6) \end{array}$$

When we speak of ID schemata, we mean the disjuncts in the consequent of an implicational constraint with the antecedent given in (3.7). Because we envisage a completely flat structure for the mittelfeld and verbal complex and ignore the vorfeld, a single ID schema can provide all that is needed in our context.

(3.8) *ID schema for finite sentences*

$$\left[\begin{array}{ll} \textit{phrase} & \\ \text{SYNSEM LOC CAT VAL COMPS} & \textit{elist} \\ \text{DTRS} & \left[\begin{array}{ll} \textit{head-comp-struc} & \\ \text{HEAD-DTR} & \left[\begin{array}{ll} \textit{word} & \\ \text{SYNSEM LOC CAT} & \left[\begin{array}{ll} \textit{cat} & \\ \text{HEAD VFORM} & \textit{fin} \\ \text{VAL COMPS} & \boxed{2} \end{array}\right] \end{array}\right] \\ \text{COMP-DTRS} & \boxed{1}\ \textit{nelist-of-signs} \\ \text{APPEND} & \left[\begin{array}{ll} \textit{part-fin-append} & \\ \text{RES} & \boxed{2} \\ \text{RIGHT} & \boxed{1} \end{array}\right] \end{array}\right] \end{array}\right]$$

(3.8) corresponds to ID schemata for German suggested by Pollard (1990, p.6) and Nerbonne (1994, p.126). It restricts admissible phrases which are headed phrases to those whose head daughter is a finite verb and whose complements are discharged in one step. The latter condition is enforced by the requirement that the COMPS list of the resulting phrase be empty.

As the COMPS list of the phrase has to be empty, only the subsort *part-fin-append* of *sub-append* is needed as value of DTRS APPEND. By definition its LEFT list is empty and it simply relates the signs on the COMP-DTRS list to the synsems on the head daughter's COMPS list. A look at the Subcategorization Principle (3.6) shows that this requirement of the ID schema (3.8) is in fact redundant because it is already part of the Subcategorization Principle. Hence we can simplify (3.8) and arrive at the more elegant version (3.9) which is in fact nearly identical to ID schema 3 of (Pollard and Sag 1994, p.402).

(3.9) *ID schema for finite sentences (revised)*

$$\left[\begin{array}{ll} \multicolumn{2}{l}{\textit{phrase}} \\ \multicolumn{2}{l}{\text{SYNSEM LOC CAT VAL COMPS } \textit{elist}} \\ \text{DTRS HEAD-DTR} & \left[\begin{array}{ll} \multicolumn{2}{l}{\textit{word}} \\ \text{SYNSEM LOC CAT} & \left[\begin{array}{ll} \multicolumn{2}{l}{\textit{cat}} \\ \text{HEAD VFORM} & \textit{fin} \\ \text{VAL COMPS} & \textit{nelist-of-synsems} \end{array}\right] \end{array}\right] \end{array}\right]$$

The simplified version of the ID schema for finite sentences makes only reference to the head daughter of the construction and to the COMPS list of the overall phrase. The head daughter is required to be a finite verb with a nonempty COMPS list.

Demanding the COMPS list of the head daughter to be nonempty prevents projections from words to phrases in cases where the potential head of the phrase has no complements. A case in point are verb second sentences in German with an intransitive verb whose subject is in the vorfeld.

(3.10) *A verb second sentence with intransitive verb*

Terry lacht.

Adopting the traceless analysis of extraction of chapter 9 of (Pollard and Sag 1994), without the requirement of a nonempty COMPS list in schema (3.9) sentence (3.10) could either be formed by projecting *lacht* to phrase level and then combining it with the constituent *Terry* in the vorfeld via the head filler schema, or *lacht* could directly be combined with the constituent in the vorfeld. (3.9) excludes the first option.

Demanding an empty COMPS list for the overall phrase guarantees that the arguments of the finite verb are discharged in one step. The relationship between the synsem objects on the finite verb's COMPS list and the complement daughters of the phrase is governed by the Subcategorization Principle alone.

The rather simple principles presented so far are already sufficient to capture the flat description of finite sentences illustrated in example (3.2). Before we finally introduce the construction of the ALL-DTRS list for this sentence structure, let us take a closer look at the content of the COMPS lists of the verbs in a somewhat complex sentence. The understanding of their structure provides the necessary background to the understanding of the interface we will need for the LP principles in the next chapter. The COMPS lists of the verbs are the crucial source of information for some of the recursive sorts connecting the syntax to the LP principles.

(3.11) *An example for argument inheritance*[24]

$$\begin{bmatrix} \textit{phrase} \\ \text{PHON} < \textit{Terry, dem, Kind, das, Buch, hat, geben, dürfen} > \\ \text{DTRS } \textit{head-comp-struc} \end{bmatrix}$$

$$\begin{array}{cccccc} \text{C} & \text{C} & \text{C} & \text{H} & \text{C} & \text{C} \\ \begin{bmatrix} \text{P} < \textit{Terry} > \\ \text{SS } \boxed{1} \end{bmatrix} & \begin{bmatrix} \text{P} < \textit{dem, Kind} > \\ \text{SS } \boxed{2} \end{bmatrix} & \begin{bmatrix} \text{P} < \textit{das, Buch} > \\ \text{SS } \boxed{3} \end{bmatrix} & \begin{bmatrix} \text{P} < \textit{hat} > \\ \text{SS } \boxed{6} \end{bmatrix} & \begin{bmatrix} \text{P} < \textit{geben} > \\ \text{SS } \boxed{4} \end{bmatrix} & \begin{bmatrix} \text{P} < \textit{dürfen} > \\ \text{SS } \boxed{5} \end{bmatrix} \end{array}$$

Supposing that *dürfen* is a verb for which argument inheritance is defined in the same way as for *haben* in (3.3) we arrive at the COMPS lists in (3.12). The tags in the AVM (3.11) are meant to refer to the tags in (3.12), i.e., the AVMs for the verbal signs below are simply more explicit elaborations on the verbal daughters in the AVM above.

(3.12) *Argument inheritance in the verbal complex*

$$\begin{bmatrix} \textit{word} \\ \text{PHON} \quad < \textit{geben} > \\ \text{SYNSEM } \boxed{4} \text{ CAT VAL COMPS} < \boxed{1}, \boxed{2}, \boxed{3} > \end{bmatrix}$$

$$\begin{bmatrix} \textit{word} \\ \text{PHON} \quad < \textit{dürfen} > \\ \text{SYNSEM } \boxed{5} \text{ CAT VAL COMPS} < \boxed{1}, \boxed{2}, \boxed{3}, \boxed{4} > \end{bmatrix}$$

$$\begin{bmatrix} \textit{word} \\ \text{PHON} \quad < \textit{hat} > \\ \text{SYNSEM } \boxed{6} \text{ CAT VAL COMPS} < \boxed{1}, \boxed{2}, \boxed{3}, \boxed{4}, \boxed{5} > \end{bmatrix}$$

(3.12) illustrates the steady growth of COMPS lists in the verbal complex starting from the main verb of the sentence *geben* up to the finite verb *hat*. Each verb that governs one more verb adds the synsem value of the governed verb to the end of a list containing the synsem values of the governed verb's arguments. On the basis of the three COMPS lists in (3.12) it can easily be figured out which arguments stem from which verb by calculating the difference between a governing verb's COMPS list and the COMPS list of the governed verb. This property of COMPS lists in the verbal complex will turn out to be useful for our analysis.

The definition of a theory of *head-comp-struc* that delivers the desired constraint for the ALL-DTRS list is in fact trivial. Since the mittelfeld

[24] For reasons of space the attribute PHON is abbreviated by P in the descriptions of the daughters in example (3.11).

has a flat structure, all constituents and the infinite verbs of the verbal complex are directly accessible in the COMP-DTRS list. The finite verb is head daughter. The appropriateness conditions look as follows:

(3.13) *Appropriateness Conditions for the sort head-comp-struc*

head-comp-struc	HEAD-DTR	*sign*
	COMP-DTRS	*list-of-phrases*
	ALL-DTRS	*list-of-signs*
	APPEND	*sub-append*

The attribute APPEND plays no role in the construction of the ALL-DTRS list but is needed for the formalization of the Subcategorization Principle. To collect all daughters of a head complement structure in a single list one implicational constraint suffices.

(3.14) *Theory of head-comp-struc*

$$:\sim \textit{head-comp-struc} \;\rightarrow\; (\text{:ALL-DTRS FIRST} \approx \text{:HEAD-DTR} \wedge \text{:ALL-DTRS REST} \approx \text{:COMP-DTRS})$$

as AVM:

$$\begin{bmatrix}\textit{head-comp-struc}\end{bmatrix} \Longrightarrow \begin{bmatrix}\textit{head-comp-struc} & \\ \text{HEAD-DTR} & \boxed{1} \\ \text{COMP-DTRS} & \boxed{2} \\ \text{ALL-DTRS} & < \boxed{1} \mid \boxed{2} > \end{bmatrix}$$

It follows trivially from the definition of *head-comp-struc* that every sign that is either on the COMP-DTRS list or equal to the head daughter is also on the ALL-DTRS list. Clearly, the theory of *build-phon* in combination with the theory of *head-comp-struc*, when connected in a theory of *phrase*, delivers every permutation of the constituents in the mittelfeld. Even more, the verbs of coherent constructions are also included in these permutations. The latter fact is of no impact for the theory of constituent order in the mittelfeld because LP principles make statements about the relative order of the phonologies of signs with certain intrinsic or functional properties. For this reason principles which concern the phonologies of signs in the mittelfeld are blind with respect to signs in the verbal complex which obey other principles. In the same vein, the relative order of the verbs and the constituents in the mittelfeld is governed by an extra LP principle.

We have now laid the syntactic foundation for the treatment of word order phenomena in the mittelfeld. We have seen that for a flat structure of the mittelfeld it is easy to define a permutation domain which comprises the envisaged constituents. In the next section we show how the same permutations can be licensed for a binary branching sentence structure.

3.2 Structuring German

In this section we will define an alternative analysis of German sentence structure and investigate its impact on the formulation of the interfaces to the LP module. As starting point we choose the theory of sentence structure and verbal complex developed in (Hinrichs and Nakazawa 1994) and furnish it with a binary branching mittelfeld.[25] As theories of German sentence structure are not topic of the present thesis we will not go into a discussion of the virtues or problems of such a theory in the light of a broader range of data. The purpose of this section is to show how current syntactic assumptions fit into our concept of LP. It will become clear in the course of our discussion that nothing crucial hinges upon the simplistic analysis we have outlined in section 3.1. As in the preceding section, we abstract away from the vorfeld and verb second phenomena and treat only finite verb final sentences.

After a discussion of the lexical entries and the principles needed for binary branching structures we show how a theory of *head-comp-struc* that licenses the same permutations of constituents in the mittelfeld as the corresponding theory in section 3.1 is formalized on the new basis.

The Subcategorization Principle (3.6) allowed the realization of an arbitrary number of arguments of the head daughter of a phrase as complement daughters. Whereas it was liberal with respect to the number of arguments, it required that the arguments being realized as complement daughters correspond to a tail of the COMPS list of the head daughter. A binary branching structure is more restrictive in terms of the number of arguments that can be realized. It must only be one at a time. On the other hand, one could imagine that it is allowed to take an arbitrary element from the COMPS list of the head daughter and realize it as complement daughter. This is exactly what the revised Subcategorization Principle in (3.15) does.

(3.15) *Subcategorization Principle for binary branching structures*

$$\begin{bmatrix} \textit{phrase} & \\ \text{DTRS} & \textit{headed-struc} \end{bmatrix} \Longrightarrow$$

[25]In order to preserve consistency in our appropriateness conditions, we tacitly adjust the differing appropriateness conditions of (Hinrichs and Nakazawa 1994) to the appropriateness conditions of (Pollard and Sag 1994). The changes made by Hinrichs and Nakazawa (1994) have no harmful consequences for the construction of the ALL-DTRS list of a phrase as it is done below.

$$\left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{phrase}} \\
\multicolumn{2}{l}{\text{SYNSEM LOC CAT VAL COMPS } \boxed{1}\ \textit{list-of-synsems}} \\
\text{DTRS} & \left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{headed-struc}} \\
\multicolumn{2}{l}{\text{HEAD-DTRS SYNSEM LOC CAT VAL COMPS } \boxed{3}} \\
\text{COMP-DTRS} & \left[\begin{array}{l}
\textit{nelist-of-signs} \\
\text{FIRST SYNSEM } \boxed{2}\ \textit{synsem} \\
\text{REST } \textit{elist}
\end{array}\right] \\
\text{TAKE-OUT} & \left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{take-out-sp}} \\
\text{SET} & \boxed{3}\ \textit{nelist-of-synsems} \\
\text{ELE} & \boxed{2}\ \textit{synsem} \\
\text{NSET} & \boxed{1}\ \textit{list-of-synsems}
\end{array}\right]
\end{array}\right]
\end{array}\right]$$

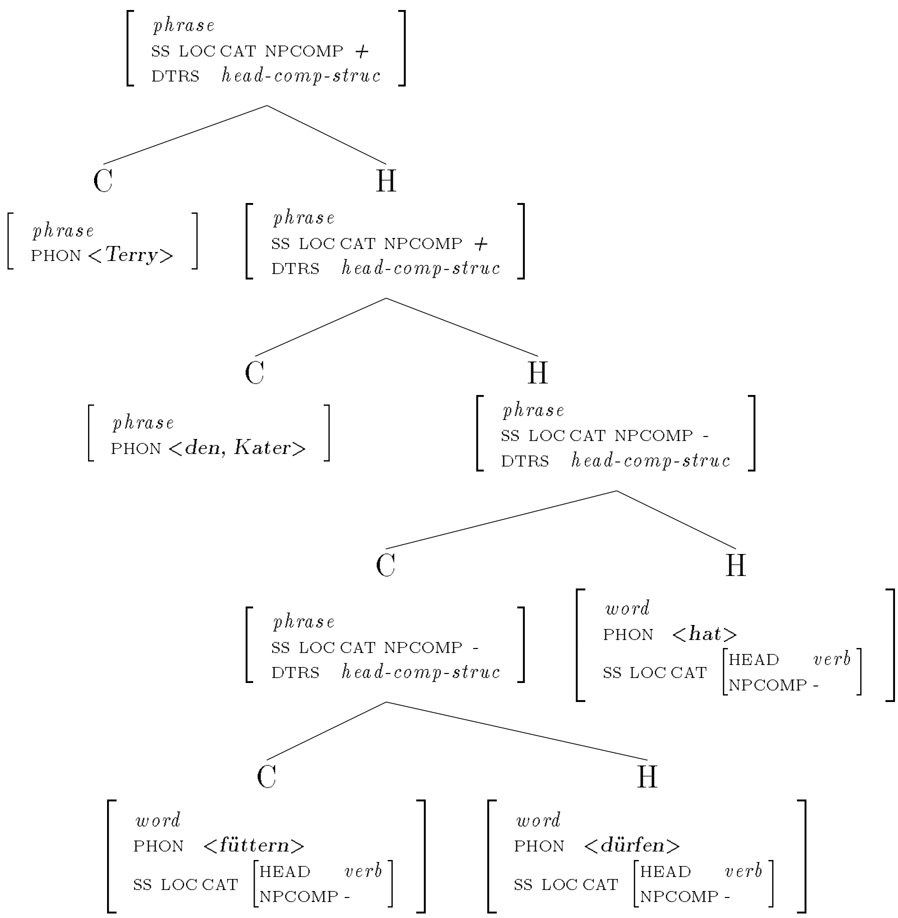
A comparison of the classical Subcategorization Principle in (3.6) with the new Subcategorization Principle reveals only one substantial change. The attribute APPEND, defined on *headed-struc*, has been replaced by TAKE-OUT. The new sort *take-out-sp* is a slight modification of the recursive sort *take-out* which was defined in (2.5) and (2.6) of chapter 2. Instead of removing an arbitrary sign from the SET list *take-out-sp* removes an arbitrary synsem object which is in ELE from a list of synsems in the SET list and specifies the shortened list of synsems in NSET. Since the sorts *take-out* and *take-out-sp* are supposed to be entirely parallel, *take-out-sp* will not be presented explicitly and we simply assume that it is defined in the desired way.

The sort *take-out-sp* in (3.15) is responsible for the binary branching structure of phrases which are headed phrases. It decrees that the COMPS list of the head daughter consist of the COMPS list of the entire phrase plus the SYNSEM value of its single complement daughter. Nothing is said about the position of the synsem object of the complement daughter on the head daughter's COMPS list.

The redefinition of the Subcategorization Principle shows the power and flexibility of recursive sort definitions beyond their use for linearization regularities. A simple modification of an independently motivated sort could be used for the definition of a complex new principle. It should be noted that without changing the value of the attribute COMPS from a list to a set it is now possible to pick out an arbitrary element instead of reducing the list starting from the tail only. This operation on lists increases the possible uses a linguist can make of syntactic sentence structure while preserving the encoding of the obliqueness hierarchy in the COMPS list. For example, it is conceivable to relate scope effects in the mittelfeld to the structural position of complement daughters. This means the structural position of signs in the mittelfeld can be tied to scope properties which in turn interact with the possible orderings of the phonologies of these signs in the sentence. Under this analysis the relative linear order of the phonologies of two signs and their scope relations do not necessarily agree, that is, the sign whose phonology precedes the other sign's phonology does not necessarily have wider scope. Nevertheless, one would expect a strong correspondence of scope and linear order. This correspondence can be expressed by LP principles.

The following AVM gives an overview over the sentence structure that we are going to define.

(3.16) *A binary branching sentence structure*



The use of the attribute NPCOMP is taken over from (Hinrichs and Nakazawa 1994). NPCOMP denotes a partial function from objects of sort *cat* to objects of sort *boolean* which has the two species + and -. The basic idea is that for phrases which dominate signs that are part of the mittelfeld the term SYNSEM LOC CAT NPCOMP receives species + and for phrases which only dominate signs that belong to the verbal complex the term SYNSEM LOC CAT NPCOMP receives species -. As we will see below, the distribution of the species + and - is governed by the lexical entries of verbs which introduce coherent constructions and the ID schemata for headed phrases. In addition, we assume that all verbs are described by SYNSEM LOC CAT NPCOMP $\sim$ -.

The distinction between NPCOMP $\sim$ - and NPCOMP $\sim +$ in combination with the binary branching sentence structure allows to deduce some functional properties of signs in head complement structures from the syntactic structure without examining the COMPS lists of verbs. The complements in the mittelfeld have to be arguments of the verbs. The verb that is in the deepest syntactic position in the verbal complex is the main verb and every head daughter in the verbal complex is a verb that governs another verb. Governed verbs in coherent constructions are either the head of the complement daughter of a governing verb or they are identical to the complement daughter if they do not govern another verb. In the light of this syntactic structure it can be said that functional properties are encoded in a simple recursive structure which may be employed in the construction of objects that mediate between syntax and LP principles.

For a binary branching analysis it is necessary to revise the appropriateness conditions for sort *val* slightly. In order to obtain the desired structure of the verbal complex, one more recursive sort is needed. It occurs as value of the new attribute CHECK.

(3.17) *Appropriateness Conditions for the sort val*

val	SUBJ	*list-of-synsems*
	COMPS	*list-of-synsems*
	APPEND	*append-of-synsems*
	SUCO	*append-of-synsems*
	CHECK	*nonverblist*

The function of the new sort *nonverblist* will be described immediately in the explanation of the revised lexical entry of *haben*.

The definition of lexical entries of coherently constructing verbs is essential for the structuring of verbal complexes as depicted in (3.16). The lexical entry of *haben* in (3.18) illustrates the crucial mechanisms.

(3.18) *The lexical entry of haben*

$$
\begin{bmatrix}
\textit{word} & \\
\text{PHON} \ \langle \textit{haben} \rangle & \\
\text{SS} \quad \text{LOC CAT} & \begin{bmatrix}
\text{HEAD} & \begin{bmatrix} \textit{verb} \\ \text{VFORM } \textit{bse} \end{bmatrix} \\
\text{VAL} & \begin{bmatrix}
\text{SUBJ} & \boxed{3}\ \textit{list-of-synsems} \\
\text{COMPS} & \boxed{1}\ \textit{list-of-synsems} \\
\text{APPEND} & \begin{bmatrix}
\textit{append-of-synsems} & \\
\text{RES} & \boxed{1}\ \textit{list-of-synsems} \\
\text{LEFT} & \boxed{2}\ \textit{list-of-synsems} \\
\text{RIGHT} & \left\langle \begin{bmatrix} \text{LOC CAT} & \begin{bmatrix} \text{HEAD} & \begin{bmatrix} \textit{verb} \\ \text{VFORM } \textit{psp} \end{bmatrix} \\ \text{VAL} & \begin{bmatrix} \text{SUBJ} & \boxed{3} \\ \text{COMPS} & \boxed{2} \end{bmatrix} \\ \text{NPCOMP} & - \end{bmatrix} \end{bmatrix} \right\rangle
\end{bmatrix} \\
\text{CHECK} & \begin{bmatrix} \textit{nonverblist} \\ \text{LIST}\ \boxed{2}\ \textit{list-of-synsems} \end{bmatrix}
\end{bmatrix} \\
\text{NPCOMP} & -
\end{bmatrix}
\end{bmatrix}
$$

Like every verb, *haben* is described by SYNSEM LOCAL CAT NPCOMP ∼ -. It inherits the arguments of its verbal complement (by tag $\boxed{2}$) whose SYNSEM value is on the last position of the COMPS list. Additionally, the verbal complement on the COMPS list is specified for LOC CAT NPCOMP ∼ -. In combination with the ID schemata below this entails that no nonverbal arguments of the verbal complement are already syntactically realized as complement daughters deeper in the structure. In other words, the verbal complex contains verbal complements only. In addition, there is another condition imposed on the inherited arguments. It is specified in the recursive sort *nonverblist* which is appropriate for the new attribute CHECK of *val* objects. It decrees that it is not allowed to inherit verbs from the governed verb's COMPS list. Before we comment on this condition, let us briefly examine its technical definition.

(3.19) *Appropriateness Conditions for the sort nonverblist*

nonverblist		LIST	*list-of-synsems*
	nonverblist1	LIST	*elist*
	nonverblist2	LIST	*nelist-of-synsems*
		REC	*nonverblist*

nonverblist1 is the base case and *nonverblist2* introduces recursion. In *nonverblist2* it is checked that the first element on LIST is not verbal.

(3.20) *Theory of nonverblist*

$$:\sim \textit{nonverblist2} \quad \rightarrow \quad (\neg\ (\text{:LIST FIRST LOC CAT HEAD} \sim \textit{verb}) \wedge \text{:LIST REST} \approx \text{:REC LIST})$$

as AVM:

$$\begin{bmatrix}\textit{nonverblist2}\end{bmatrix} \quad \Longrightarrow \quad \begin{bmatrix}\textit{nonverblist2} & \\ \text{LIST} & < \boxed{1} \mid \boxed{2} > \\ \text{REC LIST} \boxed{2} & \end{bmatrix}$$

$$\& \ \boxed{1} \ [\text{LOC CAT HEAD} \neg \textit{verb}]$$

The theory of *nonverblist2* says that the head of the first element on LIST is not verbal. The rest of the list is passed on to the recursion step. As *nonverblist1* is specified for an empty list only, it is trivial to see that LIST may not contain an element with a verbal head.

With tag $\boxed{2}$ in the lexical entry of *haben* the theory of *nonverblist* prevents verbal elements from being inherited by a coherently constructing verb. This means they have to be syntactically realized before the governed verb can be combined with *haben*. In example (3.16) this mechanism prevents the SYNSEM value of the complement *füttern* of *dürfen* from possibly being raised into the mittelfeld.[26]

Lexical entries for verbs (and words of other categories) which do not inherit arguments take *nonverblist1* as value of the CHECK attribute and do not specify any structure sharing for the empty list LIST.

For the relation between the COMPS list and the SUBJ list we presuppose the theory of *val* defined in (3.1) in the previous section, i.e., the element on the SUBJ list (in case there is one) is the first element on the COMPS LIST. The term equation represented by tag $\boxed{3}$ in the lexical entry (3.18) ensures that the SUBJ list is inherited from the governed verb.

In contrast to the flat analysis of sentence structure in section 3.1 one ID schema is not sufficient for a binary branching analysis. It is necessary to define two ID schemata in order to distinguish between projections in the mittelfeld and the verbal complex.

[26]As it stands, the constraint imposed by the theory of *nonverblist* is too strong since it bars sentential and VP complements in the mittelfeld of sentences with a verbal complex. We will not attempt to solve the problem and instead simply assume with Hinrichs and Nakazawa (1994) that verbs which take sentential or VP complements are not part of the present fragment of German.

(3.21) *ID schema for the mittelfeld*

$$\begin{bmatrix} \textit{phrase} & \\ \text{SYNSEM LOC CAT NPCOMP } + & \\ \text{DTRS} & \begin{bmatrix} \textit{head-comp-struc} \\ \text{HEAD-DTR SYNSEM LOC CAT HEAD } \textit{verb} \\ \text{COMP-DTRS FIRST SYNSEM LOC CAT HEAD } \neg\ \textit{verb} \end{bmatrix} \end{bmatrix}$$

The ID schema for the mittelfeld describes a phrase whose head is verbal and which is described by SYNSEM LOC CAT NPCOMP ∼ +. The first specification ensures that the phrase at hand is a verbal projection. The second condition makes the ID schema for the mittelfeld incompatible with phrases that are a complement of coherently constructing verbs. This follows from the lexical entry of coherently constructing verbs as exemplified by *haben* in (3.18) which describes its complement as LOC CAT NPCOMP ∼ -. The requirement that the complement daughter be not verbal aims at excluding an application of (3.21) to the phrase which is the mother of a finite verb that constructs coherently. For example, the mother of *hat* in (3.16) can not be described by (3.21) because the complement of *hat* has a verbal head. If the condition on complement daughters was missing, the mother of the finite verb in (3.16) could be SYNSEM LOC CAT NPCOMP ∼ +, although it dominates no sign of the mittelfeld. It should be noted that the condition on the complement daughter makes use of the precondition that verbs with VP complements are not part of the present fragment of German.

By means of the Subcategorization Principle (3.15) it is automatically ensured that the head daughter's COMPS list is nonempty and there are no projections from words to phrases in which no complement daughter is realized. Moreover, we know that one of the elements on the head daughter's COMPS list is identical with the SYNSEM value of the complement daughter, and the COMPS list of the entire phrase is the COMPS list of the head daughter with the value of ELE removed.

Next we need an ID schema for the verbal complex. It should be recalled at this point that the consequent of the ID principle is a disjunction of the ID schemata as illustrated in (3.7). For the present grammar this means that a phrase which is a headed phrase is either in the denotation of (3.21) or in the denotation of (3.22).

(3.22) *ID schema for the verbal complex*

$$
\begin{bmatrix}
\textit{phrase} & \\
\text{SYNSEM LOC CAT NPCOMP } \text{-} & \\
\text{DTRS} & \begin{bmatrix}
\textit{head-comp-struc} & \\
\text{HEAD-DTR} & \begin{bmatrix}
\textit{word} & \\
\text{SS LOC CAT} & \begin{bmatrix}
\text{HEAD } \textit{verb} & \\
\text{VAL} & \begin{bmatrix}
\text{COMPS} & \boxed{3}\ \textit{list-of-synsems} \\
\text{APPEND} & \begin{bmatrix}
\textit{append-of-synsems} & \\
\text{RES} & \boxed{3} \\
\text{LEFT} & \textit{list-of-synsems} \\
\text{RIGHT} & < \boxed{2} >
\end{bmatrix}
\end{bmatrix}
\end{bmatrix}
\end{bmatrix} \\
\text{COMP-DTRS FIRST SYNSEM} & \boxed{2}\ \textit{synsem}
\end{bmatrix}
\end{bmatrix}
$$

The ID schema for the verbal complex describes a phrase which is in the denotation of SYNSEM LOC CAT NPCOMP $\sim$ -. Therefore it is compatible with phrasal complements of coherently constructing verbs like the complement of *hat* in (3.16). Furthermore (3.22) requires that the head daughter be a verbal word. It follows immediately that (3.22) can never describe a phrase above the first projection of the mittelfeld, i.e., above the phrase dominating the finite verb in verb final sentences without verbal complex or above the highest phrase dominating only the verbal complex otherwise.

There remains one additional case to be excluded, though. It occurs when there is no verbal complex but only a finite verb in the sentence. According to what has been said so far, the first projection in the mittelfeld could then still be in the denotation of SYNSEM LOC CAT NPCOMP $\sim$ -, since the head daughter is in fact a verbal word. It is the value of the attribute APPEND which serves to define argument inheritance for coherently constructing verbs that prevents the application of (3.22). As we have noted above, usually verbs do not allow argument inheritance except for coherently constructing verbs. Therefore their attribute APPEND receives species *finished-append-of-synsems* in the lexical entry. But this description is not compatible with the description of the head daughter in (3.22) which presupposes a coherently constructing verb with argument inheritance. Thus, if a phrase is in the denotation of (3.22) then we know that its head daughter is a coherently constructing verb, the complement daughter is its verbal complement and the phrase dominates the verbal complex only.

By tag $\boxed{2}$ the ID schema for the verbal complex ensures that the complement daughter is always the complement of the coherently constructing verb. It is not possible to realize an arbitrary element on the COMPS list of the head daughter as complement daughter.

With the definition of the verbal complex ID schema we have completed the specification of a grammar fragment for German that is based on a binary branching sentence structure. We can now turn to the specification of an ALL-DTRS list for the new analysis that will provide an input to the sort *build-phon* such that the same permutations of constituents is specified for the mittelfeld as it was the case for the flat analysis of 3.1.

In order to illustrate the choices a grammar writer faces when she writes a theory for the construction of ALL-DTRS lists, we proceed in two steps. In a first step a theory of *head-comp-struc* is defined that corresponds to the theory of *head-comp-struc* that was formalized in section 3.1 to capture the ALL-DTRS list in a flat analysis. It is noteworthy that such a theory can only refer to the NPCOMP value of the daughters of a phrase, since in the present architecture of phrases the value of the phrase's own NPCOMP attribute is not accessible for a theory of its *head-comp-struc* object.

In the theory of the construction of the ALL-DTRS list we discuss first the verbal complex and the mittelfeld will be treated differently. The constituents of the mittelfeld will be represented as separate signs on the ALL-DTRS list of the lowest phrase dominating the entire mittelfeld, whereas the verbal complex is represented by a single sign. This entails that in sentences the phonology of a constituent in the mittelfeld can not occur between phonological elements of the verbal complex.

It will then be shown that the discrimination of mittelfeld and verbal complex yields an ALL-DTRS list that is too restrictive for the treatment of certain data. In a second step a new theory is developed. It implies that the constituents of the mittelfeld as well as the words of the verbal complex are separately represented on the ALL-DTRS list of the lowest phrase dominating the entire mittelfeld. The new theory will be formalized as a partial theory of *phrase*. This allows to refer to the NPCOMP value of the current phrase and not only to the NPCOMP value of the daughters. Therefore, the new theory can make more intuitive use of the distinction between the NPCOMP value + for phrases dominating constituents of the mittelfeld and the NPCOMP value - for phrases only dominating (parts of) the verbal complex.

We begin with the appropriateness conditions for the sort *head-comp-struc*. They hold for both ways of constructing the ALL-DTRS list we are going to present.

(3.23) *Appropriateness Conditions for the sort head-comp-struc*

head-comp-struc	HEAD-DTR	*sign*
	COMP-DTRS	*list-of-phrases*
	ALL-DTRS	*list-of-signs*
	TAKE-OUT	*take-out-sp*

The attribute TAKE-OUT which replaces the attribute APPEND of the flat analysis is used in the definition of the Subcategorization Principle. It is not relevant for collecting the daughters of a phrase in the ALL-DTRS list.

For the construction of the ALL-DTRS list two implicational constraints are needed. The first one concerns the specification of the ALL-DTRS list for phrases which are the mother of a phrase that dominates signs of the mittelfeld and the second one applies to phrases only dominating either the finite verb (in case there is no verbal complex) or (parts of) the verbal complex (in case there is one).

(3.24) *Theory of head-comp-struc*

$$\begin{array}{l} (:\sim\textit{head-comp-struc} \\ \wedge \text{ :HEAD-DTR SYNSEM LOC CAT NPCOMP} \sim +) \quad \rightarrow \\ \qquad (\text{:ALL-DTRS FIRST} \approx \text{:COMP-DTRS FIRST} \\ \qquad \wedge \text{ :ALL-DTRS REST} \approx \text{:HEAD-DTR DTRS ALL-DTRS}) \end{array}$$

as AVM:

$$\begin{bmatrix} \textit{head-comp-struc} \\ \text{HEAD-DTR SYNSEM LOC CAT NPCOMP } + \end{bmatrix} \Longrightarrow$$

$$\begin{bmatrix} \textit{head-comp-struc} & \\ \text{HEAD-DTR} & \text{DTRS ALL-DTRS } \boxed{2}\ \textit{list-of-signs} \\ \text{COMP-DTRS} & < \boxed{1}\ \textit{sign} \mid \textit{list-of-signs} > \\ \text{ALL-DTRS} & < \boxed{1}\ \textit{sign} \mid \boxed{2}\ \textit{list-of-signs} > \end{bmatrix}$$

$$\begin{array}{l} (:\sim\textit{head-comp-struc} \\ \wedge \text{ :HEAD-DTR SYNSEM LOC CAT NPCOMP} \sim \text{-}) \quad \rightarrow \\ \qquad\qquad (\text{:ALL-DTRS FIRST} \approx \text{:HEAD-DTR} \\ \qquad\qquad \wedge \text{ :ALL-DTRS REST} \approx \text{:COMP-DTRS}) \end{array}$$

as AVM:

$$\begin{bmatrix} \textit{head-comp-struc} \\ \text{HEAD-DTR SYNSEM LOC CAT NPCOMP -} \end{bmatrix} \Longrightarrow$$

$$\begin{bmatrix} \textit{head-comp-struc} & \\ \text{HEAD-DTR} & \boxed{1}\ \textit{sign} \\ \text{COMP-DTRS} & \boxed{2}\ \textit{list-of-signs} \\ \text{ALL-DTRS} & < \boxed{1}\ \textit{sign} \mid \boxed{2}\ \textit{list-of-signs} > \end{bmatrix}$$

The antecedent of the first implication describes a phrase whose head daughter dominates a sign of the mittelfeld. Hence we know that the phrase in question is a verbal projection whose complement daughter is a constituent of the mittelfeld. As all headed phrases are binary branching there is exactly one complement daughter. The consequent requires that the complement daughter be the first sign on the phrase's ALL-DTRS list. Since the head daughter is also a phrase we know that it has an ALL-DTRS list. By the second term equation of the consequent the ALL-DTRS list of the head daughter forms the rest of the current phrase's ALL-DTRS list.

As a result, the first constraint guarantees that all constituents of the mittelfeld whose sister dominates another constituent of the mittelfeld are members of the ALL-DTRS list of the lowest phrase in the sentence dominating the entire mittelfeld and the finite verb or verbal complex respectively.

As far as constituents of the mittelfeld are concerned, there remains only one more case to be clarified. It concerns that constituent of the mittelfeld which is either sister of the finite verb (if there is no verbal complex) or sister of the highest phrase which only dominates signs of the verbal complex. In (3.16) this constituent is the phrase with the phonology *den Kater*. It is treated in the second constraint of (3.24) together with the signs of the verbal complex.

As both verbs and phrases which exclusively dominate signs of the verbal complex are described by SYNSEM LOC CAT NPCOMP $\sim$ -, the lowest verbal projection dominating a phrase of the mittelfeld is described by the antecedent of the second constraint in (3.24). Therefore its single complement daughter is the second sign on that phrase's ALL-DTRS list while the head daughter is its first element.

Since, as we have seen, the ALL-DTRS list is passed on to higher projections every constituent of the mittelfeld is on the ALL-DTRS list of the lowest phrase dominating the entire mittelfeld. Hence a theory of *phrase* connecting the *build-phon* object of chapter 2 to the ALL-DTRS list will supply all permutations of the phonologies of the constituents in the mittelfeld. Thus the LP principles work on the same permutations as for the flat analysis of German sentence structure.

Let us now turn to the treatment of signs within the verbal complex in the theory of *head-comp-struc* in (3.24). According to (3.24) within every projection of the verbal complex the ALL-DTRS list consists of the head daughter and the complement daughter. Thus only the phonologies of these two signs are subject to permutation and LP principles may only order the phonology of the head with respect to the phonology of the entire complement. On the ALL-DTRS list of phrases dominating constituents of the mittelfeld only a single sign containing the phonology of all signs of the verbal complex or the phonology of the finite verb is present, whereas in the case of the flat analysis the sign of every verb was on the overall ALL-DTRS list. While this is of no consequence for the LP principles which refer to the mittelfeld, the theory in (3.24) makes two predictions that turn out to be wrong.

It says that the phonology of signs that do not belong to the verbal complex may not occur between the phonology of signs in the verbal complex, which contradicts the data in (1) and (2) below. And it predicts that the phonology of the head of a phrase in the verbal complex may never appear between the phonology of signs that are dominated by its sister, which contradicts (3).

1. daß sie ihm hat die Leviten lesen wollen.
2. Daß ihr ausgerechnet gestern hat der Reifen platzen müssen!
3. daß Terry das Buch gestern beenden hat können.

According to (3.24) the permutation of the phonologies of verbs in (3), which is an instance of the so-called zwischenstellung, is not licensed since *beenden* and *können* can not be separated by another sign's phonology. They are in a single sign on the ALL-DTRS list of the lowest phrase dominating the verbal complex.

Although the restrictive theory of *head-comp-struc* in (3.24) looks attractive at first sight and can account for standard word order in the verbal complex and simple oberfeldphenomena, it does not provide an ALL-DTRS list that can cope with more intricate data.

The most simple way to circumvent the problem of ending up with an ALL-DTRS list which is not fine-grained enough to license all possible permutations of phonologies in combination with the theory of *build-phon* is to construct an ALL-DTRS list containing all words of the verbal complex in addition to the constituents of the mittelfeld. This is the idea behind the partial theory of *phrase* which we formalize in (3.25). It is meant to replace the theory of *head-comp-struc* in (3.24). The theory of *phrase* in (3.25) is partial because a complete theory of *phrase* comprises other constraints that have nothing to do with the treatment of the ALL-DTRS list.

(3.25) *Theory of phrase (partial)*

$$
\begin{array}{l}
(\ :\sim \mathit{phrase} \\
\wedge\ \text{:DTRS} \sim \mathit{head\text{-}comp\text{-}struc} \\
\wedge\ \text{:SYNSEM LOC CAT NPCOMP} \sim \text{-} \\
\wedge\ \text{:DTRS COMP-DTRS FIRST} \sim \mathit{word}) \quad \rightarrow \\
\qquad\qquad\qquad (\text{:DTRS ALL-DTRS FIRST} \approx \text{:DTRS HEAD-DTR} \\
\qquad\qquad\qquad \wedge\ \text{:DTRS ALL-DTRS REST} \approx \text{:DTRS COMP-DTRS})
\end{array}
$$

as AVM:

$$
\begin{bmatrix}
\mathit{phrase} & \\
\text{SYNSEM LOC CAT NPCOMP -} & \\
\text{DTRS} & \begin{bmatrix} \mathit{head\text{-}comp\text{-}struc} \\ \text{COMP-DTRS FIRST } \mathit{word} \end{bmatrix}
\end{bmatrix}
\Longrightarrow
$$

$$\begin{bmatrix} \textit{phrase} & \\ \text{DTRS} & \begin{bmatrix} \textit{head-comp-struc} & \\ \text{HEAD-DTR} & \boxed{1}\ \textit{sign} \\ \text{COMP-DTRS} & \boxed{2}\ \textit{list-of-signs} \\ \text{ALL-DTRS} & < \boxed{1} \mid \boxed{2} > \end{bmatrix} \end{bmatrix}$$

$$\begin{array}{l} (\ :\sim \textit{phrase} \\ \wedge\ :\text{DTRS} \sim \textit{head-comp-struc} \\ \wedge\ :\text{SYNSEM LOC CAT NPCOMP} \sim \text{-} \\ \wedge\ :\text{DTRS COMP-DTRS FIRST} \sim \textit{phrase}) \quad \rightarrow \\ \quad (:\text{DTRS ALL-DTRS FIRST} \approx :\text{DTRS HEAD-DTR} \\ \quad \wedge\ :\text{DTRS ALL-DTRS REST} \approx :\text{DTRS COMP-DTRS FIRST DTRS ALL-DTRS}) \end{array}$$

as AVM:

$$\begin{bmatrix} \textit{phrase} & \\ \text{SYNSEM LOC CAT NPCOMP} & \text{-} \\ \text{DTRS} & \begin{bmatrix} \textit{head-comp-struc} \\ \text{COMP-DTRS FIRST}\ \textit{phrase} \end{bmatrix} \end{bmatrix} \Longrightarrow$$

$$\begin{bmatrix} \textit{phrase} & \\ \text{DTRS} & \begin{bmatrix} \textit{head-comp-struc} & \\ \text{HEAD-DTR} & \boxed{1}\ \textit{sign} \\ \text{COMP-DTRS FIRST} & \begin{bmatrix} \textit{phrase} \\ \text{DTRS ALL-DTRS}\ \boxed{2}\ \textit{list-of-signs} \end{bmatrix} \\ \text{ALL-DTRS} & < \boxed{1} \mid \boxed{2} > \end{bmatrix} \end{bmatrix}$$

$$\begin{array}{l} (\ :\sim \textit{phrase} \\ \wedge\ :\text{DTRS} \sim \textit{head-comp-struc} \\ \wedge\ :\text{SYNSEM LOC CAT NPCOMP} \sim + \\ \wedge\ :\text{DTRS HEAD-DTR} \sim \textit{word}) \quad \rightarrow \\ \qquad\qquad (:\text{DTRS ALL-DTRS FIRST} \approx :\text{DTRS HEAD-DTR} \\ \qquad\qquad \wedge\ :\text{DTRS ALL-DTRS REST} \approx :\text{DTRS COMP-DTRS}) \end{array}$$

as AVM:

$$\begin{bmatrix} \textit{phrase} & \\ \text{SYNSEM LOC CAT NPCOMP} & + \\ \text{DTRS} & \begin{bmatrix} \textit{head-comp-struc} \\ \text{HEAD-DTR}\ \textit{word} \end{bmatrix} \end{bmatrix} \Longrightarrow$$

$$\begin{bmatrix} \textit{phrase} & \\ \text{DTRS} & \begin{bmatrix} \textit{head-comp-struc} & \\ \text{HEAD-DTR} & \boxed{1}\ \textit{word} \\ \text{COMP-DTRS} & \boxed{2}\ \textit{list-of-signs} \end{bmatrix} \\ \text{ALL-DTRS} < \boxed{1} \mid \boxed{2} > & \end{bmatrix}$$

$$\begin{array}{l} (\ :\sim \textit{phrase} \\ \wedge\ :\text{DTRS} \sim \textit{head-comp-struc} \\ \wedge\ :\text{SYNSEM LOC CAT NPCOMP} \sim + \\ \wedge\ :\text{DTRS HEAD-DTR} \sim \textit{phrase}) \quad \rightarrow \end{array}$$

$$(\text{:DTRS ALL-DTRS FIRST} \approx \text{:DTRS COMP-DTRS FIRST}$$
$$\wedge\ \text{:DTRS ALL-DTRS REST} \approx \text{:DTRS HEAD-DTR DTRS ALL-DTRS})$$

as AVM:

$$\left[\begin{array}{ll} \multicolumn{2}{l}{\textit{phrase}} \\ \text{SYNSEM LOC CAT NPCOMP} & + \\ \text{DTRS} & \left[\begin{array}{l} \textit{head-comp-struc} \\ \text{HEAD-DTR}\ \textit{phrase} \end{array}\right] \end{array}\right] \Longrightarrow$$

$$\left[\begin{array}{ll} \multicolumn{2}{l}{\textit{phrase}} \\ \text{DTRS} & \left[\begin{array}{ll} \multicolumn{2}{l}{\textit{head-comp-struc}} \\ \text{HEAD-DTR} & \left[\begin{array}{l} \textit{phrase} \\ \text{DTRS ALL-DTRS}\ \boxed{2}\ \textit{list-of-signs} \end{array}\right] \\ \text{COMP-DTRS FIRST} & \boxed{1}\ \textit{sign} \\ \text{ALL-DTRS} & < \boxed{1} \mid \boxed{2} > \end{array}\right] \end{array}\right]$$

The partial theory of *phrase* only makes statements about phrases which are headed phrases. Even more specifically, their DTRS value has to be *head-comp-struc*.

The implicational constraints in (3.25) fall into two groups. The first group of constraints, characterized by the NPCOMP value -, describes phrases in the verbal complex. The second group of constraints, characterized by the NPCOMP value +, describes phrases in the mittelfeld. In both groups of constraints two cases can be distinguished. In one case the ALL-DTRS list of a dominated phrase is inherited and in the other case no ALL-DTRS list is inherited since there is no phrase from which it could be inherited.

The first two constraints of (3.25) take effect in the verbal complex. The first one treats the case when the deepest phrase in the verbal complex has been reached. The complement daughter is a word and has no ALL-DTRS list. Only the head daughter and the complement daughter have to be put into the ALL-DTRS list of the dominating phrase. The second constraint will apply when the complement daughter is a phrase. This means, there is still more than one verb embedded and the ALL-DTRS list of the complement daughter has to be inherited in order to guarantee the presence of all verbal words on the ALL-DTRS list of phrases higher up in the syntactic structure.

Applied to our example (3.16) above the first two constraints in (3.25) lead to an ALL-DTRS list of the phrase immediately dominating the finite verb *hat* which contains the signs of *füttern*, *dürfen* and *hat*. The ALL-DTRS list of the phrasal sister of *hat* contains the signs of *füttern* and *dürfen*.

The third and the fourth constraint concern phrases dominating constituents of the mittelfeld. The third constraint is only applicable when the sentence has only a finite verb but no verbal complex. Then the head daughter of the phrase immediately dominating the finite verb is a word (namely the finite verb itself) from which no ALL-DTRS list can be inherited. The finite verb and the complement daughter are put into the ALL-DTRS list of the dominating phrase. We can always be sure that there is only one complement daughter by virtue of the Subcategorization Principle for binary branching structures. In all other cases in the mittelfeld the head daughter of the phrase under consideration is in turn a phrase and has an ALL-DTRS list of its own. By virtue of the last constraint in (3.25) the head daughter's ALL-DTRS list equals the rest of the current phrase's ALL-DTRS list while the complement daughter is the first element on it.

As a result the ALL-DTRS list of the highest phrase in (3.16) contains the signs of the constituents *Terry* and *den Kater* and the signs of all verbs in the verbal complex, namely *füttern*, *dürfen* and *hat*.

With the partial theory of *phrase* in (3.25) we have shown how a binary branching sentence structure can be furnished with constraints that result in an ALL-DTRS list corresponding to the ALL-DTRS list associated with a flat analysis of German sentence structure. We conclude this chapter with some remarks on the consequences the binary branching analysis brings about for the LP component beyond the different constraints for the ALL-DTRS list.

Surprisingly, there is only one major difference and it does not even affect the formalization of the LP principles themselves. It consists in the fact that in the new analysis the ALL-DTRS lists are in some sense incomplete for all projections below the lowest phrase dominating the entire mittelfeld. In other words, only the ALL-DTRS list of that phrase directly corresponds to the ALL-DTRS list occurring in the single projection of the flat analysis which dominates mittelfeld and verbs of the verbal complex alike. The other projections in the binary branching analysis contain shorter ALL-DTRS lists.

Two approaches to this situation are possible. First, one could apply LP principles exclusively to phrases with an empty COMPS list. The phonologies of lower projections could simply be determined by a stepwise reduction of the content of PHON list by the phonology of the sister of the given projection when going down the syntactic tree. For example, assume that the PHON list of the highest phrase in (3.16) is <*Terry, den, Kater, hat, füttern, dürfen*>. Then we arrive at the phonology of its head daughter <*den, Kater, hat, füttern, dürfen*> by simply taking out the phonology of its sister <*Terry*>.

A second possibility is to construct the objects mediating between syntax and LP principles as dynamically as the ALL-DTRS list. This means basically that the mediating objects have to take into account that at a given projection not all of the complements of the verbs have necessarily been realized yet. Since we choose the flat analysis as basis for the formalization of the LP component in chapter 4, the mediating theory of *ws2* developed there relies on the assumption that all signs corresponding to synsem objects on the COMPS lists of the verbs are on the ALL-DTRS list, i.e., they have been syntactically realized. A modified theory would have to check first which complements have already been realized when an appropriate *ws2* object is specified.

Apart from the diverging status of the ALL-DTRS list, the modifications caused by differing analyses affect the choices the grammar writer has to build the objects that are specified as necessary interfaces to the LP principles. For example, in the binary branching structure the syntactic tree can be employed to determine easily which signs are arguments of others. Assumptions on the species of the objects on COMPS lists and the relation between SUBJ list and COMPS list are other instances of variations in the syntax that lead to the same type of modifications. In the next chapter we will point out where such syntactic assumptions of section 3.1 are important for the specific formalization presented in chapter 4.

We conclude that our approach to linearization is largely independent of the syntactic analysis a grammar writer prefers. A grammar is compatible with our LP module if the information for the interfaces which we discuss in more detail in the next chapter can somehow be provided. This means it must be possible to construct certain objects based on the information available in the syntactic analysis. An important example for such objects is the ALL-DTRS list. Another one will be objects of sort *embed* which are discussed in the next chapter.

4 Linear Precedence in HPSG

This section comprises the core of the work on linear precedence in this thesis. In section 4.1, there will be a short discussion of the classical GPSG conception of LP rules, which will be contrasted with the more flexible way to express precedence regularities that is taken in this thesis. Section 4.2 consists of a discussion of the attribute LP which was introduced on *phrases* in chapter 2. The internal structure of the possible values of this attribute is to be presented and discussed in detail in sections 4.3 – 4.6. This includes i.a. two concrete LP principles (in sections 4.4 and 4.5). Section 4.7 contains summarizing remarks on the way precedence regularities are technically captured under the present approach and an outline of how to exploit the given devices to state LP principles other than the ones discussed in sections 4.4 and 4.5.

4.1 A new conception of LP

This introductory part of chapter 4 sketches the basic ideas of the approach to LP put forward in this thesis and shows the relation between our approach and the way LP rules are used in some of the HPSG literature. In section 4.1.1 an overview will be given of how three basic properties of GPSG LP statements (cf. the remarks in section 1.1) are reflected in HPSG. The outline of a new approach to LP in section 4.1.2 will then identify the devices used to avoid the GPSG restrictions in a desired way.

4.1.1 Classical LP statements and HPSG

Linear precedence statements were introduced into GPSG in the classical article of Gazdar and Pullum (1981). The authors use the sign $<$, which later became $\prec$, to state precedence relations. In the 1985 formulation of the theory (Gazdar et al. 1985), LP rules look as in (4.1).

(4.1) = *Gazdar et al. (1985, example (14),page 50)*

$$[\text{SUBCAT}] \prec \sim[\text{SUBCAT}]$$

LP-statements are interpreted the following way:[27]:

> To be more precise, we intend 'A $\prec$ B' to be interpreted as 'a node labeled B cannot appear to the left of a node labeled A in a local tree'
>
> (Gazdar et al. 1985, p. 48)

As has been pointed out in section 1.1 of the introduction, GPSG like LP rules exhibit the following properties:

(4.2) *Properties of GPSG LP statements*

1. All LP statements have to be obeyed by all local trees.
2. The ordering occurs only among sister constituents.
3. The elements to be ordered have to be identifiable via attribute specifications on the elements themselves.

We shall now briefly show some shortcomings of the restrictions listed under (4.2) and indicate the means used by some authors to circumvent them.[28]

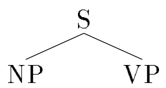
As far as (4.2-1) is concerned, a linguist might want to state a generalization that the head daughter has to precede complement daughters,

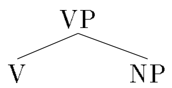
[27] This interpretation of the $\prec$-symbol is different from the one in Gazdar and Pullum (1981). The difference however is not crucial for the present purpose.

[28] Some brief comments on the properties listed in (4.2) might be useful. Properties (4.2-1) and (4.2-3) follow from stepping away from traditional phrase structure rules. As linearization is not encoded together with the hierarchical structure in one rule, not all of the information given in a traditional PS rule is available for ordering

constraints. As an example, there is no need to determine the 'order' of the mother category in a local tree. Therefore, information about the mother category is not available to the linearization part of an ID/LP grammar. As a consequence it is impossible to make the ordering of sisters in a local tree dependent on properties of the mother. Put differently precedence regularities can only be stated with respect to information on the right-hand side of an ID rule.

The second property given under (4.2) follows from the definition of LP rules. The restriction to the ordering of sister constituents only leads to some open questions in larger trees. In Gazdar et al. (1985, section 5.5.) nothing excludes trees like the one in (iii), which consists of the local trees in (i) and (ii):

(i) 

(ii) 

but to follow filler daughters (as it is the case for English). As it is impossible to make GPSG's LP statements dependent on ID rules, such a generalization cannot be expressed directly.[29]

In the Pollard and Sag (1987)-formulation of the *Constituent Order Principle* (page 169, repeated above as example (2.1)), there is a token identity between the value of the DTRS attribute of a *phrase* and the argument of the function `order-constituents`. By this identity the information about the species of the value of the DTRS attribute is available to the function `order-constituents`. Thus the outline of an HPSG theory of word order leaves open the option to make precedence regularities sensitive to construction types.

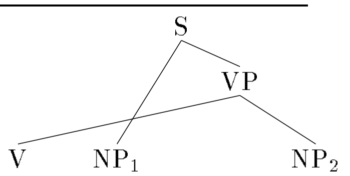
The next property of GPSG LP statements mentioned in (4.2-2) is the restriction of their application to local trees. The consequences of this have been listed in section 1.1. We shall repeat two major shortcomings of a theory of LP which restricts itself to local trees.

First, it forces GPSG to adopt a completely flat analysis for sentences with subject-aux-inversion (Gazdar et al. 1985, section 4.3), as the authors assume there to be no verb movement involved. This problem carries over to the standard HPSG analysis of verb second phenomena in German and other languages as expressed in Pollard and Sag (1987, p. 156 f.), Pollard (1990) and Pollard and Sag (1994, p. 40 f.).

Second, if one wanted to allow only binary branching structures even for head-complement structures, coarguments would no longer be sister constituents in a local tree. Then, there would be no possibility to state the relative order of the arguments via LP statements. The ordering would have to follow from other parts of the grammar.[30] In HPSG binary branching structures are easily enforced, as it is done e.g. in Kiss (1993,

---

(iii)



In (iii) the linear order among sister constituents is kept unchanged, globally, however, the V daughter of the VP may precede NP_1, a daughter of S, even though NP_1 precedes VP in (ii) and (iii). But, to our knowledge, trees like (iii) have never been used in any GPSG analysis. So one might think of a redefinition of *admissibility* (Gazdar et al. 1985, p.104) which excludes crossing branches.

[29]For this concrete case, Gazdar et al. (1985) offer an alternative account which makes reference to the SUBCAT feature of a lexical category for the ordering of head and complements and to the categorial features for structures corresponding to head-filler structures. The first case is treated by the LP statement in (4.1), the second by the statement $N^2 \prec P^2 \prec V^2$ (Gazdar et al. 1985, p. 110).

[30]We are however aware of the fact that there is no direct way to introduce binary branching structures into the theory of Gazdar et al. (1985). The point made here nevertheless would apply to any such binary branching variant of GPSG.

page 129) for German. This step motivates an extension of the domain for LP statements from local trees to something like head domains (Kiss 1993, page 148).

The third property of GPSG LP statements given in (4.2) has to do with the kind of information that can be used in the formulation of an LP statement. For GPSG only information which is directly available in a category can be used for this purpose. Thus it would be possible to make the order of categories dependent on their syntactic category (verb, noun , ...), case (nominative, accusative, ...). Their thematic role, however, is not accessible for ordering constraints, because there is no attribute like THEMATIC-ROLE, but this information is expressed in the content of the subcategorizing element. Meurers and Morawietz (1994) describe how this problem can be circumvented by simply adding such an attribute,[31] a technique which has actually been exploited in Uszkoreit (1986a) for GPSG.[32]

The theory outlined in Pollard and Sag (1987), adopted from Sag (1987), makes use of a much more complicated structural property of a sign. They introduce a precedence symbol $\ll$ (Pollard and Sag 1987, p. 174), which they want to mean the following:

(4.3) = (334) LP2 (first formulation):

COMPLEMENT $\ll$ COMPLEMENT

> Here the symbol $\ll$ designates a special kind of restricted linear precedence constraint which has force only in case the left-hand element is less oblique than the right-hand element.
>
> (Pollard and Sag 1987, p.174)

The symbol $\ll$ thus refers to a property external to the signs to be ordered, i.e. their position on the obliqueness hierarchy. Furthermore it also compares these positions relative to each other. Such an LP statement cannot be reformulated in a straightforward way by introducing a new attribute, as it refers to relative positions on a hierarchy and not to absolute positions.

[31]Meurers and Morawietz (1994) demonstrate this technique by encoding whether a given sign is a HEAD-DTR or a FILLER-DTR in a structure.

[32]We reject an attribute THEMATIC-ROLE on complements as long as there is no independent evidence that the information about a thematic role is really of any use on the complement itself. We assume thematic information to be relevant for the semantics of an event, not for the semantics of its participants (cf. the literature on event semantics as e.g. Parsons (1990), and Rentier (n.d., footnote 3) for work on HPSG which explicitly incorporates a version of event semantics). Our position is furthermore fully in line with the treatment of thematic roles in Pollard and Sag (1987, chapter 4) and Pollard and Sag (1994, chapter 8).

The discussion of the properties listed in (4.2) has brought to light various shortcomings of the GPSG LP component. Firstly, these properties constrain linguistic theories in an unintuitive and perhaps even empirically problematic way by disallowing certain plausible LP statements and enforcing certain structural analyses. Secondly, reference was made to HPSG researchers who made use of LP constraints unexpressable in an LP theory that sticks to the properties in (4.2)

Having looked at the GPSG LP heritage, one wonders where to find an updated theory of LP which meets the needs of HPSG. This theory has to be formally as precise as the GPSG theory and should fit into the general logical foundations of HPSG. It was shown in the introductory chapter 1 that there is no such HPSG specific LP component developed which could be considered satisfactory from a theoretical linguistic point of view. In 4.1.2 an overall architecture of a theory of LP is presented which has been explicitly designed to fill this gap.

4.1.2 Outline of an LP component for HPSG

Within the second part of section 4.1 we intend to give an outline of the basic elements of an LP component for HPSG. The general functions of these elements will be described on an intuitive level. To elaborate on the connection between the new treatment of LP and the accounts mentioned in section 4.1.1, reference will be made to the three properties of classical LP statements listed under (4.2) whenever relevant. For every part of the LP component there will be a precise definition given in one of the following sections.

In chapter 2 the sort *phrase* has been enriched by the attributes BUILD-PHON and LP. It was shown how to capture the generalization of Pollard and Sag (1987), that the phonology of a phrase is always a permutation of the PHONOLOGY (PHON) values on the phrase's daughters. The technical devices needed to ensure this property were put into the value of the attribute BUILD-PHON. The second new attribute, LP, will now be used to restrict the phrase's phonology to be not just any permutation, but a grammatically possible permutation. To get an idea of what is meant by this, consider the example in (4.4). In (4.4-1) the hierarchical structure of a German interrogative sentence is given. (4.4-2) contains the permutations of the PHON values which occur in (4.4-1).

(4.4) *Some permutations*

1. *The hierarchical structure:*

$$\begin{bmatrix} \textit{head-filler-struc} & \\ \text{HEAD-DTR} & \boxed{1} \begin{bmatrix} \text{PHON} < \textit{hat, Peter, gelesen} > \\ \ldots \end{bmatrix} \\ \text{FILLER-DTR} & \boxed{2} \begin{bmatrix} \text{PHON} < \textit{was} > \\ \ldots \end{bmatrix} \\ \text{COMP-DTRS} & \textit{elist} \\ \text{ALL-DTRS} & < \boxed{1}, \boxed{2} > \end{bmatrix}$$

2. *Some permutations of the phonologies:*
   (a) was hat Peter gelesen
   what has Peter read
   (b) * hat Peter gelesen was
3. *A linearization regularity:*
   In the phonology of the mother, the phonology of the head daughter follows the phonologies of the other elements to be ordered.

For every sentence in (4.4-2) it holds, that it represents a permutation of phonology values of the daughters in (4.4-1). Nevertheless, one might assume that there is a linearization regularity like (4.4-3) which orders the phonology of the head after the phonologies of the other daughters. As the string *hat Peter gelesen* in (4.4-1) is the PHON value of the head daughter, the permutation in (4.4-2a) is in line with the linearization regularity, whereas permutation (b) is to be ruled out by the LP component. Thus the value of the attribute LP will force the phonology of the head to follow the phonologies of the other elements of the ALL-DTRS list.

A look at the linearization regularity in (4.4-3) shows a general property of all LP principles: a relation is established between the PHON value of a *phrase* and the PHON values of the elements on its DTRS ALL-DTRS value. This leads us to reconsider the locality restriction which holds for LP rules in GPSG, cf. (4.2-2). In chapter 3 some structural assumptions for German have been discussed, including the way the ALL-DTRS list is constructed under these assumptions.

In the 'flat' analysis, sketched in section (3.1), the ALL-DTRS list contains just sisters of a local tree. In this case the LP principles, too, are local as they express a relation between the PHON list and the immediate daughters of a *phrase*. However, it has been shown in section (3.2) how the ALL-DTRS list can be constructed for a 'structured' analysis for German. In the structured case the LP principles remain unchanged as the

ALL-DTRS list contains exactly the same elements as in the 'flat' case.[33] Now, however, the LP principles are nonlocal: they encode a relation between the PHON list and non-immediate daughters of the *phrase*.

Thus, the answer to the question of whether the LP principles just apply to the phonologies of sisters in a local tree depends on the way the ALL-DTRS list is constructed. But, as the present system is built in a modular way, the LP principles themselves are blind to the particular construction mode of the ALL-DTRS list. They always apply to the elements on the ALL-DTRS list no matter where these elements stem from in the hierarchical structure. This treatment of LP departs from the close link between the LP rules and the structural analysis that has been expressed in (4.2-2), i.e. the strict locality restriction for LP rules in GPSG. The main advantage of this aspect of the new conception of LP is that it enables the grammar writer to change her structural analysis and still stick to the same LP principles and vice versa.

A second aspect of the new conception of LP can be illustrated in contrast to the generality restriction of classical LP rules, (4.2-1). The GPSG restriction says that every LP rule holds in every local tree.

In (4.4-3) a very simple case of an LP principle has been mentioned. In the present approach this LP principle has to be put into the value of the attribute LP, together with all other LP principles which are to apply. The LP principle informally given in (4.4-3) is appropriate i.a. for German head-filler structures and head-complement structures with an uninverted verbal head. In the case of English, however, it must not apply in head-complement structures, as heads precede their complements in English. In contrast to the traditional GPSG conception of LP rules, it should be possible within the HPSG LP component to state more flexibly in which phrases a given LP principle is to hold. This is achieved in the following way:

The value of the attribute LP is of sort *lp*. As *lp* is a sort, it may have several subsorts (species). We will define the sort *lp* in such a way that its species contain different collections of LP principles. The distribution of the species of *lp* has to be accounted for in the theory of *phrase*. For the purpose of this thesis we assume that there is a direct link between the species of a phrase's DTRS specification and the one of its LP value.[34]

This brings us back to the case where the head daughter comes first in

[33]At least the elements of the mittelfeld are the same. It depends on the treatment of the verbal complex whether the ALL-DTRS list really consists of the same elements in both analyses.

[34]Of course, a link to any other attribute specification is possible as well. A good candidate for this would be the syntactic category, i.e. the species of SYNSEM LOCAL CAT HEAD. It is a task subject to empirical research to decide which attribute specifications correlate with which set of LP principles.

head-complement structures, but it comes last in head-filler structures. This case has been introduced to illustrate a problem that arises with assumption (4.2-1). In the new approach to LP one can include an LP principle for a head-initial linearization in the species of *lp* which is adequate for a *phrase* with a DTRS value of sort *head-comp-struc*. An LP principle like (4.4-3) which enforces head-final ordering would then be part of the *lp* species that is adequate for a phrase with a DTRS value of sort *head-filler-struc* on their DTRS attribute.

It is of course still possible to express that a given LP principle holds for every *constituent-structure* of a language. This LP principle would then be included in all species of *lp*, via inheritance from the sort hierarchy. It is obvious that such a differentiation in the species of the sort *lp* can overcome the problems of the very restrictive GPSG conception. It neither excludes general LP principles, nor does it automatically impose them.

To further illustrate this point, we give the partition of the sort *lp* in (4.5). The names of the species are chosen to recall the species of *constituent-structure* which correspond to them. Thus *lp-hf* goes together with *head-filler-struc*, *lp-hc* with *head-comp-struc* and so on.

(4.5) *Subsorts (species) of lp*

lp

lp-hf *lp-hc* *lp-ha* *lp-hm* ...

With this partition in mind, the LP principle given in (4.4-3) which enforces a head final linearization, would be part of the species *lp-hf* in English, but not of the species *lp-hc*. In head final languages like Japanese, one expects this LP principle to be present on all species of *lp*.

So far we have commented on the GPSG locality and generality restrictions, (4.2-1, 2). We said that, for the LP component developed in this thesis, every species of *lp* represents a collection of LP principles. Furthermore, an LP principle is thought of as a relation between the PHON value and the ALL-DTRS value of a *phrase*. The last GPSG restriction mentioned in (4.2-3) contains the claim that LP rules only make reference to properties inherent to the elements to be ordered. At the end of the present section, 4.1.2, it is shown how the LP component of this thesis is able to handle linearization regularities that are sensitive to other properties as well. Before this can be done, however, the attributes and sorts used within the sort *lp* have to be presented in some detail.

In the following an example species of the sort *lp* is presented, the species *lp-example*. We shall present the different kinds of attributes a species of *lp* is specified for by using the species *lp-example*. In sections 4.2 – 4.6 parts of a concrete linguistically motivated species of *lp*, *lp-hc*, are discussed. Thus it is the purpose of the discussion of *lp-example* to abstract away from particular LP principles and to identify the common properties of all species of *lp*. Looking at *lp-example*, the overall structure of the LP component of this thesis will become clear.

This level of abstraction leads to severe problems of presentation, as the formalism chosen in chapter 1 forces us to always give a concrete signature. As a consequence, the attributes given for *lp-example* are to be interpreted as examples for the kinds of attributes that occur on 'real' species of *lp*, like *lp-hc*. Therefore, attribute names given in this section will not be found in the subsequent sections of chapter 4. The functions associated with the attributes of *lp-example* are, however, taken over by the attributes of the other species of *lp*. The attribute labels for *lp-example* are chosen to evoke the function of the attributes within the LP component.

In the discussion of the locality properties of the present LP component we have said that an LP principle establishes a relation between the PHON list and the DTRS ALL-DTRS value of a *phrase*. Therefore, it is known that at least this information from a *phrase* is needed within its LP value. Internally to the *lp* object this information is restructured for use within the LP principles. This leads to three different kinds of attributes in the *lp* object:

- external interface attributes which use information from the *phrase* to build certain auxiliary structures. (In (4.6), the external interface attributes are BUILD-AUX1, ... BUILD-AUXn.)

- auxiliary structure attributes whose values are provided by the external interface attributes and which serve as internal interfaces to the LP principles. (In (4.6), the auxiliary structure attributes are AUX1, ... AUXn.)

- attributes which encode the LP principles proper. (In (4.6), the LP principle attributes are LPP1-1, ... LPPn-m_n.)

Before some motivation and comment on the different kinds of attributes is given, their interaction is demonstrated in a toy *lp-example* object as the LP value in (4.6).

(4.6) *Exemplifying the LP component*

$$
\left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{phrase}} \\
\text{PHON} & \boxed{x}\ \textit{list-of-phons} \\
\text{SYNSEM} & \textit{synsem} \\
\text{BUILD-PHON} & \textit{build-phon} \\
\text{LP} & \left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{lp-example}} \\
\text{PHON} & \boxed{x} \\
\text{BUILD-AUX}1 & \left[\begin{array}{ll} \multicolumn{2}{l}{\textit{build-aux-1}} \\ \text{INPUT} & \boxed{y} \\ \ldots & \\ \text{OUTPUT} & \boxed{1} \end{array}\right] \\
\ldots & \\
\text{BUILD-AUX}n & \left[\begin{array}{ll} \multicolumn{2}{l}{\textit{build-aux-n}} \\ \text{INPUT} & \boxed{2} \\ \ldots & \\ \text{OUTPUT} & \boxed{n} \end{array}\right] \\
\text{AUX}1 & \boxed{1} \\
\ldots & \\
\text{AUX}n & \boxed{n} \\
\text{LPP}1\text{-}1 & \left[\begin{array}{ll} \multicolumn{2}{l}{\textit{lpp-1-1}} \\ \text{PHON} & \boxed{x} \\ \text{AUX} & \boxed{1} \\ \ldots & \end{array}\right] \\
\ldots & \\
\text{LPP}1\text{-}m_1 & \left[\begin{array}{ll} \multicolumn{2}{l}{\textit{lpp-1-}m_1} \\ \text{PHON} & \boxed{x} \\ \text{AUX} & \boxed{1} \\ \ldots & \end{array}\right] \\
\vdots & \\
\text{LPP}n\text{-}1 & \left[\begin{array}{ll} \multicolumn{2}{l}{\textit{lpp-n-1}} \\ \text{PHON} & \boxed{x} \\ \text{AUX} & \boxed{n} \\ \ldots & \end{array}\right] \\
\ldots & \\
\text{LPP}n\text{-}m_n & \left[\begin{array}{ll} \multicolumn{2}{l}{\textit{lpp-n-}m_n} \\ \text{PHON} & \boxed{x} \\ \text{AUX} & \boxed{n} \\ \ldots & \end{array}\right]
\end{array}\right] \\
\text{DTRS} & \left[\begin{array}{l}
\textit{con-struc} \\
\ldots \\
\text{ALL-DTRS}\ \boxed{y}\ \textit{list-of-signs}
\end{array}\right]
\end{array}\right]
$$

The *phrase* in (4.6) is specified for the attributes PHON, SYNSEM and DTRS given in Pollard and Sag (1994), and those introduced in chapter 2, example (2.2). There, the signature of Pollard and Sag (1994) was augmented by the attribute BUILD-PHON whose value is discussed in chapter 2, and the path DTRS ALL-DTRS defined in chapter 3.

The LP value of the *phrase* in (4.6) has only two links to other parts of the *phrase*: One established by the tag $\boxed{x}$ which makes the overall phonology accessible inside the LP value. The second makes the DTRS ALL-DTRS list (tagged as $\boxed{y}$) available to the LP component.

The LP principles encoded in the species *lp-example* are localized in the values of the attributes LPP1-1, ...LPPn-m_n, the LP principle attributes. As noted earlier, we consider LP principles to establish a relation between the PHON value and the DTRS ALL-DTRS list of a *phrase*. In fact, as can be seen in (4.6), the information about the overall PHON value is found in every LP principle attribute LPPi-j ($1 \leq i \leq n, 1 \leq j \leq m_i$) by being the LP LPP$i$-$j$ PHON value.

Things are slightly more complicated with the value of the DTRS ALL-DTRS list. There is no structure sharing between this list and an attribute within an LPPi-j value. Rather, procedurally speaking, the ALL-DTRS list is transformed into a new structure which is of better use for the LP principle attributes. In the *lp-example* object in (4.6), there are n such restructurings, encoded in the values of the attributes BUILD-AUX1, ...BUILD-AUXn, the attributes external interface attributes. The INPUT value of every BUILD-AUXi ($1 \leq i \leq n$) is guaranteed to be token identical to the phrase's DTRS ALL-DTRS value in the theory of *phrase*. Thus, the tag $\boxed{y}$ can be found as INPUT value on every *build-aux-i* in (4.6). Within the *build-aux* object the INPUT value information is restructured to yield the OUTPUT value. The restructuring fulfills certain criteria. For example, one of the external interface attributes that are needed is such that the ordering of the signs in the OUTPUT value corresponds to the order of their *synsem*s on the head daughter's COMPS list.

The OUTPUT value of each *build-aux-i* is token identical to the value of some attribute AUXi, an auxiliary structure attribute. The value of any AUXi is linked to the ALL-DTRS list via an external interface attribute BUILD-AUXi in the way just described.

Having arrived at the auxiliary structure attributes, we can go back to the information used to encode LP principles in LP principle attributes. For each auxiliary structure AUXi there is a group of LP principle attributes, LPPi-1, ...LPPi-m_i. As can be seen in (4.6), for every LP principle attribute LPPi-j in this group, there is structure sharing between the AUXi value and the LPPi-j AUX value of *lp-example*. Intuitively this means that there is a finite number of useful criteria to restructure the ALL-DTRS list. For every restructuring, there is a group of LP principles which can be expressed with reference to the properties of the restructuring.

As all attributes that occur in (4.6) have now been discussed, this is a good place to again draw attention to the modularity and transportability of the approach to LP outlined in this thesis. The values for the external interface attributes (BUILD-AUX1, ... BUILD-AUXn) and the PHON attribute are the only parts of the *lp-example* object which are referred to in the theory of *phrase* (cf. section 5). Therefore they are all that has to be changed if one wants to adopt a structural analysis different from the one presented in this thesis. The rest of the LP component is largely independent of any particular analysis.

A second look at the AVM in (4.6) might give the impression that one could easily do without the attributes PHON and the auxiliary structure attributes AUX1, ... AUXn of the *lp-example* object. The relation between the overall PHON list and the PHON value of the LP principle attributes (the LPPi-js) could be taken care of directly in the theory of *phrase*. Such a modification of the LP component would, however, lead to a loss in modularity. If one had an LP component without a PHON specification in the LP value, the theory of *phrase* would have to be changed whenever a new LP principle attribute is added, or an old one is removed. Therefore it seems more attractive to keep the LP component as independent of the rest of the grammar as possible and to make the overall PHON list available in the LP value by means of a PHON attribute.

Not only the attribute PHON on the *lp-example* sort might seem superfluous, but the attributes AUX1, ... AUXn in (4.6), too. In fact, it would be easy to do without these attributes. As an example, let's look at the structure sharing between the BUILD-AUXi OUTPUT and the LPPi-j AUX path. As the LP component is set up in (4.6), this structure sharing is established via the identity of the paths BUILD-AUXi OUTPUT and AUXi of the *lp-example* object on the one hand and the paths AUXi and LPPi-j AUX on the other.

Our reason for not giving up the auxiliary structures is not just aesthetical in nature. We noted that the external interface attributes are subject to change whenever the hierarchical analysis of a sentence is changed. Nevertheless, the AUXi values remain constant whatever the syntactic structure looks like. These values can be seen as input specifications of the LP principles. This implies that linguists working on the syntactic structure have to provide the BUILD-AUXi values which determine the values of the auxiliary structure attributes AUXi. On the other hand, grammar writers only concerned with linearization regularities would prefer to ignore the external interface attributes. In defining LP principles, they may simply refer to the auxiliary structures. So, the first group of linguists does not have to bother about any part of the LP principle attributes, whereas the second group does not need any information from the external interface attribute values or from elsewhere in the *phrase*. The only objects both groups of linguists have to deal with are the values of the auxiliary structure attributes. Thus, the presence of the attributes AUXi, ... AUXn contributes to the modularity of the present approach.

After these considerations concerning the overall structure of the LP component outlined in (4.6), we can address its different parts, i.e. the three kinds of attributes found in (4.6). The first group of attributes to address consists of the external interface attributes BUILD-AUX1, ... BUILD-AUXn. As they are not part of the LP component proper, there is not much left to say about them. They are only of technical use and serve to provide the restructurings of the ALL-DTRS list that are needed for the LP principles.

The second group of attributes consists of the auxiliary structure attributes given in (4.6) as AUX1, ... AUXn. They are used to allow for very flexible LP principles. We assume there to be at least three auxiliary structures:

- a list of signs on which the elements of the phrase's DTRS ALL-DTRS list appear in such a way that all complements of the DTRS HEAD-DTR occur in the order of the corresponding *synsem* objects on the head daughter's COMPS list. This will be the value of an attribute called BASE.

- a structure which organizes the complements of the head-daughter in such a way that the structure of semantic embedding becomes evident by undoing the effects of argument attraction. This structure will be presented in section 4.3 where it figures as the value of an attribute EMBED.

- a structure which mirrors the scope relations among the signs on the phrase's DTRS ALL-DTRS list. In section 4.7.3 this structure is sketched as the value of an attribute SCOPOI.

In this thesis only the second of these auxiliary structure attributes, EMBED, is presented in detail (in section 4.3). It has been chosen for presentation because the corresponding external interface attribute is much more complex than the external interface attribute that is needed for the first auxiliary attribute.[35] The reason for not discussing the third attribute, SCOPOI, lies in the fact that there are too many open questions concerning the precise nature of the SCOPOI structure. Some of these questions are addressed in section 4.7.3.

[35]The construction of the BASE value is sketched in section 4.7.2.

There is empirical evidence for the relevance of the auxiliary structures just listed for capturing linearization regularities. The ordering on the COMPS list (or on the obliqueness hierarchy) is claimed to be essential for linearization in Sag (1987) and Pollard and Sag (1987) and captured by introducing the special precedence symbol $\ll$.[36] In sections 4.4 and 4.5 two examples for LP principles will be given which make reference to the structure of semantic embedding. The influence of scope inclusion on linearization has been pointed out e.g. by Jacobs (1988, p.20) and Reape (1992, p. 253, example 4.139)

All auxiliary structures have in common that they encode information which is not inherent to the signs on the ALL-DTRS list, but which can be inferred from the environment in which these signs occur. By making this information available to the LP component, i.e. inside the LP attribute of a *phrase*, quite complex LP regularities can be stated. Thus, the auxiliary structures provide a means to overcome the last of the restrictions of GPSG LP statements mentioned in (4.2) and to accommodate the present system to the needs imposed by the data and to the practice of HPSG researchers.

So far it has been sketched how to incorporate LP principles into the structure of a *phrase* and how to collect the information relevant for the LP principles. However, nothing has been said about the LP principles themselves, nor has there been anything like a symbol $\prec$ that expressed precedence. To fill the gap, the third group of attributes of an *lp* object is to be looked at next. As can be seen in (4.6), every LP principle ends up as one of the LP principle attributes LPP1-1, ... LPPn-m_n. We assume that there is only a finite number LP principles to be encoded.[37] Let's take a closer look at one of these LP principle attributes, LPPi-j with a value of sort *lpp-i-j*.

Every *lpp-i-j* object has two attributes whose values are structure shared with other parts of the *lp* object. These are the attributes PHON and AUX. As can be seen in the AVM in (4.6), these attributes get the values of the *lp-example*'s PHON list and of one of the auxiliary structure attributes AUXi. This relation between the *lp* object and the attributes in its LPPi-j feature is expressed in the theory of *lp-example*. For the fragment of the LP component worked out in this thesis, this is done in section 4.6.

Now that we have established how the necessary information is linked

[36]This list is also the starting point for the permutation rules used by Meurers and Morawietz (1994) in their Successive Permutation (SUP) grammars.

[37]The logic defined in section (1.2) would permit objects with an infinite number of attributes, provided the set of attributes is infinite. However, as no description can quantify over attributes, the set of attributes that can practically be referred to in linguistic theory is always finite.

to the LPPi-j value, the formulation of an LP principle can be focussed on. An LP principle consists of two parts: First, a method is stated of how the linearization regularity is to be checked with respect to the auxiliary structure. And second, the checking procedure is stated.

There are basically two options for the first part. The auxiliary structure can either be looked at as a whole or the checking can proceed along its (recursive) structure. Two simple examples may serve to illustrate this point. Both examples describe LP principles which make use of the PHON value of the *phrase* and of the BASE structure. Let's call the two LP principle attributes LPP-SIMPLE and LPP-RECURSIVE, of sort *lpp-simple* and *lpp-recursive* respectively. Assume the AUX attribute of both LP principle attributes to be the list of the elements of the ALL-DTRS value ordered according to their position on the COMPS list (or the obliqueness hierarchy).

We shall first discuss the simpler case of *lpp-simple* and illustrate at the same time the way the precedence checking procedure works. After this, we are ready to demonstrate the recursive type of LP principle attributes by looking at the sort *lpp-recursive*. Comparing the two sorts it will become evident that the 'simpler' case

contains a precedence checking mechanism that is applied only once. In the 'recursive' case exactly the same checking procedure is used, but it is at work several times depending on the length of the auxiliary structure. It is only in this sense that the sort *lpp-simple* can be considered 'simpler' than the sort *lpp-recursive*.

The attribute LPP-SIMPLE encodes an LP principle which orders the phonology of a sign on its AUX value marked INV $+$ before the phonologies of all other signs on the AUX value. This LP principle can be checked by applying a checking procedure to the auxiliary structure only once. Discussing LPP-SIMPLE gives us the opportunity to roughly sketch the way precedence checking is done in the present approach. It will become evident that this part of our LP component corresponds to an intuitive interpretation of a precedence symbol such as $\prec$. The sort *lpp-simple* has the following general structure:

(4.7) *Exemplifying the lpp-simple object*

$$\left[\begin{array}{ll}
\textit{lpp-simple} & \\
\text{PHON} & \boxed{1}\ \textit{list-of-phons} \\
\text{AUX} & \boxed{2}\ \textit{list-of-signs} \\
\text{FIND-A/B} & \left[\begin{array}{ll}
\textit{find-a/b} & \\
\text{AUX} & \boxed{2}\ \textit{list-of-signs} \\
\text{A-ELEMENTS} & \boxed{3}\ \textit{list-of-phons} \\
\text{B-ELEMENTS} & \boxed{4}\ \textit{list-of-phons}
\end{array}\right] \\
\text{PREC} & \left[\begin{array}{ll}
\textit{prec} & \\
\text{PHON} & \boxed{1}\ \textit{list-of-phons} \\
\text{A} & \boxed{3}\ \textit{list-of-phons} \\
\text{B} & \boxed{4}\ \textit{list-of-phons}
\end{array}\right]
\end{array}\right]$$

As we pointed out before, the first two attributes of the *lpp-simple* object get their values from the theory of *lp-example*. The others are only relevant internal to the *lpp-simple* object. The task of the attribute FIND-A/B is to identify the phonologies of those signs from the AUX value whose relative order on the PHON list is to be checked. The 'preceding' phonologies are collected in the A-ELEMENTS value, the 'preceded' ones in the B-ELEMENTS list. The value of the attribute PREC(edence) makes sure controlled that on $\boxed{1}$ all elements of $\boxed{3}$ occur before any element of $\boxed{4}$, to use the tags from (4.7).

To guarantee the correct working of the precedence checking procedure, it must be the case that the phonologies collected in the values of A-ELEMENTS and B-ELEMENTS actually occur on the PHON list of the *lpp-simple* object. This precondition is satisfied in the present approach: as *lpp-simple* occurs in a *phrase* only as the value of the path LP LPP-SIMPLE, it is assured by the theory sketched in (4.6) that the LP LPP-SIMPLE PHON value is identical to the *phrase*'s PHON value. Thus, the PHON values of the relevant *lp-simple* objects will contain the phonologies of all elements of the AUX list.

The LP principle attribute LPP-SIMPLE represents the simplest group of LP principles. Let's call this group of LP principles the 'simple' group, in contrast to the 'recursive' group represented by LPP-RECURSIVE sketched below. All LP principles of the simple group share some encoding properties when they are expressed as values of an LP principle attribute. Firstly they bear attributes as FIND-A/B and PREC that encode the actual precedence checking mechanism. This is — just as the presence of the PHON value and an auxiliary structure — a common property of all sorts that encode LP principles

Secondly, sorts as *lpp-simple* always come as species, i.e. there are no further subsorts of *lpp-simple*. As a consequence thereof, a sort like *lpp-simple* cannot have an attribute that takes an *lpp-simple* object as its value. This characteristic shows that the sort *lpp-simple* is not recursive itself, even though some of its attributes may take recursive sorts as their values. This property can be compared with properties of sorts we have already defined in previous sections. The sort *build-phon* from (2.3) and (2.13) is not recursive: there is only one species and thus there is no attribute on it that takes a *build-phon* object as its value. Nevertheless, the sort *build-phon* bears two attributes PERM and SIGNPHON that are recursive. The values of the PERM attribute for example are of sort *perm* ((2.4) and (2.7)). The sort *perm* comes in two species, one (*perm0*) being non-recursive, i.e. there is no attribute with a *perm* object as its value, the other species (*perm1*) being recursive as there is an attribute REC on it whose value is again of sort *perm*.

If one does not look into the structure of the attributes of a given sort but is only concerned with the shape of the sort itself, it is possible to say that the sort *build-phon* is simpler than the sort *perm*, even though the latter appears in an attribute value of the former. In this sense, a sort like *lpp-simple* belongs to the nonrecursive sorts just as *build-phon*.

If GPSG LP rules were to encode as LP principles within the present framework, the LP rules would all end up as LP principles of the simple group. But, the simple group is in fact stronger than GPSG LP rules. An LP principle which orders the phonology of the most oblique NP at the beginning of the overall PHON value would be a simple LP principle, as it can be verified by a single application of the precedence checking procedure. Yet, it is not expressible as a GPSG LP rule, because it refers to the position of an element on the obliqueness hierarchy. The additional restriction necessary for classical LP rules is that they directly use the *phrase*'s ALL-DTRS list as their auxiliary structure, i.e. as their AUX value. GPSG LP rules are not allowed to refer to any of the auxiliary structures sketched in the discussion of the attributes AUX1, ... AUXn. We shall refer to the subgroup of the simple LP principles that uses the unchanged ALL-DTRS list as the 'restricted simple' LP principles.

Narrowing down the possibilities of formulating LP principles to the restricted simple group results in LP principles that have the third property of GPSG LP rules listed in (4.2). Namely, they can only make reference to properties of the signs themselves and not to any property of the context in which a sign appears, i.e. its thematic role, level of embedding or position on the obliqueness hierarchy.

We noted in section 4.1.1 that HPSG grammarians assume linearization regularities that are not in line with the restriction to sign inherent properties given in (4.2-3). This means that we need LP principles which are outside the restricted simple group are needed. Even the simple group is not strong enough. According to Pollard and Sag (1987, p.174), cited in (4.3), the phonologies of complements are linearized parallel to their order on the obliqueness hierarchy. For the sake of illustration, we assume this linearization regularity to be encoded as an LP principle in the value of the attribute LPP-RECURSIVE, which is thought of as one of the LP principle attributes LPP*i*-*j* of the *lp-example* object in (4.6).

(4.8) illustrates how precedence checking has to proceed to express Pollard and Sag's LP principle, repeated in (4.8-1). In (4.8-2) the value of the relevant auxiliary structure attribute in the *lpp-recursive* object is given. In (4.8-3) the different steps for precedence checking are illustrated.[38]

(4.8) *Recursive precedence checking*

1. linearization regularity:
   COMPLEMENT $\ll$ COMPLEMENT
2. value of the relevant AUX attribute:
   $\langle$[PHON <*a*>], [PHON <*b*>], [PHON <*c*>], [PHON <*d*>]$\rangle$
3. precedence checking:
   a precedes *b*, *c*, *d*
   b precedes *c*, *d*
   c precedes *d*
   d precedes nothing

If one looks at the intended interpretation of the precedence generalization in (4.8-1), it becomes clear that there have to be multiple applications of the precedence checking procedure to verify it. In particular, checking has to occur along the auxiliary list given in (4.8-2). At any depth of recursion in the list it is checked whether the FIRST value's phonology precedes the phonology of all the elements of the REST list on the PHON value of the *lpp-recursive* object.

Example (4.8) illustrates that to encode the LP principle intended by Pollard and Sag (1987), the precedence checking procedure needs to be applied recursively along the auxiliary structure. To express this notion, the sort *lpp-recursive* has to be organized as a recursive sort just as the sort *perm* and in contrast to sorts as *lpp-simple* and *build-phon*. Thus, *lpp-recursive* comes in two species, one representing the base case of an empty auxiliary structure (*lpp-recursive0*), the second representing the recursion step (*lpp-recursive1*). In (4.9-1) the appropriateness conditions of the sort *lpp-recursive* are given, followed by the theory of this sort.

[38](4.8) is a simplified example, since the head daughter of a *head-comp-struc* will occur on the ALL-DTRS list as well but is not given any position on the obliqueness hierarchy. In section 4.7.2, it will be shown, however, how this complication can be overcome.

(4.9) *Exemplifying the sort lpp-recursive*

1. *Appropriateness Conditions for the sort lpp-recursive :*

$$\begin{array}{lll} \textit{lpp-recursive} & \textsc{phon} & \textit{list-of-phons} \\ & \textsc{aux} & \textit{list-of-signs} \end{array}$$

$$\begin{array}{lll} \textit{lpp-recursive0} & \textsc{phon} & \textit{list-of-phons} \\ & \textsc{aux} & \textit{elist-of-signs} \\[1ex] \textit{lpp-recursive1} & \textsc{phon} & \textit{list-of-phons} \\ & \textsc{aux} & \textit{nelist-of-signs} \\ & \textsc{find-a/b} & \textit{find-a/b} \\ & \textsc{prec} & \textit{prec} \\ & \textsc{rec} & \textit{lpp-recursive} \end{array}$$

2. *Theory of lpp-recursive (as AVM):*

$$\begin{bmatrix} \textit{lpp-recursive1} \end{bmatrix} \Longrightarrow$$

$$\begin{bmatrix} \textit{lpp-recursive1} & \\ \textsc{phon} & \boxed{1}\ \textit{list-of-phons} \\ \textsc{aux} & \boxed{2} \begin{bmatrix} \textit{list-of-signs} \\ \textsc{first}\ \textit{sign} \\ \textsc{rest}\ \boxed{3}\ \textit{list-of-signs} \end{bmatrix} \\ \textsc{find-a/b} & \begin{bmatrix} \textit{find-a/b} & \\ \textsc{aux} & \boxed{2}\ \textit{list-of-signs} \\ \textsc{a-elements} & \boxed{4}\ \textit{list-of-phons} \\ \textsc{b-elements} & \boxed{5}\ \textit{list-of-phons} \end{bmatrix} \\ \textsc{prec} & \begin{bmatrix} \textit{prec} & \\ \textsc{phon} & \boxed{1}\ \textit{list-of-phons} \\ \textsc{a} & \boxed{4}\ \textit{list-of-phons} \\ \textsc{b} & \boxed{5}\ \textit{list-of-phons} \end{bmatrix} \\ \textsc{rec} & \begin{bmatrix} \textit{lpp-recursive} & \\ \textsc{phon} & \boxed{1}\ \textit{list-of-phons} \\ \textsc{aux} & \boxed{3}\ \textit{list-of-signs} \end{bmatrix} \end{bmatrix}$$

The first species of *lpp-recusive*, *lpp-recursive0*, is just specified for the PHON list and an empty auxiliary structure. As the AUX list has to be empty, no precedence checking occurs within the *lpp-recursive0* object, and there is no need to give a theory of *lpp-recursive0*.

The second species, *lpp-recursive1*, looks more complex It contains the precedence checking mechanism presented in (4.6) which consists of a search function providing the elements whose precedence relation is to be checked (encoded in the value of FIND-A/B) and the real checking (localized in the value of PREC).[39] In addition to the checking mechanism there is also a recursion expressed in the sort *lpp-recursive1*. This is done via the attribute REC whose value is again of sort *lpp-recursive*. The AUX value of the next recursion step is specified in the theory of *lpp-recursive1* as being one element shorter than the present AUX list.

It is obvious how the LP principle stated in (4.8-1) can be formulated by means of a sort like *lp-recursive*. The AUX list is the list of signs corresponding to the COMPS list. The sort *find-a/b* is defined so that the phonology of the first element of the AUX list ends up in the value of A-ELEMENTS, and that the phonologies of the other elements of the AUX list stand in the value of B-ELEMENTS. Within the attribute PREC, then, precedence would be verified. Thus, the sort *lpp-recursive* encodes a precedence checking procedure that corresponds exactly to the way of checking given in (4.8-3) where the LP principle was illustrated with a short example.

LP principles of the kind of LPP-RECURSIVE can be understood as an application of the precedence checking mechanism along an auxiliary structure. With reference to their recursive definition, we call these LP principles 'recursive'. The sorts encoding recursive LP principles share properties of all sorts that encode LP principles and properties of all recursive sorts. Sorts like *lpp-recursive* have in common with sorts like *lpp-simple* that they both have attributes whose values express a precedence checking mechanism. Furthermore, sorts such as *lpp-recursive* come in at least two species some of which are specified for an attribute whose value is again of sort *lpp-recursive*. As noticed above, the second characteristic is shared by all recursive sorts.

In the present section we have introduced three types of LP principles. A short comparison of them may provide some clarity: Encodings of LP principles of the *restricted simple group* use the ALL-DTRS list instead of any extra auxiliary structure and the precedence checking procedure is applied only once. LP principles of the *simple group* are expressed by making reference to an auxiliary structure, i.e. a restructuring of the ALL-DTRS value. Precedence is checked with respect to such an auxiliary structure, but still by a single application of the checking procedure. In the case of the encoding of LP principles of the *recursive group* an auxiliary structure AUXi is used. Furthermore, the precedence checking mechanism is recursively applied along the structure of AUXi.

As the simple group already is more powerful than the restricted

[39]It is, of course, wrong to use the very sort *find-a/b* in the case of *lpp-recursive* that was used for *lpp-simple* in (4.7), because the elements to identify certainly differ from one LP principle to the other. We have used the same names in both cases just to stress the parallels between the different kinds of LP principles. The sort *prec*, however, is the same in all LP principles.

simple group — and thus more powerful than GPSG LP rules — the recursive group is able to capture a much greater variety of linearization regularities than could be expressed within the classical conception of LP as formulated in Gazdar et al. (1985).

With the discussion of a recursive LP principle all elements of the LP component put forward in this thesis have been presented. We are now in a position to comment on the third restriction of GPSG style LP rules listed in (4.2). During the presentation of the auxiliary structures and the LP principle attributes some remarks concerning this point have already been made which are worth being resumed again. As noted in (4.2-3), according to the classical conception of LP rules, ordering constraints can only be sensitive to inherent properties of the elements that are to be ordered. The auxiliary structures encoded in the values of AUX1, ... AUXn can be used to represent signs whose phonologies are to be ordered with respect to criteria which are not available in the signs themselves. These criteria can be inferred from relations that hold between the signs and, possibly, between the signs on the ALL-DTRS list or between the signs and the overall *phrase*. Examples for such criteria are subcategorization relations and scope inclusion relations. Once the auxiliary structures are made available, properties of the signs that arise from their position within an auxiliary structure can be used to state LP principles. Furthermore, LP principles can make use not only of the noninherent properties of single signs but can even refer to the relative positions of signs in an auxiliary structure. This last possibility is used in recursive LP principles. The massive enrichment of the ways linearization regularities can be stated is not unmotivated. As we have seen, one of the LP rules given in Pollard and Sag (1987) has to be formulated as a recursive LP principle. Thus, the present theory does not introduce LP principles that are more complex than the ones envisioned in the traditional HPSG literature.

In the sections to follow, a formally precise fragment of an LP component for German will be introduced. The presentation parallels the one chosen in this section: first, one species of *lp* is described in section 4.2. Then an external interface attribute is introduced in section 4.3. Two example LP principles (sections 4.4 and 4.5) follow. Section 4.6 gives the theory of the sort *lp* which brings together the results of the preceding sections. The chapter is closed with section 4.7 by a review of what has been achieved and by some proposals for possible extensions of the *lp* object discussed throughout chapter 4.

4.2 The sort *lp-hc*

As the overall architecture of the present theory of LP has been section 4.1.2, it is now possible to illustrate it in a concrete example. The species of *lp* that we pick out for this purpose is the one occurring in phrases that have a DTRS value of the sort *head-comp-struc*. This subsort of *lp* is called *lp-hc*, as introduced in (4.5).

The sort *lp-hc* encodes all the information necessary for an empirically adequate treatment of linearization phenomena in the German mittelfeld. As these are quite numerous only a small part of them will be formalized here, which serve as examples for the shape such a formalization takes in principle. In (4.10) the attributes of a complete *lp-hc* object are presented.

(4.10) *Appropriateness Conditions for the sort lp-hc*

lp-hc	PHON	*list-of-phons*
	WS1	*ws1*
	WS2	*ws2*
	WS3	*ws3*
	WS4	*ws4*
	⋮	
	WSn	*ws-n*
	BASE	*list-of-signs*
	EMBED	*embed*
	SCOPOI	*list-of-signs*
	AUX4	*aux4*
	⋮	
	AUXn	*aux-n*
	LPP1	*lpp1*
	LPP2	*lpp2*
	LPP3	*lpp3*
	⋮	
	LPPm	*lpp-m*

The general function of the attributes in (4.10) are clear from what has been said in the preceding: WS1, ..., WSn are a finite number of external interface attributes such as BUILD-AUX1, ... BUILD-AUXn in (4.6).[40] They are used to construct the auxiliary structures.

In section 4.1.2 three auxiliary structure attributes were shortly introduced. They correspond to the attributes BASE, EMBED and SCOPOI.

[40]No confusion should arise from the fact that the external interface attributes of an *lp-hc* object in (4.10) are called WSi, whereas those on an *lp-example* object in (4.6) are called AUXi. As noted during the presentation of the sort *lp-examples*, the attribute names used will not appear at other species of *lp* as they are chosen for expository purposes only.

In the BASE list the signs collected in the value of ALL-DTRS are ordered parallel to the order of the *synsem* objects on the head daughter's COMPS list. EMBED represents them according to their structure of embedding, i.e. the effects of raising (Pollard and Sag 1994, p. 140) or argument attraction (Hinrichs and Nakazawa 1994, p. 2-4) are undone. The third auxiliary attribute, SCOPOI, is a reordering of the elements of the ALL-DTRS list corresponding to their scope inclusion relations. Other auxiliary structure attributes may be needed as well. They are simply referred to in (4.10) as AUX4, ..., AUXn. It is open to further empirical work whether these attributes and the corresponding external interface WS4, ..., WSn will be needed altogether.

The attributes LPP1, ..., LPPm represent a finite number of LP principle attributes. Their values encode exactly those LP principles that are to apply to a sign specified for *lp-hc* as its LP value.

In this thesis the discussion of the sort *lp-hc* is restricted to only a small part of the attributes in (4.10). The value of just one external interface attribute, WS2, will be given, together with the value of the corresponding auxiliary structure attribute EMBED. From the group of LP principle attributes two are chosen, whose values make use of the value of the auxiliary structure attribute EMBED. The corresponding LP principles are encoded in the sorts *lpp2* and *lpp3*. The sort *lp-hc* is treated in this thesis as if it contained only the attributes in (4.11).

(4.11) *Appropriateness Conditions for the sort lp-hc (revised)*

lp-hc	PHON	*list-of-phons*
	WS2	*ws2*
	EMBED	*embed*
	LPP2	*lpp2*
	LPP3	*lpp3*

The sorts that appear as values of the attributes in (4.11) will be discussed in the following three sections (4.3 - 4.5). The manner of presentation will be similar to the one in chapter 3. In 4.6 the overall structure of the sort *lp-hc* is addressed again and it will be shown in section 4.7 how the technical devices used that far can be exploited to handle further LP principles.

The value of the auxiliary structure attribute EMBED is the core of the *lp* object, thus, one should develop some intuitions about it, before the other parts of the *lp-object* can be presented. As mentioned in sections 4.1.2 the value of the attribute EMBED is to mirror the structure of embedding of the entire phrase. In the following we will give some German examples to demonstrate what such an *embed* object will look like in practice.

The first example represents a coherent construction. The structure of sentences as (4.12-1) has been discussed in chapter 3. In (4.12) the example sentence is given together with its analysis.

(4.12) *An example from German*

1. (Ich glaube nicht, daß...)
   I believe not that
   Peter Maria das Buch gegeben hat
   Peter Maria the book given has
   'I don't believe that Peter has given the book to Maria'

2. *Hierarchical syntactic structure:*[41]

$$
\begin{bmatrix}
\textit{phrase} \\
\vdots \\
\text{LP} \begin{bmatrix} \textit{lp-hc} \\ \text{EMBED } \textit{embed} \end{bmatrix} \\
\vdots \\
\text{DTRS} \begin{bmatrix}
\textit{head-comp-struc} \\
\text{HEAD-DTR } \boxed{5} \begin{bmatrix} \text{PHON} & \langle \textit{hat} \rangle \\ \text{SYNSEM} \ldots \text{VAL} & \begin{bmatrix} \text{SUBJ} & \boxed{1'} \\ \text{COMPS} & \langle \boxed{1'}, \boxed{2'}, \boxed{3'}, \boxed{4'} \rangle \end{bmatrix} \end{bmatrix} \\
\text{COMP-DTRS} \langle \boxed{1} \begin{bmatrix} \text{PHON} & \langle \textit{Peter} \rangle \\ \text{SYNSEM} & \boxed{1'} \end{bmatrix}, \\
\quad \boxed{2} \begin{bmatrix} \text{PHON} & \langle \textit{Maria} \rangle \\ \text{SYNSEM} & \boxed{2'} \end{bmatrix}, \\
\quad \boxed{3} \begin{bmatrix} \text{PHON} & \langle \textit{das, Buch} \rangle \\ \text{SYNSEM} & \boxed{3'} \end{bmatrix}, \\
\quad \boxed{4} \begin{bmatrix} \text{PHON} & \langle \textit{gegeben} \rangle \\ \text{SYNSEM } \boxed{4'} \ldots \text{VAL} & \begin{bmatrix} \text{SUBJ} & \langle \boxed{1'} \rangle \\ \text{COMPS} & \langle \boxed{1'}, \boxed{2'}, \boxed{3'} \rangle \end{bmatrix} \end{bmatrix} \rangle \\
\text{ALL-DTRS} \ \langle \boxed{5}, \boxed{1}, \boxed{2}, \boxed{3}, \boxed{4} \rangle
\end{bmatrix}
\end{bmatrix}
$$

As can be seen in (4.12), both verbs have the NPs on their COMPS list. In the same fashion, the most deeply embedded verb occurs on the COMPS lists of the higher verb. The *embed* object adequate for the phrase in (4.12) restructures the information from the ALL-DTRS list the way given in (4.13), where the tags refer to the same objects as in (4.12).

(4.13) *The structure of embedding:*

[41] The '...' in the AVM in (4.12) and (4.14) stand as abbreviation for the path LOC CAT

$$
\left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{phrase}} \\
\vdots & \\
\text{LP} & \left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{lp-hc}} \\
\text{EMBED} & \left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{embed}} \\
\text{VERB} & \boxed{5}\left[\text{PHON} <\textit{hat}>\right] \\
\text{ARGS} & < \boxed{4} > \\
\text{DEEPER} & \left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{embed}} \\
\text{VERB} & \boxed{4}\left[\text{PHON} <\textit{gegeben}>\right] \\
\text{ARGS} & < \boxed{1}, \boxed{2}, \boxed{3} > \\
\text{DEEPER} & \textit{embed0}
\end{array}\right]
\end{array}\right]
\end{array}\right] \\
\vdots & \\
\text{DTRS} & \left[\begin{array}{l}
\textit{head-comp-struc} \\
\text{ALL-DTRS}\ \textit{list-of-signs}
\end{array}\right]
\end{array}\right]
$$

In (4.13) an *embed* object appears as the LP EMBED value of the *phrase* in (4.12). The value of the LP EMBED attribute of a *phrase* makes it possible to have direct access to the following two bits of information: Firstly, the embedding structure of the verbs that occur in the verbal complex can be seen directly if one follows the path DEEPER (DEEPER ...) VERB. The length of the path indicates the level of embedding. Secondly, within the LP EMBED value, a complement daughter *c* appears as elements of the ARGS list at the same level of embedding as the most deeply embedded verb that has *c*'s SYNSEM value on its COMPS list.

On the 'highest' *embed* object in (4.13), the matrix verb *hat* appears in the VERB value. Its ARGS value only contains the embedded verb (*gegeben*, tagged as $\boxed{4}$). All other elements of the COMPS list of *hat* occur on *gegeben*'s COMPS list as well. Thus, even though the SYNSEM value of the sign $\boxed{2}$ (*Maria*) occurs on all COMPS lists in (4.12), its deepest occurrence is on the COMPS list of(*gegeben*). Therefore the sign tagged as $\boxed{2}$ appears as element of the ARGS list of the *embed* object that takes the sign $\boxed{4}$ as its VERB value.

In (4.13) the matrix verb has only the embedded verb on the ARGS list. All other arguments ended up in the deepest ARGS list of the *embed* object. In other cases, a matrix verb introduces nonverbal arguments of its own. This occurs e.g. in German with a verb like *scheinen* (*seem*), which forms a verbal complex as well. In (4.14) such an example is given, followed by the hierarchical structure and the relevant *embed* object.

(4.14) *A second example from German*

1. (weil) es mir gestimmt zu haben scheint
because it to-me been correct to have seems
'(because) it seems to me to have been correct'

2. *Hierarchical syntactic structure:*

$$\left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{phrase}} \\
\vdots & \\
\text{LP} & \left[\begin{array}{l} \textit{lp-hc} \\ \text{EMBED } \boxed{5}\ \textit{embed} \end{array}\right] \\
\vdots & \\
\text{DTRS} & \left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{head-comp-struc}} \\
\text{HEAD-DTR} & \boxed{6} \left[\begin{array}{ll} \text{PHON} & \langle \textit{scheint} \rangle \\ \text{SYNSEM} \ldots \text{VAL} & \left[\begin{array}{ll} \text{SUBJ} & \langle \boxed{1'} \rangle \\ \text{COMPS} & \langle \boxed{1'}, \boxed{2'}, \boxed{3'}, \boxed{4'} \rangle \end{array}\right] \end{array}\right] \\
\text{COMP-DTRS} & \langle\ \boxed{1} \left[\begin{array}{ll} \text{PHON} & \langle \textit{es} \rangle \\ \text{SYNSEM} & \boxed{1'} \end{array}\right], \\
& \quad \boxed{2} \left[\begin{array}{ll} \text{PHON} & \langle \textit{mir} \rangle \\ \text{SYNSEM} & \boxed{2'} \end{array}\right], \\
& \quad \boxed{3} \left[\begin{array}{ll} \text{PHON} & \langle \textit{gestimmt} \rangle \\ \text{SYNSEM} & \boxed{3'} \ldots \text{VAL} \left[\begin{array}{ll} \text{SUBJ} & \langle \boxed{1'} \rangle \\ \text{COMPS} & \langle \boxed{1'} \rangle \end{array}\right] \end{array}\right], \\
& \quad \boxed{4} \left[\begin{array}{ll} \text{PHON} & \langle \textit{zu haben} \rangle \\ \text{SYNSEM} & \boxed{4'} \ldots \text{VAL} \left[\begin{array}{ll} \text{SUBJ} & \langle \boxed{1'} \rangle \\ \text{COMPS} & \langle \boxed{1'}, \boxed{3'} \rangle \end{array}\right] \end{array}\right] \rangle \\
\text{ALL-DTRS} & \langle \boxed{6}, \boxed{1}, \boxed{2}, \boxed{3}, \boxed{4} \rangle
\end{array}\right]
\end{array}\right]$$

3. *The* LP EMBED *value:*

$$\boxed{5} = \left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{embed}} \\
\text{VERB} & \boxed{6} \left[\text{PHON } \langle \textit{scheint} \rangle \right] \\
\text{ARGS} & \langle \boxed{2}, \boxed{4} \rangle \\
\text{DEEPER} & \left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{embed}} \\
\text{VERB} & \boxed{4} \left[\text{PHON } \langle \textit{zu haben} \rangle \right] \\
\text{ARGS} & \langle \boxed{3} \rangle \\
\text{DEEPER} & \left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{embed}} \\
\text{VERB} & \boxed{3} \left[\text{PHON } \langle \textit{gestimmt} \rangle \right] \\
\text{ARGS} & \langle \boxed{1} \rangle \\
\text{DEEPER} & \textit{embed0}
\end{array}\right]
\end{array}\right]
\end{array}\right]$$

In example (4.14) the matrix verb *scheint* has two ‘native’ arguments, i.e. complements which are not inherited from a subcategorized verb. One is the dative NP (*mir*, tagged as $\boxed{2}$), the second the subcategorized verb (*zu haben*, $\boxed{4}$) from which more arguments are inherited. The intermediate verb *zu haben*, however, has only the next verb (*gestimmt*, $\boxed{3}$) on the ARGS list in the *embed* object, as all other arguments are inherited.

It has now been demonstrated what the needed auxiliary structure, i.e. an the EMBED value of an *lp-hc*, is supposed to look like, but there is still some work to do to establish the desired relation of the value of the ALL-DTRS list to the value of the EMBED attribute. This is accomplished in the value of the external interface attribute WS2 in the section 4.3. The two LP principles discussed in the subsections 4.4 and 4.5 will then show a practical application of the EMBED value for linearization.

4.3 Constructing an auxiliary structure

In this section the external interface attribute WS2 of sort *lp-hc* and the associated sort *ws2* will be formalized in the logic of section 1.2. WS2 denotes a partial function from objects of sort *lp-hc* to objects of sort *ws2*. The theory of *phrase* will restrict the occurrence of *lp-hc* to phrases whose DTRS value is *head-comp-struc*, thereby designating *lp-hc* objects as containing the external interface attributes and LP principles appropriate for head complement structures.

The following AVM provides an overview over the position of the attribute WS2 and the species *ws2* in the architecture of phrases and it shows the relations of *ws2* to the overall phrase.

(4.15) *The placement of ws2 objects in phrase*

$$
\begin{bmatrix}
\textit{phrase} & \\
\textsc{phon} & \boxed{1}\ \textit{list-of-phons} \\
\textsc{build-phon} & \textit{build-phon} \\
\textsc{lp} & \begin{bmatrix}
\textit{lp-hc} & \\
\textsc{ws2} & \begin{bmatrix}
\textit{ws2} & \\
\textsc{base} & \boxed{2}\ \textit{list-of-signs} \\
\textsc{find-verbs} & \textit{find-verbs} \\
\textsc{order} & \textit{order-embed} \\
\textsc{embedding} & \begin{bmatrix}
\textit{embedding} \\
\ldots \\
\textsc{embed}\ \boxed{3}\ \textit{embed}
\end{bmatrix}
\end{bmatrix} \\
\textsc{phon} & \boxed{1}\ \textit{list-of-phons} \\
\textsc{embed} & \boxed{3}\ \textit{embed} \\
\textsc{lpp2} & \textit{lpp2} \\
\textsc{lpp3} & \textit{lpp3}
\end{bmatrix} \\
\textsc{dtrs} & \begin{bmatrix}
\textit{head-comp-struc} & \\
\textsc{head-dtr} & \textit{sign} \\
\textsc{comp-dtrs} & \textit{list-of-signs} \\
\textsc{all-dtrs} & \boxed{2}\ \textit{list-of-signs}
\end{bmatrix}
\end{bmatrix}
$$

As (4.15) shows, *ws2* has a single link to the part of the phrase outside the *lp-hc* object, namely to the ALL-DTRS list. The theory of *ws2* will exploit functional and intrinsic properties of the signs on the ALL-DTRS list for the specification of an object *embed* which is then used by the LP principles of the mittelfeld to constrain the possible PHON lists of the phrase. The structure of *ws2* may vary according to the linguistic analysis at hand. Consequently, the following description of *ws2* objects is specifically geared towards the analysis of German sentence structure presented in section 3.1.

Objects of sort *ws2* come with four attributes which are intuitively named after the functions the associated sorts fulfill. Basically, *find-verbs* searches for the verbal words on the BASE list (which is identical with the ALL-DTRS list), *order-embed* orders the verbs and their complements according to their structure of embedding and *embedding* specifies an object which encodes the structure that was found in an object that can easily be used by the LP principles.

(4.16) *Appropriateness Conditions for the sort ws2*

ws2	FIND-VERBS	*find-verbs*
	ORDER	*order-embed*
	EMBEDDING	*embedding*
	BASE	*list-of-signs*

The easiest way to understand how the ensuing theory of *ws2* works is to think of it in procedural terms. In the value *find-verbs* of the first attribute, FIND-VERBS, a sublist of the BASE list is created that contains every verbal word of the BASE list. Because *ws2* is a sort defined as value of the attribute WS2 in head complement structures, the eventual list of verbs contains all verbs of a the verbal complex, if there is one. Otherwise it just contains the main verb of the sentence.

In the value *order-embed* of the next attribute, ORDER, the list of lists of COMPS values corresponding to the verblist specified by the theory of *find-verbs* is constructed and ordered in such a way that the largest COMPS list comes first, succeeded by the second largest and so on. The resulting ordering of lists directly reflects the structure of embedding. This is also the case for the list of verbs because the verbs on it are ordered in such a way that their position on the list corresponds to the position of their respective COMPS list on the list of lists of complements. The verbs with the longest COMPS list comes first, succeeded by the verb with the second largest COMPS list and so on. In effect, the verbs are ordered according to their level of semantic embedding in the verbal complex.

This particular proceeding is well-defined if one assumes that in coherent constructions every argument of a verb that is deeper embedded, including the embedded verb, is appended to the COMPS list of the immediately governing verb. A syntax of German sentence structure which complies with this assumption was presented in section 3.1 and its predictions for the COMPS lists of verbs in the verbal complex were illustrated in (3.12). It should be noted, though, that nothing hinges upon this specific analysis and the richer syntactic structure of the verbal complex in the alternative analysis of German sentence structure of section 3.2 derived from (Hinrichs and Nakazawa 1994) even provides means to make it easier to derive the ordering of verbs and complements we are aiming at. In line with our general approach, the current analysis starts from a worst case assumption and shows that it does not undermine the general idea of the theory of *ws2*.

Another assumption that comes to bear at this point is the presence of the subject on the COMPS list of infinite verbs which is a consequence of the theory of *val* in (3.5). Only in that specific case one can be sure that the members of the COMPS list of a verb in a coherent construction that are not members of the COMPS list of the governed verb do in fact semantically belong to the verb in question. Again, we do not intend to defend this specific analysis and take the position that the theory of *ws2* could be changed to accommodate differing assumptions.

To summarize the theory of *order-embed* so far, what was constructed were two lists, one containing the verbs ordered according to the level of embedding and the other being a corresponding list of COMPS lists.

The next thing to do in the theory of *order-embed* is to work on the collected and ordered COMPS lists. For each COMPS list we construct a list which contains exactly those elements of the COMPS list which can not be found on the succeeding COMPS list. The succeeding COMPS list belongs to the verb which is one level deeper embedded in the verbal complex. The COMPS list of the verb which is at the deepest level of embedding in the verbal complex is taken over unchanged. Consequently we arrive at a list of lists of the semantic arguments of the verbs, ordered in lists according to the level of embedding of the verbs they belong to.

Within the value *embedding* of the remaining attribute, EMBEDDING, we encode the output of the sort *ws2* in the more conveniently structured object presented in (4.13) above that fits the needs of a perspicuous formalization of LP principles within the *lpp* objects. Most importantly, in this step the synsem objects found on the COMPS lists of the verbs are converted into the corresponding signs in the syntactic structure in order to gain access to the associated phonological values. The conversion is necessary since we assume with Pollard and Sag (1994) that the COMPS list is a list of synsem objects. The main reason for not choosing the simple alternative solution to require that the COMPS list be a list of signs is to illustrate the availability of all components of the sign independent from their actual presence on the COMPS list.

For the conversion of the synsem objects into signs it is presupposed that all corresponding signs are in fact present on the BASE list (which is identical with the ALL-DTRS list). This condition is unconditionally met because we adopt the flat analysis of German sentence structure of section 3.1 as syntactic basis of the auxiliary object *ws2*.

The *embed* object we finally arrive at is structure-shared with the value of the EMBED attribute of *lp-hc* as illustrated in (4.15). It provides the input for the LP principles.

The rest of section 4.3 is organized as follows. In sections 4.3.1 -

4.3.3 the theories for the sorts we have just introduced and informally explained will be defined. After the subparts of each sort have been defined there will a proposition about the properties of objects of the respective sort and a proof of the proposition at the end of each section. Therefore all three sections that follow are rather technical and will not provide anything new beyond what has already been said except for the formalization. In section 4.3.4 the three components of the theory of *ws2* will finally be put together, and in a last proof we show that our claims about the properties of *ws2* objects hold.

4.3.1 The sort *find-verbs*

Let us now turn to the sort used as value of the first attribute of the *ws2* object, to the sort *find-verbs*. Objects of this sort take the BASE list of *ws2* objects as value of their own BASE attribute and extract all lexical verbs contained in it. Since we are not concerned with verbal arguments that are realized as full VPs such as incoherent constructions or sentential complements, only verbal elements that are lexical signs, i.e., of sort *word*, have to be identified.

The extraction of all verbs of the verbal complex from the BASE list presupposes the presence of every verb of a coherent construction on the BASE list. As we have seen in chapter 3 this assumption is not necessarily linked to a flat analysis of the verbal complex. An alternative option to an analysis that treats infinite verbs as sisters of each other is to regard them as occurring in a binary branching verbal complement of the finite verb as it is done in section 3.2 in an analysis based on ideas of (Hinrichs and Nakazawa 1994). For the applicability of the theory of *find-verbs* in its present form it then has to be made sure that the verbs are projected into the ALL-DTRS list. In section 3.2 we have shown how this can be formalized. Once this has been done it is a direct consequence of our formalization that all verbs of the verbal complex turn up in the BASE list of *ws2* objects, from which they are passed up into the theory of *find-verbs* by term equation in the theory of *ws2*.

Before we proceed with the appropriateness conditions and the theory of *find-verbs*, let us shortly consider the structure of *find-verbs* and what lists its attributes are supposed to contain.

(4.17) *The structure of find-verbs*

$$\begin{bmatrix} \textit{find-verbs1} & \\ \text{BASE} & \textit{list-of-signs} \\ \text{VERBL} & \textit{list-of-signs} \\ \text{REC} & \textit{find-verbs} \end{bmatrix}$$

The BASE list supplies the input for the theory of *find-verbs* and defines the domain of its application. The recursive theory of *find-verbs* simply collects the verbs which are contained in the BASE list in a new list which is the value of the attribute VERBL.

Objects of sort *find-verbs* come in three different species which account for different states of the BASE list. (4.18) presents the appropriateness conditions.

(4.18) *Appropriateness Conditions for the sort find-verbs*

find-verbs	BASE	*list-of-signs*
	VERBL	*list-of-signs*
find-verbs0	BASE	*elist*
	VERBL	*elist*
find-verbs1	BASE	*nelist-of-signs*
	VERBL	*list-of-signs*
	REC	*find-verbs*
find-verbs2	BASE	*nelist-of-signs*
	VERBL	*list-of-signs*
	REC	*find-verbs*

There are three cases that can be distinguished. In the first one, treated by *find-verbs0*, the BASE list is empty. Since there is nothing to be found on an empty list, the VERBL list is empty, too.

The remaining two cases concern the nonempty BASE list and make use of recursion. The idea is to examine the first element on BASE. If it is a verbal word then it is identical with the first element on VERBL, and the rest of BASE and VERBL are passed on to the recursion step. If the first element on BASE in not verbal or not a word then we go into recursion with the rest of BASE and the complete VERBL, i.e., the first element on BASE is not on VERBL.

The theory of *find-verbs* is formalized in (4.19).

(4.19) *Theory of find-verbs*

$$
\begin{array}{ll}
{:}{\sim}\textit{find-verbs1} & \rightarrow \\
& (\text{:BASE FIRST SYNSEM LOC CAT HEAD} \sim \textit{verb} \\
& \wedge \text{ :BASE FIRST} \sim \textit{word} \\
& \wedge \text{ :VERBL FIRST} \approx \text{:BASE FIRST} \\
& \wedge \text{ :REC BASE} \approx \text{:BASE REST} \\
& \wedge \text{ :REC VERBL} \approx \text{:VERBL REST})
\end{array}
$$

as AVM:

$$\begin{bmatrix} \textit{find-verbs1} & \end{bmatrix} \Longrightarrow$$

$$\left[\begin{array}{ll} \textit{find-verbs1} & \\ \text{BASE} & \left[\begin{array}{ll} \textit{nelist-of-signs} & \\ \text{FIRST} & \boxed{1}\left[\begin{array}{l}\textit{word}\\ \text{SYNSEM LOC CAT HEAD } \textit{verb}\end{array}\right] \\ \text{REST} & \boxed{2}\ \textit{list-of-signs}\end{array}\right] \\ & \\ \text{VERBL} & \left[\begin{array}{ll} \textit{nelist-of-signs} & \\ \text{FIRST} & \boxed{1}\ \textit{word} \\ \text{REST} & \boxed{3}\ \textit{list-of-signs}\end{array}\right] \\ & \\ \text{REC} & \left[\begin{array}{ll} \textit{find-verbs} & \\ \text{BASE} & \boxed{2}\ \textit{list-of-signs} \\ \text{VERBL} & \boxed{3}\ \textit{list-of-signs}\end{array}\right] \end{array}\right]$$

$$\begin{array}{ll} :\sim \textit{find-verbs2} \quad \rightarrow & \\ & (\neg(:\text{BASE FIRST SYNSEM LOC CAT HEAD} \sim \textit{verb} \\ & \quad \wedge\ :\text{BASE FIRST} \sim \textit{word}) \\ & \wedge\ :\text{REC BASE} \approx\ :\text{BASE REST} \\ & \wedge\ :\text{REC VERBL} \approx\ :\text{VERBL}) \end{array}$$

In the first description (*find-verbs1*) of (4.19) we treat the case in which the first element on the BASE list is a lexical verb. It is added to the list of verbs in VERBL. The second description (*find-verbs2*) takes care of those cases in which no lexical verb is present. Then the VERBL list remains unchanged and the next element of the BASE list is considered.

In order to make the proposition and proof for the theory of *find-verbs* more readable, we introduce definitions which say what it is for an object to be a verb and what it is for an object to be an element of some finite list.

Def 1 *Verb*

v **is a verb** iff

$S([\text{SYNSEM LOC CAT HEAD}]u) \sqsubseteq \textit{verb}$.

Def 2 *membership in a list*

for any object o of a given sort *sort*, and any finite list $l = < o_1, \ldots, o_n >$ of objects of the sort *sort*,

o **is an element of** l iff

there is an $i, 1 \leq i \leq n$ such that $o = o_i$.

Def 1 allows us to refer directly to objects which are verbs. Def 2 allows us to refer to an object as being an element of a certain list. We use these definitions in our proposition for the sort *find-verbs*.

PROPOSITION
If u is a *find-verbs* and [BASE]u is a finite list then [VERBL]u consists exactly of those v which are an element of [BASE]u, a verb and for which it holds that S(v) = *word*.

PROOF
(by induction on the length of [BASE]u)

Base:
For each u such that S(u) $\sqsubseteq$ *find-verbs*, if [BASE]u = $<>$, then S(u) = *find-verbs0*. Therefore S([VERBL]u) = *elist*. Hence the proposition trivially holds for u.

Hypothesis:
Suppose we have shown for each u with S(u) $\sqsubseteq$ *find-verbs* that if [BASE]u = $< u_1, \ldots, u_n >$ is of length n then [VERBL]u consists exactly of those v which are an element of [BASE]u, a verb and for which it holds that S(v) = *word*.

Step:
For each u such that S(u) $\sqsubseteq$ *find-verbs*, if [BASE]u = $< u_1, \ldots, u_{n+1} >$, then either S(u) = *find-verbs1* or S(u) = *find-verbs2*.

Case 1: S(u) = *find-verbs1*
By definition it holds that S(u_1) = *word*, u_1 is a verb and [VERBL FIRST]u = u_1. Furthermore, we know by definition of *find-verbs1* that [BASE REST]u = $< u_2, \ldots, u_{n+1} >$ = [REC BASE]u of length n. Since S([REC]u) = S(u') $\sqsubseteq$ *find-verbs* the hypothesis applies to [BASE]u' = [REC BASE]u = $< u_2, \ldots, u_{n+1} >$. This tells us that [REC VERBL]u consists exactly of those v which are an element of [REC BASE]u, a verb and for which it holds that S(v) = *word*. With [REC VERBL]u = [VERBL REST]u the proposition holds for u.

Case 2: S(u) = *find-verbs2*
In this case we know by definition that S(u_1) $\neq$ *word* or S([SYNSEM LOC CAT HEAD]u_1) $\neq$ *verb*. Moreover, [BASE REST]u = $< u_2, \ldots, u_{n+1} >$ = [REC BASE]u of length n. Since S([REC]u) = S(u') $\sqsubseteq$ *find-verbs* the hypothesis applies to [BASE]u' = [REC BASE]u = $< u_2, \ldots, u_{n+1} >$ and [REC VERBL]u consists exactly of those v which are an element of [REC BASE]u, a verb and for which it holds that S(v) = *word*. Since [REC VERBL]u = [VERBL]u we know that u_1 which is not a verb or not of sort *word* is not in [VERBL]u. Hence the proposition holds for u.

Because an element on [BASE]u is either a verb and of sort *word* or it is not a verb or not a *word* the two relevant cases have been considered. The proposition follows by induction. ■

With the theory of *find-verbs* a list of all verbs in the verbal complex is specified which can be employed by other theories. Objects of sort *order-embed* will use the list of verbs specified by *find-verbs* and arrange the complements of the verbs in new lists according to certain criteria.

4.3.2 The sort *order-embed*

Next we turn to the attribute ORDER of *ws2* and define its value *order-embed*. *order-embed* objects are more complex than *find-verbs* objects which provide their input. The overall picture that will finally emerge looks as follows:

(4.20) *The structure of order-embed*

$$
\left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{order-embed}} \\
\text{PERMVERBS} & \left[\begin{array}{ll} \multicolumn{2}{l}{\textit{perm}} \\ \text{ORIGIN} & \textit{list-of-signs} \\ \text{PERMUTED} & \boxed{1}\ \textit{list-of-signs} \end{array}\right] \\
\text{CHECK} & \left[\begin{array}{ll} \multicolumn{2}{l}{\textit{check}} \\ \text{TO-CHECK} & \boxed{2}\ \textit{list-of-lists-of-synsems} \\ \text{DIFFL} & \textit{list-of-lists-of-synsems} \\ \text{TEST} & \textit{subset} \end{array}\right] \\
\text{COMPSVERB} & \left[\begin{array}{ll} \multicolumn{2}{l}{\textit{compsverb}} \\ \text{VERBL} & \boxed{1}\ \textit{list-of-signs} \\ \text{COMPSL} & \boxed{2}\ \textit{list-of-list-of-synsems} \end{array}\right]
\end{array}\right]
$$

order-embed has three components. The attribute PERMVERBS utilizes the sort *perm* defined in (2.4) and (2.7) of chapter 2 to supply an arbitrary permutation of signs. The theory of *ws2* will see to it that the list of signs the permutations are defined for is the list of verbs delivered by *find-verbs*. The permutations of the verblist are needed, since we are interested in one specific ordering of the verbs (and their complements) but do not know in the general case in which order the verbs that are supplied by *find-verbs* come.

The attribute COMPSVERB has objects of the new sort *compsverb* as its value. The purpose of *compsverb* is to specify a list of the lists complements of the verbs on verblist such that the verblist and the list of lists of complements are in corresponding order. Because the value of PERMVERBS PERMUTED and COMPSVERB VERBL are structure-shared, the combination of the theories of *perm* and *compsverb* provides all possible permutations of the list of verbs and the corresponding list of lists of complements.

The final component of *order-embed*, the theory of *check*, is designed to pick out the specific permutation of the list of lists of complements we are interested in. This is the permutation which directly reflects the level of embedding of the verbs associated with the COMPS lists. At this point, certain syntactic assumptions of section 3.1 come into play.

Since we assumed that every complement of a governed verb in a coherent construction is inherited by the list of complements of the governing verb including the verbs the governed verbs might in turn govern, we know that the list of complements of a governing verb is always at least one element longer than that of the governed verb and contains all elements of the list of complements of the governed verb. Therefore the structure of embedding can be found by ordering the lists of complements in such a way that each list contains every element of its succeeding lists. In terms of the length of the lists, the longest list always belongs to the finite verb, the shortest one to the main verb. The theory of *check* selects the ordering of the COMPS lists in which the elements of each succeeding list are contained in the list preceding it. Since the order of the list of verbs and the list of their complements are linked to each other by the theory of *compsverb*, we can be sure that the list of verbs will also be in the desired order.

Having specified the order of COMPS lists as an order of lists of decreasing length, it is easy to determine the elements by which any two immediately succeeding lists differ. This way we arrive at a list of native complements for each verb v, i.e., those complements on the verb's COMPS list which do not occur on the COMPS list of any other verb v' which is on the COMPS list of v. The emerging list of lists of native complements of the verbs is contained in CHECK DIFFL. In other words, we can say that CHECK DIFFL provides a list of lists of synsems which directly reflects which synsem objects originate from the COMPS list of which verb.

Before we can turn to the definition of the two new sorts *compsverb* and *check* introduced by *order-embed*, the appropriateness conditions of *order-embed* have to be stated formally:

(4.21) *Appropriateness Conditions for the sort order-embed*

order-embed	PERMVERBS	*perm*
	CHECK	*check*
	COMPSVERB	*compsverb*

In the following paragraphs we will introduce the appropriateness conditions and theories of *compsverb* and *check*. Then we will come back to the sort *order-embed* and formalize its theory.

Two attributes are defined on *compsverb*. The first one is called VERBL and contains a list of signs (which are the verbs of the BASE list of *ws2*). The second one is called COMPSL and is supposed to contain a list of lists of synsems which are the lists of complements of the verbs on VERBL. This idea leads to the following appropriateness conditions.

(4.22) *Appropriateness Conditions for the sort compsverb*

compsverb	VERBL	*list-of-signs*
	COMPSL	*list-of-lists-of-synsems*

compsverb0	VERBL	*elist*
	COMPSL	*elist*
compsverb1	VERBL	*nelist-of-signs*
	COMPSL	*nelist-of-lists-of-synsems*
	REC	*compsverb*

In the theory of *compsverb* we can distinguish two cases. *compsverb0* is the base case in which VERBL is empty. Then the list of lists of complements is also empty.

In the nontrivial case, *compsverb1*, there is at least one verb on VERBL. The obvious thing to do is to take the verb's COMPS list, put it into the first position of COMPSL and go into the recursion with the rest of VERBL and COMPSL in order to repeat the process. In the description language this looks as follows:

(4.23) *Theory of compsverb*

$$
\begin{array}{l}
:\sim \mathit{compsverb1} \quad \longrightarrow \\
\quad (:\text{VERBL FIRST SYNSEM LOC CAT VAL COMPS} \approx :\text{COMPSL FIRST} \\
\quad \wedge\ :\text{REC VERBL} \approx :\text{VERBL REST} \\
\quad \wedge\ :\text{REC COMPSL} \approx :\text{COMPSL REST})
\end{array}
$$

as AVM:

$$
\begin{bmatrix} \mathit{compsverb1} \end{bmatrix} \Longrightarrow
\begin{bmatrix}
\mathit{compsverb1} & \\
\text{VERBL} & \begin{bmatrix} \mathit{list\text{-}of\text{-}signs} \\ \text{FIRST SYNSEM LOC CAT VAL COMPS } \boxed{1} \\ \text{REST } \boxed{2}\ \mathit{list\text{-}of\text{-}signs} \end{bmatrix} \\
\text{COMPSL} & < \boxed{1}\ \mathit{list\text{-}of\text{-}synsems} \mid \boxed{3} > \\
\text{REC} & \begin{bmatrix} \mathit{compsverb} \\ \text{VERBL } \boxed{2}\ \mathit{list\text{-}of\text{-}signs} \\ \text{COMPSL } \boxed{3}\ \mathit{list\text{-}of\text{-}lists\text{-}of\text{-}synsems} \end{bmatrix}
\end{bmatrix}
$$

Note that in the following proposition for the theory of *compsverb* no reference is made to the syntactic category of the signs on VERBL. The theory of *compsverb* is fit to specify lists of lists of complements for signs of any syntactic category.

PROPOSITION
If u is a *compsverb* and [VERBL]$u = < u_1, \ldots, u_m >$ then [COMPSL]$u =$ $<$ [SYNSEM LOC CAT VAL COMPS]u_1,..., [SYNSEM LOC CAT VAL COMPS]$u_m >$.

PROOF
(by induction on the length of [VERBL]u)

Base:
For each u such that S(u) $\sqsubseteq$ *compsverb*, if [VERBL]u = <>, then S(u) = *compsverb0*, therefore S([COMPSL]u) = *elist*. Thus the proposition trivially holds for for u.

Hypothesis:
Assume that we have shown for each u with S(u) $\sqsubseteq$ *compsverb*, if [VERBL]u $= < u_1, \ldots, u_n >$ is of length n then it is the case that [COMPSL]$u =$ $< Cl_1$,..., $Cl_n >$, where Cl_i = [SYNSEM LOC CAT VAL COMPS]u_i for each i with $1 \leq i \leq n$.

Step:
For each u such that S(u) $\sqsubseteq$ *compsverb*, if $m = n + 1$ and [VERBL]$u =$ $< u_1, \ldots, u_{n+1} >$, then we know that S(u) = *compsverb1*. By definition we know that [VERBL FIRST SYNSEM LOC CAT VAL COMPS]u = [COMPSL FIRST]$u = u_1$ and [REC VERBL]u = [VERBL REST]$u = < u_2, \ldots, u_{n+1} >$ of length n. Let [SYNSEM LOC CAT VAL COMPS]$u_1 = Cl_1$. Furthermore the theory of *compsverb1* tells us that [REC COMPSL]u = [COMPSL REST]u. Since S([REC]u) = S(u') $\sqsubseteq$ *compsverb* the hypothesis applies to [REC VERBL]u = [VERBL]u' and we get [REC COMPSL]u = < [SYNSEM LOC CAT VAL COMPS]u_2,...,[SYNSEM LOC CAT VAL COMPS]u_{n+1} > = $< Cl_2$,..., $Cl_{n+1} >$ = [COMPSL REST]u. It follows that [COMPSL]$u = < Cl_1$,..., $Cl_{n+1} >$ for $m = n + 1$.

The proposition follows by induction. ∎

The second attribute introducing a new sort in the appropriateness conditions of *order-embed* (4.21) is CHECK. Its value *check* takes the list of lists of complements specified by the theory of *compsverb* as input and admits only the order of lists in which they are ordered with decreasing length. At the same time it determines for each pair of immediately succeeding lists by which elements they differ. The structure of *check* is stated in (4.24) and its theory will follow in (4.27).

(4.24) *Appropriateness Conditions for the sort check*

check	TO-CHECK	*list-of-lists-of-synsems*
	DIFFL	*list-of-lists-of-synsems*

check0	TO-CHECK	*elist*
	DIFFL	*elist*
check1	TO-CHECK	*nelist-of-lists-of-synsems*
	DIFFL	*list-of-lists-of-synsems*
	REC	*check0*
check2	TO-CHECK	*nelist-of-lists-of-synsems*
	DIFFL	*list-of-lists-of-synsems*
	TEST	*subset*
	REC	*check*

check takes a *list-of-lists-of-synsems* as value of its TO-CHECK attribute. The theory of *order-embed* will structure-share the TO-CHECK list with the list of COMPS lists delivered by the theory of *compsverb*. The theory of *check* employs the sort *subset* which is the value of TEST in order to make sure that for two succeeding lists on TO-CHECK the one that comes first is at least as long as the one succeeding it. Furthermore, *subset* determines the list of elements by which the two lists differ. What is effectively done by the theory of *check* can be described as follows: It guarantees that the list of COMPS lists in TO-CHECK are arranged in such a way that for any two COMPS lists l_1 and l_2 on that list it is the case that if l_1 precedes l_2, then l_1 contains at least all elements of l_2. The case that two lists l_1 and l_2 contain exactly the same elements, i.e., they are of equal length, does not occur in our fragment of German as defined in section 3.1, since argument inheritance and the properties of verbs entail that the COMPS list of a governing verb in the verbal complex is always at least one element longer than the COMPS list of the verb it governs. For our fragment we therefore know that there is always a unique ordering of COMPS lists according to decreasing length.

The theory of *check* distinguishes three cases. *check0* is the base case. Both the TO-CHECK and the DIFFL list are empty. *check1* handles the case in which there is only one list on the TO-CHECK list. It is identical with the single list on DIFFL, and no differences between lists must be specified. In *check2* there are at least two lists l_1 and l_2 on TO-CHECK. Then it has to be checked that l_1 contains at least all elements of l_2 and the list l_3 by which l_1 and l_2 differ is put into DIFFL.

The sort responsible for the actual checking and specification of l_3 is the sort *subset*. In order to fully specify *check* we therefore have to present the definition of *subset* first. The appropriateness conditions for this sort are stated in (4.25), and its theory follows in (4.26).

(4.25) *Appropriateness Conditions for the sort subset*

subset	BIG	*list-of-synsems*	
	SMALL	*list-of-synsems*	
	DIFF	*list-of-synsems*	
subset0	BIG	*list-of-synsems*	
	SMALL	*elist*	
	DIFF	*list-of-synsems*	
subset1	BIG	*nelist-of-synsems*	
	SMALL	*nelist-of-synsems*	
	DIFF	*list-of-synsems*	
	TAKE-OUT	*take-out''*	
	REC	*subset*	

The sort *take-out''* which is the value of the attribute TAKE-OUT is defined exactly as the sort *take-out* in (2.5) and (2.6) except that we are assuming synsem objects now where we used to have signs. Consequently *take-out''* has three attributes: a SET list (*nelist-of-synsems*), a NSET list (*list-of-synsems*) and an ELE element (of sort *synsem*). NSET differs from SET only by ELE which is a member of SET but not of NSET. Since specification and proof of *take-out''* are entirely parallel to the specification and proof of *take-out*, we will not carry them out and simply suppose that it can be done.[42]

As can be seen in the appropriateness conditions (4.25), objects of sort *subset* have three list-valued attributes called BIG, SMALL and DIFF. All elements of the SMALL list have to be members of the BIG list, which additionally contains all elements that occur on the DIFF list. The BIG list contains exactly the elements of the SMALL list and the DIFF list without necessarily being a concatenation of the two. We say that DIFF is the list of elements by which BIG and SMALL differ.
We are now ready for the theory of *subset*.

(4.26) *Theory of subset*

$$:\sim subset0 \quad \rightarrow \quad (:\text{BIG} \approx :\text{DIFF})$$

as AVM:

$$\begin{bmatrix} \textit{subset0} \end{bmatrix} \Longrightarrow \begin{bmatrix} \textit{subset0} & \\ \text{BIG} & \boxed{1}\ \textit{list-of-synsems} \\ \text{SMALL} & <> \\ \text{DIFF} & \boxed{1}\ \textit{list-of-synsems} \end{bmatrix}$$

[42] *take-out''* is identical with the sort *take-out-sp* which is used in section 3.2 for the definition of the Subcategorization Principle for a binary branching sentence structure.

$$
\begin{array}{lll}
:\!\sim\! \textit{subset1} & \rightarrow & (\text{:TAKE-OUT SET} \approx \text{:BIG} \\
 & & \wedge\ \text{:TAKE-OUT ELE} \approx \text{:SMALL FIRST} \\
 & & \wedge\ \text{:REC BIG} \approx \text{:TAKE-OUT NSET} \\
 & & \wedge\ \text{:REC SMALL} \approx \text{:SMALL REST} \\
 & & \wedge\ \text{:REC DIFF} \approx \text{:DIFF})
\end{array}
$$

as AVM:

$$
\begin{bmatrix} \textit{subset1} \end{bmatrix} \Longrightarrow
\begin{bmatrix}
\textit{subset1} & \\
\text{BIG} & \boxed{1}\ \textit{list-of-synsems} \\
\text{SMALL} & <\ \boxed{2}\ \textit{synsem} \mid \boxed{4}\ > \\
\text{DIFF} & \boxed{5}\ \textit{list-of-synsems} \\
\text{TAKE-OUT} & \begin{bmatrix} \textit{take-out}'' & \\ \text{SET} & \boxed{1}\ \textit{nelist-of-synsems} \\ \text{ELE} & \boxed{2}\ \textit{synsem} \\ \text{NSET} & \boxed{3}\ \textit{list-of-synsems} \end{bmatrix} \\
\text{REC} & \begin{bmatrix} \textit{subset} & \\ \text{BIG} & \boxed{3}\ \textit{list-of-synsems} \\ \text{SMALL} & \boxed{4}\ \textit{list-of-synsems} \\ \text{DIFF} & \boxed{5}\ \textit{list-of-synsems} \end{bmatrix}
\end{bmatrix}
$$

subset0 is the base case. If SMALL is empty, then BIG and DIFF are identical. If SMALL is nonempty, then we are dealing with a *subset1*. In this case the first occurrence of an arbitrary element on BIG is identical with the first element of SMALL. The BIG list, shortened by the first element on SMALL, the rest of SMALL and the entire DIFF list are passed on to the recursion step.

PROPOSITION
If u is a *subset* and $[\text{SMALL}]u = < w_1, \ldots, w_k >$ then

1. for each w which is an element of $[\text{SMALL}]u = < w_1, \ldots, w_k >$, w is also an element of $[\text{BIG}]u$.

2. for each v which is an element of $[\text{DIFF}]u$, v is also an element of $[\text{BIG}]u$.

3. for each x which is an element of $[\text{BIG}]u$, x is either an element of $[\text{SMALL}]u$ or an element of $[\text{DIFF}]u$.

4. the length of $[\text{BIG}]u$ equals the length of $[\text{SMALL}]u$ plus the length of $[\text{DIFF}]u$.

For any lists of synsem objects *list*1 and *list*2 such that [BIG]u = *list*1, [SMALL]u = *list*2 with u a *subset* we say that *list*3 = [DIFF]u is the list of elements by which *list*1 and *list*2 differ.

Claim (4) of the proposition is necessary to guarantee that elements on SMALL and DIFF can not occur several times on these lists unless they do so on BIG.

PROOF
(by induction on the length of [SMALL]u)

Base:
For each u such that S(u) $\sqsubseteq$ *subset*, if [SMALL]u = <> then it follows by definition that S(u) = *subset0* and [BIG]u = [DIFF]u. Thus (1) trivially holds for the empty list [SMALL]u = <>, (2) and (3) are fulfilled since [DIFF]u = [BIG]u. (4) is true because the length of [SMALL]u is 0 by assumption and the length of [BIG]u equals the length of [DIFF]u. Since all four claims are met the proposition holds for u.

Hypothesis:
Suppose we have shown for each u with S(u) $\sqsubseteq$ *subset* that if [SMALL]u $= < w_1, \ldots, w_n >$ is of length n, then the proposition holds for u.

Step:
For each u such that S(u) $\sqsubseteq$ *subset*, if [SMALL]u $= < w_1, \ldots, w_{n+1} >$ is of length $(n+1)$ then it follows that S(u) = *subset1*. Therefore the following equations hold by definition: [TAKE-OUT SET]u = [BIG]u, [TAKE-OUT ELE]u = [SMALL FIRST]u = w_1 and [TAKE-OUT NSET]u = [REC BIG]u. It follows by the theory of *take-out*$''$ that w_1 is in [BIG]u but not in [REC BIG]u. Since [SMALL REST]u $= < w_2, \ldots, w_{n+1} >$ = [REC SMALL]u of length n, w_1 is not in [REC SMALL]u, either.
Furthermore S([REC]u) = S(u') $\sqsubseteq$ *subset* by definition, thus the hypothesis applies to [SMALL]u' = [REC SMALL]u and we know that

1. for each w which is an element of [REC SMALL]u $= < w_2, \ldots, w_{n+1} >$, w is also an element of [REC BIG]u.

2. for each v which is an element of [REC DIFF]u, v is also an element of [REC BIG]u.

3. for each x which is an element of [REC BIG]u, x is either an element of [REC SMALL]u or an element of [REC DIFF]u.

4. the length of [REC BIG]u equals the length of [REC SMALL]u plus the length of [REC DIFF]u.

By definition we know [REC DIFF]u = [DIFF]u, thus `length`([REC DIFF]u) = `length`([DIFF]u). Since we have shown that both [BIG]u and [SMALL]u differ from [REC BIG]u and [REC SMALL]u respectively only by the absence of w_1 in the latter lists, while w_1 is an element of both [BIG]u and [SMALL]u it follows that
(a) the length of [BIG]u equals the length of [SMALL]u plus the length of [DIFF]u and
(b) each w which is an element of [SMALL]u is also an element of [BIG]u.
Thus the claims (4) and (1) of the proposition are met. In addition, it's clear that for every v which is an element of [DIFF]u = [REC DIFF]u, v is also an element of [BIG]u, since [BIG]u differs from [REC BIG]u only by w_1, which is in [SMALL]u. Thus claim (2) of the proposition is met. Since w_1 is an element of [SMALL]u and [REC BIG]u equals [BIG]u except that the element w_1 which is on [BIG]u is missing, we know by hypothesis that each x which is on [BIG]u is either an element of [SMALL]u or an element of [DIFF]u. Thus claim (4) of the proposition is met.
Hence we have shown the proposition for u.

The proposition follows by induction. ∎

We can now use *subset* and the notion of two lists differing by a third list to define the theory of *check*.

(4.27) *Theory of check*

$$
\begin{array}{lll}
:\sim \mathit{check1} & \rightarrow & (:\text{TO-CHECK REST} \sim \mathit{elist} \\
 & & \wedge\ :\text{DIFFL FIRST} \approx :\text{TO-CHECK FIRST} \\
 & & \wedge\ :\text{DIFFL REST} \approx :\text{REC DIFFL} \\
 & & \wedge\ :\text{REC TO-CHECK} \approx :\text{TO-CHECK REST})
\end{array}
$$

as AVM:

$$
\begin{bmatrix} \mathit{check1} \end{bmatrix} \Longrightarrow
$$

$$
\begin{bmatrix}
\mathit{check1} & \\
\text{TO-CHECK} & \begin{bmatrix} \mathit{nelist\text{-}of\text{-}lists\text{-}of\text{-}synsems} \\ \text{FIRST}\ \boxed{1}\ \mathit{list\text{-}of\text{-}synsems} \\ \text{REST}\ \boxed{3}\ \mathit{elist} \end{bmatrix} \\
\text{DIFFL} & \begin{bmatrix} \mathit{list\text{-}of\text{-}lists\text{-}of\text{-}synsems} \\ \text{FIRST}\ \boxed{1}\ \mathit{list\text{-}of\text{-}synsems} \\ \text{REST}\ \boxed{2}\ \mathit{list\text{-}of\text{-}lists\text{-}of\text{-}synsems} \end{bmatrix} \\
\text{REC} & \begin{bmatrix} \mathit{check} \\ \text{TO-CHECK}\ \boxed{3} \\ \text{DIFFL}\ \boxed{2} \end{bmatrix}
\end{bmatrix}
$$

$$
\begin{array}{lll}
:\sim \mathit{check2} & \rightarrow & (\text{:TO-CHECK FIRST} \approx \text{:TEST BIG} \\
& & \wedge\ \text{:TO-CHECK REST FIRST} \approx \text{:TEST SMALL} \\
& & \wedge\ \text{:DIFFL FIRST} \approx \text{:TEST DIFF} \\
& & \wedge\ \text{:DIFFL REST} \approx \text{:REC DIFFL} \\
& & \wedge\ \text{:REC TO-CHECK} \approx \text{:TO-CHECK REST})
\end{array}
$$

as AVM:

$$
\left[\mathit{check2}\quad\right] \Longrightarrow
$$

$$
\left[\begin{array}{ll}
\mathit{check2} & \\
\text{TO-CHECK} & \left[\begin{array}{ll} \mathit{nelist\text{-}of\text{-}lists\text{-}of\text{-}synsems} & \\ \text{FIRST} & \boxed{1}\ \mathit{list\text{-}of\text{-}synsems} \\ \text{REST} & \boxed{5}\ \left[\text{FIRST}\quad \boxed{2}\right] \end{array}\right] \\
\text{DIFFL} & \left[\begin{array}{ll} \mathit{list\text{-}of\text{-}lists\text{-}of\text{-}synsems} & \\ \text{FIRST} & \boxed{3}\ \mathit{list\text{-}of\text{-}synsems} \\ \text{REST} & \boxed{4}\ \mathit{list\text{-}of\text{-}lists\text{-}of\text{-}synsems} \end{array}\right] \\
\text{TEST} & \left[\begin{array}{ll} \mathit{subset} & \\ \text{BIG} & \boxed{1}\ \mathit{list\text{-}of\text{-}synsems} \\ \text{SMALL} & \boxed{2}\ \mathit{list\text{-}of\text{-}synsems} \\ \text{DIFF} & \boxed{3}\ \mathit{list\text{-}of\text{-}synsems} \end{array}\right] \\
\text{REC} & \left[\begin{array}{ll} \mathit{check} & \\ \text{TO-CHECK} & \boxed{5}\ \mathit{list\text{-}of\text{-}lists\text{-}of\text{-}synsems} \\ \text{DIFFL} & \boxed{4}\ \mathit{list\text{-}of\text{-}lists\text{-}of\text{-}synsems} \end{array}\right]
\end{array}\right]
$$

In the first case, *check1*, only one COMPS list is on TO-CHECK. No checking of length takes place and the COMPS list on TO-CHECK is identical with the single list on the DIFFL list. This case treats the complements of the main verb of the sentence.

In *check2*, where we have at least two COMPS lists on TO-CHECK, the length of two lists has to be compared and the list by which they differ must be on DIFFL. For this purpose the first two COMPS lists l_1 and l_2 of TO-CHECK are taken. The first one is structure-shared with the BIG list and the second one with the SMALL list of *subset*. By the properties of *subset* we then know that its DIFFL list contains a list l_3 by which l_1 and l_2 differ and put l_3 into the first position of DIFFL. Since l_1 equals the BIG list of *subset*, it is at least as long as l_2 which equals SMALL. In our grammar l_1 is always longer than l_2 as explained above.

The recursion is done with the rest of TO-CHECK and DIFFL.

PROPOSITION

If u is a *check* and [TO-CHECK]$u = < l_1, \ldots, l_m >$ is a finite list of lists of synsems then

1. for each pair of lists l_i, l_{i+1} such that l_i and l_{i+1} are elements of [TO-CHECK]u, it is the case that $\texttt{length}(l_i) \geq \texttt{length}(l_{i+1})$.
2. the last element of [DIFFL]u equals the last element of [TO-CHECK]u.
3. for each pair of lists l_i, l_{i+1} such that l_i and l_{i+1} are elements of [TO-CHECK]u, the i-th element of [DIFFL]u is the list of elements by which l_i and l_{i+1} differ.

PROOF

(by induction on the length of [TO-CHECK]u)

Base:

(a) For each u such that $\mathrm{S}(u) \sqsubseteq$ *check*, if $m = 0$ and thus [TO-CHECK]$u = <>$ then $\mathrm{S}(u) =$ *check0*. Hence S([DIFFL]u) = *elist*. Therefore all three claims of the proposition hold trivially.

(b) For each u such that $\mathrm{S}(u) \sqsubseteq$ *check*, if $m = 1$ and thus [TO-CHECK]$u = < l_1 >$, then $\mathrm{S}(u) =$ *check1* because V([TO-CHECK REST]u) = *elist*. Furthermore [DIFFL FIRST]u = [TO-CHECK FIRST]$u = l_1$. Since [TO-CHECK REST]u = [REC TO-CHECK]u and S([REC]u) $\sqsubseteq$ *check*, it follows that S([REC]u) = *check0* and therefore S([REC DIFFL]u) = *elist* by (a), which entails [DIFFL REST]u = [REC DIFFL]$u = <>$ by definition. Now we know that [DIFFL]$u = < l_1 >$. Hence claim (2) of the proposition is met. Because there is only one element on [TO-CHECK]u, the claims (1) and (3) hold trivially.

Hypothesis:

Suppose that we have shown for each u such that $\mathrm{S}(u) \sqsubseteq$ *check* that if [TO-CHECK]$u = < l_1, \ldots, l_n >$ is a finite list of lists of synsems of length n, then the proposition holds for u.

Step:

For each u such that $\mathrm{S}(u) \sqsubseteq$ *check*, if [TO-CHECK]$u = < l_1, \ldots, l_{n+1} >$ is of length $(n + 1)$ with $n \geq 1$, then $\mathrm{S}(u) =$ *check2*, [TO-CHECK]$u = < l_1, \ldots, l_{n+1} >$. Since [TO-CHECK FIRST]$u = l_1$ = [TEST BIG]u, [TO-CHECK REST FIRST]$u = l_2$ = [TEST SMALL]u and S([TEST]u) $\sqsubseteq$ *subset*, we know by the properties of *subset*, that $\texttt{length}(l_1) \geq \texttt{length}(l_2)$ and di_1 = [TEST DIFF]u is the list of elements by which l_1 and l_2 differ.

By definition of *check2* we get [TEST DIFF]u = [DIFFL FIRST]$u = di_1$ and [TO-CHECK REST]$u = < l_2, \ldots, l_{n+1} > =$ of length n. Since S([REC]u)

$= \mathrm{S}(u') \sqsubseteq$ *check* and [REC TO-CHECK]u = [TO-CHECK REST]u the hypothesis applies to [TO-CHECK]u' = [REC TO-CHECK]u. Because we have already shown that $\texttt{length}(l_1) \geq \texttt{length}(l_2)$ it follows immediately that for each pair of lists l_i, l_{i+1} such that l_i and l_{i+1} are elements of [TO-CHECK]u, it is the case that $\texttt{length}(l_i) \geq \texttt{length}(l_{i+1})$. Thus claim (1) of the proposition is met.

Furthermore, since [DIFFL REST]u = [REC DIFFL]u it follows with the hypothesis that the last element of [DIFFL]u equals the last element of [TO-CHECK]u, which is claim (2) of the proposition. The fact that [DIFFL FIRST]$u = di_1$ is the list of elements by which l_1 and l_2 differ as shown above , [DIFFL REST]u = [REC DIFFL]u and the hypothesis entail the last claim of the proposition.
Hence the proposition holds for u.

The proposition follows by induction. ■

With *check* we have a sort that specifies a list of lists of decreasing length where the elements of each list that succeeds another list are contained in the preceding lists. But this list of lists which reflects the argument raising properties of the verbs in the verbal complex is only a means to determine the native complements of each verb, i.e., the elements by which immediately succeeding lists differ. We can say that the lists on [DIFFL]u are ordered according to the level of embedding of the lists on [TO-CHECK]u with the highest level coming first.

In the theory of *embed* the sorts *perm*, *compsverb* and *check* interact. The theory of *embed* itself is not recursive but it employs recursive sorts. In (4.28) the interaction of the values of the attributes of *order-embed* is defined.

(4.28) *Theory of order-embed*

:∼*order-embed* →
(:PERMVERBS PERMUTED ≈ :COMPSVERB VERBL
∧ :COMPSVERB COMPSL ≈ :CHECK TO-CHECK)

as AVM:

$\begin{bmatrix} \textit{order-embed} \end{bmatrix} \Longrightarrow$

$$\begin{bmatrix} \textit{order-embed} & \\ \textsc{permverbs} & \begin{bmatrix} \textit{perm} & \\ \textsc{origin} & \textit{list-of-signs} \\ \textsc{permuted} & \boxed{1}\ \textit{list-of-signs} \end{bmatrix} \\ & \\ \textsc{check} & \begin{bmatrix} \textit{check} & \\ \textsc{to-check} & \boxed{2}\ \textit{list-of-lists-of-synsems} \\ \textsc{diffl} & \textit{list-of-lists-of-synsems} \end{bmatrix} \\ & \\ \textsc{compsverb} & \begin{bmatrix} \textit{compsverb} & \\ \textsc{verbl} & \boxed{1}\ \textit{list-of-signs} \\ \textsc{compsl} & \boxed{2}\ \textit{list-of-lists-of-synsems} \end{bmatrix} \end{bmatrix}$$

There is a clear division of labor between the three sorts *perm*, *compsverb* and *check*. *perm* makes sure that each permutation of the input list PERMVERBS ORIGIN, which is the list of verbs, is available. This makes it possible to find the order of verbs of the verbal complex we are looking for. *compsverbs* specifies the list of complements of the verbs in a parallel order to the list of verbs. *check* chooses the correct order of the list of complements, thereby also determining the order of the verbs and defining the list of elements by which each pair of immediately succeeding COMPS lists differs. In effect, we receive a list of lists of native complements of the verbs parallel to the list of verbs ordered according to the level of embedding with the highest level coming first.

For the following proposition and proof we define the notion of list of complements of a sign which was used informally before.

Def 3 *List of complements*

c **is the list of complements of** v iff

$c = [\textsc{synsem loc cat val comps}]v.$

We use the definition of being a list of complements of a sign in the proposition and proof for the sort *order-embed*.

PROPOSITION
If u is an *order-embed* and $[\textsc{permverbs origin}]u = < u_1, \ldots, u_n > =$ *verblist* is a finite list of signs, then $[\textsc{compsverb verbl}]u$ $= < u'_1, \ldots, u'_n > =$ *verblist'* is a permutation of *verblist* and for each i with $1 \leq i \leq n-1$ the i-th element of $[\textsc{check diffl}]u$ contains the list by which the list of complements of u'_i and u'_{i+1} differ, $\texttt{length}([\textsc{synsem loc cat val comps}]u'_i) \geq \texttt{length}([\textsc{synsem loc cat val comps}]u'_{i+1})$ and the n-th element of $[\textsc{check diffl}]u$ is the list of complements of u'_n.

PROOF

For each u such that $S(u)$ = *order-embed*, if [PERMVERBS ORIGIN]u = $< u_1, \ldots, u_n > = verblist$, then [COMPSVERB VERBL]$u$ = $< u'_1, \ldots, u'_n >$ is a permutation of *verblist*, since [PERMVERBS PERMUTED]u is a permutation of *verblist* by the theory of *perm* (because S([PERMVERBS]u) $\sqsubseteq$ *perm*) and [PERMVERBS PERMUTED]u = [COMPSVERB VERBL]u by definition.

Because S([COMPSVERB]u) $\sqsubseteq$ *compsverb* the theory of *compsverb* entails that for each i with $1 \leq i \leq n-1$ the i-th element of [COMPSVERB COMPSL]u is the list of complements of u'_i. With [COMPSVERB COMPSL]u = [CHECK TO-CHECK]u (by definition) and the theory of *check* (because S([CHECK]u) $\sqsubseteq$ *check*) it follows immediately that for each i with $1 \leq i \leq n-1$ the i-th element of [CHECK DIFFL]u contains the list by which the list of complements of u'_i and u'_{i+1} differ, `length`([SYNSEM LOC CAT VAL COMPS]u'_i) $\geq$ `length`([SYNSEM LOC CAT VAL COMPS]u'_{i+1}) and the n-th element of [CHECK DIFFL]u is the list of complements of u'_n.

Hence the proposition holds.

■

With the last proposition the theory for the sort *order-embed* has been completed. Before we can define the theory of *ws2* we have only one more theory to discuss, the theory of *embedding*.

4.3.3 The sort *embedding*

We can now turn to the last attribute of *ws2* objects, which is called EMBEDDING and takes an object of sort *embedding* as its value. It is another complex part of the theory of *ws2* and fulfills two tasks. Firstly, it looks up the signs that correspond to the synsem objects the verbs specify on their COMPS list. More precisely, the signs corresponding to the native complements of the verbs specified by the theory of *order-embed* are looked up on the BASE list (which is the ALL-DTRS list). Picking out the corresponding signs is necessary in order to get access to the phonological information of the phrases in question that has to be ordered by the LP principles but is not present in synsem objects. Once the signs are identified they are used to construct an object of sort *embed* which encodes the information of the level of embedding of the arguments of the verbs in a simple way. The following AVM depicts the important parts of the sorts that we will discuss in detail in the following theories:

(4.29) *The structure of embedding*

$$
\begin{bmatrix}
\textit{embedding} & \\
\textsc{base} & \boxed{1}\ \textit{list-of-signs} \\
\textsc{verbl} & <\ \boxed{3}\ \textit{sign} \mid \textit{list-of-signs}\ > \\
\textsc{args} & \begin{bmatrix} \textit{list-of-lists-of-synsems} \\ \textsc{first}\ \boxed{2}\ \textit{list-of-synsems} \\ \textsc{rest}\ \ \textit{list-of-lists-of-synsems} \end{bmatrix} \\
\textsc{embed} & \begin{bmatrix} \textit{embed} & \\ \textsc{verb} & \boxed{3}\ \textit{sign} \\ \textsc{args} & \boxed{4}\ \textit{list-of-signs} \\ \textsc{deeper} & \textit{embed} \end{bmatrix} \\
\textsc{make-los} & \begin{bmatrix} \textit{lsignsynsems} & \\ \textsc{base} & \boxed{1}\ \textit{list-of-signs} \\ \textsc{lsign} & \boxed{4}\ \textit{nelist-of-signs} \\ \textsc{lsynsem} & \boxed{2}\ \textit{nelist-of-synsems} \\ \textsc{test} & \begin{bmatrix} \textit{is-in-ss} & \\ \textsc{set} & \boxed{1}\ \textit{nelist-of-signs} \\ \textsc{elesynsem} & \textit{synsem} \\ \textsc{elesign} & \textit{sign} \\ \textsc{rec} & \textit{is-in-ss} \end{bmatrix} \\ \textsc{rec} & \textit{lsignsynsems} \end{bmatrix} \\
\textsc{rec} & \textit{embedding}
\end{bmatrix}
$$

The BASE list establishes the usual link to the ALL-DTRS list of the phrase. It is structure-shared with the BASE list of *ws2* objects by the theory of *ws2* and contains the list of signs of the permutation domain. Also by virtue of the theory of *ws2*, VERBL consists of all verbs in the permutation domain which were identified and ordered in the theories of *find-verbs* and *order-embed*. ARGS contains the native complements of the verbs in the order of embedding as specified by *order-embed*. Since *embedding* will be used in interaction with *order-embed*, we can presuppose that the VERBL list and the ARGS list are of the same length.

Objects of sort *lsignsynsems*, which are the value of the attribute MAKE-LOS (make list of signs), turn the list of lists of synsems of the native complements into the corresponding list of lists of signs. Because this has to be done recursively for each element of the lists in ARGS, another sort is introduced. Its name is *is-in-ss*, and its task is to match each synsem object to its sign on the BASE list. In other words, what we get is a stack of recursive sorts:

In *embedding* there is a list of lists of synsems called ARGS that has to be converted. In *lsignsynsems* of its MAKE-LOS attribute there is a list of synsems in LSYNSEM that has to be converted into a list of signs. MAKE-LOS LSYNSEM contains the first element of the ARGS list. Further lists in ARGS are treated in the recursion of *embedding*.

Because the LSYNSEM list of *lsignsynsems* can not be converted in one step either, there is another recursive sort called *is-in-ss* which takes the first element of the LSYNSEM list and finally converts it into the sign on SET (which is the BASE LIST) whose SYNSEM value it is. Further synsem objects of the LSYNSEM list are treated in the recursion of *lsignsynsems*.

The result of the conversion is stored in the *embed* value of the attribute EMBED. EMBED VERB contains the structurally highest verb of coherent constructions and EMBED ARGS a list of signs that represents its native arguments, i.e., those arguments whose SYNSEM values are not inherited from a governed verb. In EMBED DEEPER VERB the governed verb follows with its arguments in EMBED DEEPER ARGS. Since these structures are finite, the recursion ends at some level of embedding with the value *embed0* for the attribute DEEPER. The correct relation between arguments and verbs is a consequence of the theory of *order-embed*.
In (4.30) we present the structure of the new sort *embedding*.

(4.30) *Appropriateness Conditions for the sort embedding*

embedding	BASE	*list-of-signs*
	VERBL	*list-of-signs*
	ARGS	*list-of-lists-of-synsems*
	EMBED	*embed*

embedding0	BASE	*list-of-signs*
	VERBL	*elist*
	ARGS	*elist*
	EMBED	*embed0*
embedding1	BASE	*list-of-signs*
	VERBL	*nelist-of-signs*
	ARGS	*list-of-lists-of-synsems*
	EMBED	*embed1*
	MAKE-LOS	*lsignsynsems*
	REC	*embedding*

Before the theory of *embedding* can be defined, we need to define the theory of *lsignsynsems* which is supposed to deliver the conversion of the list of lists of native complements (of sort *synsem*) in ARGS into the lists of native arguments of sort *sign* which are distributed in the recursively structured *embed* objects according to their level of embedding and the verbs they belong to.

lsignsynsems uses the values of its attributes BASE, LSIGN and LSYNSEM for the conversion. LSYNSEM contains a list of synsem objects. On LSIGN there is a list of signs that are also elements of BASE and whose SYNSEM values are identical with with the synsem objects that are on the same position on LSYNSEM. This means the SYNSEM value of the n-th sign on LSIGN equals the n-th object on LSYNSEM.

(4.31) *Appropriateness Conditions for the sort lsignsynsems*

lsignsynsems	BASE	*list-of-signs*		
	LSIGN	*list-of-signs*		
	LSYNSEM	*list-of-synsems*		
	lsignsynsems0	BASE	*list-of-signs*	
		LSIGN	*elist*	
		LSYNSEM	*elist*	
	lsignsynsems1	BASE	*list-of-signs*	
		LSIGN	*nelist-of-signs*	
		LSYNSEM	*nelist-of-synsems*	
		TEST	*is-in-ss*	
		REC	*lsignsynsems*	

lsignsynsems0 is the trivial case where both LSIGN and LSYNSEM are empty. In the case of *lsignsynsems1* some real work has to be done. Since the signs on LSIGN can not be looked up on BASE in one step and thus the SYNSEM values can not be specified on LSYNSEM in one step, the attribute TEST introduces a new sort *is-in-ss* which breaks the task down into a stepwise treatment of signs and synsem objects.

Thus the appropriateness conditions of *is-in-ss* introduce three attributes (apart from the attribute for recursion) that accommodate the sign, the synsem object and the BASE list, which is called SET list.

(4.32) *Appropriateness Conditions for the sort is-in-ss*

is-in-ss	SET	*nelist-of-signs*	
	ELESYNSEM	*synsem*	
	ELESIGN	*sign*	
	is-in-ss0	SET	*nelist-of-signs*
		ELESYNSEM	*synsem*
		ELESIGN	*sign*
	is-in-ss1	SET	*nelist-of-signs*
		ELESYNSEM	*synsem*
		ELESIGN	*sign*
		REC	*is-in-ss*

As usual, there is a base case (*is-in-ss0*) and a case that uses recursion (*is-in-ss*). In (4.33) we formalize the theory for *is-in-ss*.

(4.33) *Theory of is-in-ss*

$$:\sim \textit{is-in-ss0} \quad \to \quad \begin{array}{l} (\text{:SET FIRST SYNSEM} \approx \text{:ELESYNSEM} \\ \wedge\ \text{:SET FIRST} \approx \text{:ELESIGN}) \end{array}$$

as AVM:

$$\begin{bmatrix} \textit{is-in-ss0} \end{bmatrix} \Longrightarrow$$

$$\begin{bmatrix} \textit{is-in-ss0} & \\ \text{SET} & \begin{bmatrix} \textit{nelist-of-signs} & \\ \text{FIRST}\ \boxed{2} & \begin{bmatrix} \textit{sign} \\ \text{SYNSEM}\ \boxed{1}\ \textit{synsem} \end{bmatrix} \\ \text{REST}\ \textit{list-of-signs} & \end{bmatrix} \\ \text{ELESYNSEM} & \boxed{1}\ \textit{synsem} \\ \text{ELESIGN} & \boxed{2}\ \textit{sign} \end{bmatrix}$$

$$:\sim \textit{is-in-ss1} \quad \to \quad \begin{array}{l} (\neg(\text{:SET FIRST SYNSEM} \approx \text{:ELESYNSEM}) \\ \wedge\ \text{:REC SET} \approx \text{:SET REST} \\ \wedge\ \text{:REC ELESYNSEM} \approx \text{:ELESYNSEM} \\ \wedge\ \text{:REC ELESIGN} \approx \text{:ELESIGN}) \end{array}$$

as AVM:

$$\begin{bmatrix} \textit{is-in-ss1} \end{bmatrix} \Longrightarrow$$

$$\begin{bmatrix} \textit{is-in-ss1} & \\ \text{SET} & \begin{bmatrix} \textit{nelist-of-signs} & \\ \text{FIRST} & \begin{bmatrix} \textit{sign} \\ \text{SYNSEM}\ \boxed{1}\ \textit{synsem} \end{bmatrix} \\ \text{REST}\ \boxed{3}\ \textit{list-of-signs} & \end{bmatrix} \\ \text{ELESYNSEM} & \boxed{2}\ \textit{synsem} \\ \text{ELESIGN} & \boxed{4}\ \textit{sign} \\ \text{REC} & \begin{bmatrix} \textit{is-in-ss} & \\ \text{SET} & \boxed{3}\ \textit{nelist-of-signs} \\ \text{ELESYNSEM} & \boxed{2}\ \textit{synsem} \\ \text{ELESIGN} & \boxed{4}\ \textit{sign} \end{bmatrix} \end{bmatrix} \ \&\ \boxed{1} \neq \boxed{2}$$

In *is-in-ss0* the first sign on SET equals the sign which is the value of ELESIGN, and the SYNSEM value of that sign equals the synsem object which is the value of ELESYNSEM. This means that the correspondence we are looking for has been found.

In *is-in-ss1* the SYNSEM value of the first sign of SET is different from the value of ELESYNSEM. Therefore the corresponding sign to the synsem object has not been found yet. It has to be in the rest of the SET list. The rest of the SET list and the values of ELESIGN and ELESYNSEM are passed on to the recursion.

We show that the value of ELESIGN must be an element of SET and ELESYNSEM has to be its SYNSEM value. We prove these two claims by contradiction, i.e., we start from the assumption that they are wrong and show that it leads to a contradiction.

PROPOSITION
If u is an *is-in-ss* and [SET]u is a finite list of signs, then [ELESYNSEM]u = [ELESIGN SYNSEM]u and [ELESIGN]u is an element of [SET]u.

PROOF
(by contradiction)

For each u such that S(u) $\sqsubseteq$ *is-in-ss*, if [SET]u $= < u_1, \ldots, u_n >$ is finite then

Part 1: [ELESIGN]u is an element of [SET]u

> Suppose [ELESIGN]$u \neq u_i$ for each i with $1 \leq i \leq n$.
>
> Then S(u) $\neq$ *is-in-ss0*, because otherwise [SET FIRST]u = [ELESIGN]u contrary to the assumption. Therefore S(u) = *is-in-ss1*, entailing S([SET REST]u) = *nelist-of-signs* by definition of *is-in-ss*, since [REC SET]u = [SET REST]u and S([REC]u) $\sqsubseteq$ *is-in-ss*.
>
> Furthermore we know that [REC ELESIGN]u = [ELESIGN]u. But now we are in the same position as in the beginning, more specifically, we know that S([REC]u) = *is-in-ss1*. Thus [SET]u has to be infinite, contrary to the precondition. Thus our assumption was wrong and [ELESIGN]u has to be an element of [SET]u.

Part 2: [ELESYNSEM]u = [ELESIGN SYNSEM]u

> Suppose [ELESYNSEM]$u \neq$ [ELESIGN SYNSEM]u.
>
> Then S(u) $\neq$ *is-in-ss0*, because otherwise [SET FIRST]u = [ELESIGN]u and [SET FIRST SYNSEM]u = [ELESYNSEM]u, entailing [ELESYNSEM]u = [ELESIGN SYNSEM]u contrary to the assumption.

Therefore S(u) = *is-in-ss1*, entailing S([SET REST]u) = *nelist-of-signs* by definition of *is-in-ss*, since [REC SET]u = [SET REST]u and S([REC]u) $\sqsubseteq$ *is-in-ss*.

Furthermore we know that [REC ELESIGN]u = [ELESIGN]u and [REC ELESYNSEM]u = [ELESYNSEM]u. But now we are in the same position as in the beginning, more specifically, we know that S([REC]u) = *is-in-ss1*. Thus [SET]u has to be infinite, contrary to the precondition. Thus our assumption was wrong and it has to be the case that [ELESYNSEM]u = [ELESIGN SYNSEM]u.

■

is-in-ss provides a conversion of a single synsem object into its corresponding sign on a specified list of signs. With the help of *is-in-ss* the theory of *lsignsynsems* can be defined which converts lists of synsem objects into lists of signs which are on the BASE list.

(4.34) *Theory of lsignsynsems*

$$
\begin{array}{lll}
{:}\!\sim\! \textit{lsignsynsems1} & \rightarrow & (\text{:BASE} \approx \text{:TEST SET} \\
 & & \wedge\ \text{:LSYNSEM FIRST} \approx \text{:TEST ELESYNSEM} \\
 & & \wedge\ \text{:LSIGN FIRST} \approx \text{:TEST ELESIGN} \\
 & & \wedge\ \text{:REC BASE} \approx \text{:BASE} \\
 & & \wedge\ \text{:REC LSIGN} \approx \text{:LSIGN REST} \\
 & & \wedge\ \text{:REC LSYNSEM} \approx \text{:LSYNSEM REST})
\end{array}
$$

as AVM:

$\left[\textit{lsignsynsems1} \quad\quad \right] \Longrightarrow$

$$
\left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{lsignsynsems1}} \\
\textsc{base} & \boxed{1}\ \textit{list-of-signs} \\
\textsc{lsign} & \left[\begin{array}{ll} \multicolumn{2}{l}{\textit{nelist-of-signs}} \\ \textsc{first} & \boxed{3}\ \textit{sign} \\ \textsc{rest} & \boxed{4}\ \textit{list-of-signs} \end{array}\right] \\
\textsc{lsynsem} & \left[\begin{array}{ll} \multicolumn{2}{l}{\textit{nelist-of-synsems}} \\ \textsc{first} & \boxed{2}\ \textit{synsem} \\ \textsc{rest} & \boxed{5}\ \textit{list-of-synsems} \end{array}\right] \\
\textsc{test} & \left[\begin{array}{ll} \multicolumn{2}{l}{\textit{is-in-ss}} \\ \textsc{set} & \boxed{1}\ \textit{nelist-of-signs} \\ \textsc{elesynsem} & \boxed{2}\ \textit{synsem} \\ \textsc{elesign} & \boxed{3}\ \textit{sign} \end{array}\right] \\
\textsc{rec} & \left[\begin{array}{ll} \multicolumn{2}{l}{\textit{lsignsynsems}} \\ \textsc{base} & \boxed{1}\ \textit{list-of-signs} \\ \textsc{lsign} & \boxed{4}\ \textit{list-of-signs} \\ \textsc{lsynsem} & \boxed{5}\ \textit{list-of-synsems} \end{array}\right]
\end{array}\right]
$$

lsignsynsems0 needs no description but simply describes the case where both the LSIGN list and the LSYNSEM list are empty. For *lsignsynsems1* objects both lists are nonempty. The first sign of LSIGN is identified with the ELESIGN value of the *is-in-ss* object which is the value of the TEST attribute. In the same way, the first synsem object of LSYNSEM equals the ELESYNSEM value of the *is-in-ss* object. Because the BASE list also equals the SET list of *is-in-ss*, the theory of *is-in-ss* guarantees the intended conversion, and we can go into recursion with the entire BASE list and the rest of LSIGN and LSYNSEM.

PROPOSITION
If u is an *lsignsynsems*, $[\textsc{base}]u$ is finite and $[\textsc{lsign}]u = < u_1, \ldots, u_m >$ is finite, then $[\textsc{lsynsem}]u = < [\textsc{synsem}]u_1, \ldots, [\textsc{synsem}]u_m >$ and u_i is an element of $[\textsc{base}]u$ for each i with $1 \leq i \leq m$.
We say that $[\textsc{lsign}]u$ baselist-corresponds to $[\textsc{lsynsem}]u$.

PROOF
(by induction on the length of $[\textsc{lsign}]u$)

Base:
For each u such that $\mathrm{S}(u) \sqsubseteq$ *lsignsynsems*, if $m = 0$ and $[\textsc{lsign}]u = <>$, then we know that $\mathrm{S}(u) =$ *lsignsynsems0*. Hence $\mathrm{S}([\textsc{lsynsem}]u) =$ *elist* and the proposition trivially holds for u.

Hypothesis:
Suppose we have shown for each u with $\mathrm{S}(u) \sqsubseteq$ *lsignsynsems* that if $m = n$ and $[\textsc{lsign}]u = < u_1, \ldots, u_n >$, then $[\textsc{lsynsem}]u = < [\textsc{synsem}]u_1, \ldots, [\textsc{synsem}]u_n >$ and u_i is an element of $[\textsc{base}]u$ for each i with $1 \leq i \leq n$.

Step:
For each u such that $\mathrm{S}(u) \sqsubseteq$ *lsignsynsems*, if $m = n + 1$ and $[\textsc{lsign}]u = < u_1, \ldots, u_{n+1} >$ then $\mathrm{S}(u) =$ *lsignsynsems1*. By definition of *lsignsynsems* it follows that $u_1 = [\textsc{lsign first}]u = [\textsc{test elesign}]u$, $[\textsc{lsynsem first}]u = [\textsc{test elesynsem}]u$ and $[\textsc{base}]u = [\textsc{test set}]u$. Because $\mathrm{S}([\textsc{test}]u) \sqsubseteq$ *is-in-ss*, the theory of *is-in-ss* entails that $[\textsc{synsem}]u_1 = [\textsc{lsynsem first}]u$ and u_1 is an element of $[\textsc{base}]u$.

By definition we also know that $[\textsc{rec base}]u = [\textsc{base}]u$, $[\textsc{rec lsign}]u = [\textsc{lsign rest}]u = < u_2, \ldots, u_{n+1} >$ of length n and $[\textsc{rec lsynsem}]u = [\textsc{lsynsem rest}]u$. Since $\mathrm{S}([\textsc{rec}]u) = \mathrm{S}(u') \sqsubseteq$ *lsignsynsems*, the hypothesis applies to $[\textsc{lsign}]u' = [\textsc{rec lsign}]u$. It follows that $[\textsc{rec lsynsem}]u = < [\textsc{synsem}]u_2, \ldots, [\textsc{synsem}]u_{n+1} >$ and u_i is an element of $[\textsc{base}]u$ for each i with $2 \leq i \leq n{+}1$. Since $[\textsc{rec lsynsem}]u = [\textsc{lsynsem rest}]u$ (by definition) $[\textsc{synsem}]u_1 = [\textsc{lsynsem first}]u$ (as shown above) and we know that u_1 is an element of $[\textsc{base}]u$ (as shown above) the proposition holds for u.

The proposition follows by induction.

■

With the proof of the theory of *lsignsynsems* the component of the theory of *embedding* which finds the signs belonging to the synsem objects that come from the COMPS lists of verbs has been finished. Before the theory of *embedding* itself can be presented, the appropriateness conditions for the objects in which the output of *embedding* is encoded have to be stated. No special theory for these objects is needed.

(4.35) *Appropriateness Conditions for the sort embed*

embed			
	embed0		
	embed1	VERB	*sign*
		ARGS	*list-of-signs*
		DEEPER	*embed*

With all components of *embedding* being defined, the connections can be established by simple term equations. In (4.36) we see the theory for the sort *embedding*.

(4.36) *Theory of embedding*

$$
\begin{array}{lll}
{:}{\sim}\mathit{embedding1} & \rightarrow & (\text{:MAKE-LOS BASE} \approx \text{:BASE} \\
 & & \wedge\ \text{:MAKE-LOS LSYNSEM} \approx \text{:ARGS FIRST} \\
 & & \wedge\ \text{:EMBED VERB} \approx \text{:VERBL FIRST} \\
 & & \wedge\ \text{:EMBED ARGS} \approx \text{:MAKE-LOS LSIGN} \\
 & & \wedge\ \text{:EMBED DEEPER} \approx \text{:REC EMBED} \\
 & & \wedge\ \text{:REC BASE} \approx \text{:BASE} \\
 & & \wedge\ \text{:REC VERBL} \approx \text{:VERBL REST} \\
 & & \wedge\ \text{:REC ARGS} \approx \text{:ARGS REST})
\end{array}
$$

as AVM:

$$
\begin{bmatrix}\mathit{embedding1}\end{bmatrix} \Longrightarrow
$$

$$
\begin{bmatrix}
\mathit{embedding1} & \\
\text{BASE} & \boxed{1}\ \mathit{list\text{-}of\text{-}signs} \\
\text{VERBL} & \begin{bmatrix} \mathit{nelist\text{-}of\text{-}signs} & \\ \text{FIRST} & \boxed{3}\ \mathit{sign} \\ \text{REST} & \boxed{6}\ \mathit{list\text{-}of\text{-}signs} \end{bmatrix} \\
\text{ARGS} & \begin{bmatrix} \mathit{nelist\text{-}of\text{-}lists\text{-}of\text{-}synsems} & \\ \text{FIRST} & \boxed{2}\ \mathit{list\text{-}of\text{-}synsems} \\ \text{REST} & \boxed{7}\ \mathit{list\text{-}of\text{-}lists\text{-}of\text{-}synsems} \end{bmatrix} \\
\text{EMBED} & \begin{bmatrix} \mathit{embed1} & \\ \text{VERB} & \boxed{3}\ \mathit{sign} \\ \text{ARGS} & \boxed{4}\ \mathit{list\text{-}of\text{-}signs} \\ \text{DEEPER} & \boxed{5}\ \mathit{embed} \end{bmatrix} \\
\text{MAKE-LOS} & \begin{bmatrix} \mathit{lsignsynsems} & \\ \text{BASE} & \boxed{1}\ \mathit{list\text{-}of\text{-}signs} \\ \text{LSIGN} & \boxed{4}\ \mathit{list\text{-}of\text{-}signs} \\ \text{LSYNSEM} & \boxed{2}\ \mathit{list\text{-}of\text{-}synsems} \end{bmatrix} \\
\text{REC} & \begin{bmatrix} \mathit{embedding} & \\ \text{BASE} & \boxed{1}\ \mathit{list\text{-}of\text{-}signs} \\ \text{VERBL} & \boxed{6}\ \mathit{list\text{-}of\text{-}signs} \\ \text{ARGS} & \boxed{7}\ \mathit{list\text{-}of\text{-}lists\text{-}of\text{-}synsems} \\ \text{EMBED} & \boxed{5}\ \mathit{embed} \end{bmatrix}
\end{bmatrix}
$$

The theory of *ws2* will decree that VERBL contains the list of lexical verbs that are on the BASE list, while ARGS contains the native complements of the verbs in corresponding order. The latter point will be ensured by the theory of *order-embed*. For our purposes we can therefore simplify the proof of *embedding* by presupposing that VERBL and ARGS are of equal length.

The theory of *embedding1* puts the first sign on VERBL into EMBED VERB. The native arguments belonging to that verb are delivered by the theory of *lsignsynsems*: The first list of synsems of ARGS equals the LSYNSEM list of the *lsignsynsems* object which is the value of the MAKE-LOS attribute. Thus MAKE-LOS LSIGN returns the corresponding list of signs, which is put into EMBED ARGS. The construction of the output object is continued in the value of EMBED DEEPER, where the verb and the arguments that come later are stored in a recursive structure. The rest of VERBL and ARGS is treated in the recursion of *embedding*.

In order to delimit the size of the *embed* object the value of the DEEPER attribute that belongs to the *embed* object which has the last sign on VERBL as its VERB value is *embed0*. This is a direct consequence of the appropriateness conditions of *embedding* and entails that no arbitrary signs that are not specified on VERBL may follow deeper in the *embed* structure.

PROPOSITION
If u is an *embedding*, [BASE]u is finite, [ARGS]$u = < l_1, \ldots, l_m >$ is finite and [VERBL]$u = < v_1, \ldots, v_m >$ then

1. for every i with $1 \leq i \leq m$, [EMBED ... DEEPER ... ARGS]u baselist-corresponds to the element l_i of [ARGS]$u = < l_1, \ldots, l_m >$, where DEEPER occurs $i-1$ times in the term.

2. for every i with $1 \leq i \leq m$, [EMBED ... DEEPER ... VERB]$u = v_i$, where DEEPER occurs $i-1$ times in the term.

3. S([EMBED ... DEEPER ...]u) = *embed0*, where DEEPER occurs m times in the term.

PROOF
(by induction on the length of [ARGS]u)

Base:
For each u such that S(u) $\sqsubseteq$ *embedding*, if [BASE]u is finite and [ARGS]u = <> then S(u) = *embedding0* by definition. Therefore S([VERBL]u) = *elist*, which means [VERBL]u = <>, and S([EMBED]u) = *embed0*. Hence the proposition holds for u.

Hypothesis:
Suppose that we have shown for each u with S(u) $\sqsubseteq$ *embedding*, if [BASE]u

is finite, [ARGS]u = $< l_1, \ldots, l_n >$ is of length n and [VERBL]u = $< v_1, \ldots, v_n >$, then the proposition holds for u.

Step:
For each u such that $\mathrm{S}(u) \sqsubseteq$ *embedding*, if [BASE]u is finite and [ARGS]u = $< l_1, \ldots, l_{n+1} >$, then it follows from the appropriateness conditions of EMBEDDING that $\mathrm{S}(u) =$ *embedding1*. Then we know that the following equations hold: [MAKE-LOS BASE]u = [BASE]u, [MAKE-LOS LSYNSEM]u = [ARGS FIRST]$u = l_1$ and [MAKE-LOS LSIGN]u = [EMBED ARGS]u. Because S([MAKE-LOS]u) $\sqsubseteq$ *lsignsynsems*, the theory of *lsignsynsems* entails that [EMBED ARGS]u baselist-corresponds to l_1.

By definition we also know that [EMBED VERB]u = [VERBL FIRST]u = v_1.

Next we know that [REC BASE]u = [BASE]u, [REC VERBL]u = [VERBL REST]u = $< v_2, \ldots, v_{n+1} >$ of length n, [REC ARGS]u = [ARGS REST]u = $< l_2, ..., l_{n+1} >$ of length n. Since S([REC]u) $\mathrm{S}(u') \sqsubseteq$ *embedding*, the hypothesis applies to [ARGS]u' = [REC ARGS]u. It follows immediately that

1. for every i with $1 \leq i \leq n$, [EMBED ... DEEPER ... ARGS]u' baselist-corresponds to the element l_{i+1} of [ARGS]u = $< l_2, \ldots, l_{n+1} >$, where DEEPER occurs $i-1$ times in the term.

2. for every i with $1 \leq i \leq n$, [EMBED ... DEEPER ... VERB]u' = v_{i+1}, where DEEPER occurs $i-1$ times in the term.

3. S([EMBED ... DEEPER ...]u') = *embed0*, where DEEPER occurs n times in the term.

With [EMBED DEEPER]u = [REC EMBED]u = [EMBED]u' and the results above about [EMBED ARGS]u and [EMBED VERB]u the proposition follows for u.

The proposition follows by induction. ■

4.3.4 Final remarks on the sort *ws2*

We have now set the stage for the presentation of the theory of *ws2*, which defines the links between the values of the attributes of its three components that were subject of the previous three sections. The theory of *ws2* itself is not recursive, but it makes use of the recursive theories of *find-verbs*, *order-embed* and *embedding*. After our extensive discussion of these sorts above, their combination into an overall theory is a trivial step.

(4.37) *Theory of ws2*

$$
\begin{array}{ll}
:\sim \mathit{ws2} \quad \rightarrow & \\
& (\textsc{:find-verbs verbl} \approx \textsc{:order permverbs origin} \\
& \wedge\ \textsc{:order compsverb verbl} \approx \textsc{:embedding verbl} \\
& \wedge\ \textsc{:order check diffl} \approx \textsc{:embedding args} \\
& \wedge\ \textsc{:base} \approx \textsc{:find-verbs base} \\
& \wedge\ \textsc{:base} \approx \textsc{:embedding base})
\end{array}
$$

as AVM:

$$
\begin{bmatrix} \mathit{ws2} \end{bmatrix} \Longrightarrow
$$

$$
\begin{bmatrix}
\mathit{ws2} & \\
\textsc{find-verbs} & \begin{bmatrix} \mathit{find\text{-}verbs} & \\ \textsc{base} & \boxed{6}\ \mathit{list\text{-}of\text{-}signs} \\ \textsc{verbl} & \boxed{1}\ \mathit{list\text{-}of\text{-}signs} \end{bmatrix} \\
\textsc{order} & \begin{bmatrix} \mathit{order\text{-}embed} & \\
\textsc{permverbs} & \begin{bmatrix} \mathit{perm} & \\ \textsc{origin} & \boxed{1}\ \mathit{list\text{-}of\text{-}signs} \\ \textsc{permuted} & \boxed{4}\ \mathit{list\text{-}of\text{-}signs} \end{bmatrix} \\
\textsc{check} & \begin{bmatrix} \mathit{check} & \\ \textsc{to-check} & \boxed{5}\ \mathit{list\text{-}of\text{-}lists\text{-}of\text{-}synsems} \\ \textsc{diffl} & \boxed{3}\ \mathit{list\text{-}of\text{-}lists\text{-}of\text{-}synsems} \end{bmatrix} \\
\textsc{compsverb} & \begin{bmatrix} \mathit{compsverb} & \\ \textsc{verbl} & \boxed{4}\ \mathit{list\text{-}of\text{-}signs} \\ \textsc{compsl} & \boxed{5}\ \mathit{list\text{-}of\text{-}lists\text{-}of\text{-}synsems} \end{bmatrix}
\end{bmatrix} \\
\textsc{embedding} & \begin{bmatrix} \mathit{embedding} & \\ \textsc{base} & \boxed{6}\ \mathit{list\text{-}of\text{-}signs} \\ \textsc{verbl} & \boxed{4}\ \mathit{list\text{-}of\text{-}signs} \\ \textsc{args} & \boxed{3}\ \mathit{list\text{-}of\text{-}lists\text{-}of\text{-}synsems} \\ \textsc{embed} & \mathit{embed} \end{bmatrix} \\
\textsc{base} & \boxed{6}\ \mathit{list\text{-}of\text{-}signs}
\end{bmatrix}
$$

find-verbs identifies the lexical verbs on the BASE list. In our fragment they can only be verbs that belong to the verbal complex or it is the finite main verb of the sentence if there is no verbal complex. The verbs are on FIND-VERBS VERBL, whose value equals the value of ORDER PERMVERBS. Now it is a consequence of the structure of *order-embed* that ORDER CHECK DIFFL contains a list of lists of native complements

of the verbs that are contained in ORDER COMPSVERB VERBL. Both lists are ordered in such a way that the verb and the list of its native complements which are on the highest level of embedding in the sentence come first on their respective lists and the ones that are deepest embedded come last. Speaking of the verbs only, this means that the finite verb always comes first and the main verb comes last. Finally, the value of ORDER COMPSVERB VERBL equals the value of EMBEDDING VERBL and the value of ORDER CHECK DIFFL is identical with the value of EMBEDDING ARGS. Thus *embedding* determines the signs on the BASE list which belong to the synsem objects that are the native complements of the verbs, and the verbs and their native arguments are stored in the recursive *embed* object whose specification is the purpose of the whole theory of *ws2*.
The following proposition captures the essential properties of *ws2* objects.

PROPOSITION
If u is a *ws2* and $[\text{BASE}]u = < u_1, \ldots, u_n >$ is finite then [ORDER COMPSVERB VERBL]$u = < v_1, \ldots, v_k > = verblist$ is a permutation of a list consisting exactly of those objects which are an element of $[\text{BASE}]u$, a verb and of sort *word* such that

1. for every i with $1 \leq i \leq k$, [EMBEDDING EMBED ... DEEPER ... VERB]$u = v_i$, where DEEPER occurs $i - 1$ times in the term.

2. (a) for every i with $1 \leq i \leq k-1$, [EMBEDDING EMBED ... DEEPER ... ARGS]u (with DEEPER occurring $i-1$ times in the term) baselist-corresponds to the list of those elements of the list of complements of [EMBEDDING EMBED ... DEEPER ... VERB]$u = verb$ (with DEEPER occurring $i-1$ times in the term) by which the lists of complements of *verb* and [EMBEDDING EMBED ... DEEPER ... VERB]u (with DEEPER occurring i times in the term) differ,
(b) [EMBEDDING EMBED ... DEEPER ... ARGS]u (with DEEPER occurring $k - 1$ times in the term) baselist-corresponds to the list of complements of [EMBEDDING EMBED ... DEEPER ... VERB]u (with DEEPER occurring $k - 1$ times in the term),
(c) for every i with $1 \leq i \leq k - 1$, the length of the list of complements of [EMBEDDING EMBED ... DEEPER ... VERB]u (with DEEPER occurring $i - 1$ times in the term) is greater than or equal to the length of the list of complements of [EMBEDDING EMBED ... DEEPER ... VERB]u (with DEEPER occurring i times in the term).

3. S([EMBED ... DEEPER ...]u) = *embed0*, where DEEPER occurs k times in the term.

PROOF

For each u such that $S(u) = ws2$, if [BASE]$u = < u_1, \ldots, u_n >$ is finite, then it follows with S([FIND-VERBS]u) $\sqsubseteq$ *find-verbs* and the theory of *find-verbs*, S([ORDER]u) $\sqsubseteq$ *order-embed* and the theory of *order-embed*, [FIND-VERBS VERBL]u = [ORDER PERMVERBS ORIGIN]u and [BASE]u = [FIND-VERBS BASE]u (by definition) that [ORDER COMPSVERB VERBL]u is a permutation of a list consisting exactly of those objects which are an element of [BASE]u, a verb and of sort *word*.
Then it follows with S([EMBEDDING]u) $\sqsubseteq$ *embedding* and the theory of *order-embed* and [ORDER COMPSVERB VERBL]u = [EMBEDDING VERBL]u (by definition) that for every i with $1 \leq i \leq k$, [EMBEDDING EMBED ... DEEPER ... VERB]$u = v_i$, where DEEPER occurs $i - 1$ times in the term (claim (1)). From this we conclude with the theory of *order-embed* that S([EMBED ... DEEPER ...]u) = *embed0*, where DEEPER occurs k times in the term (claim (3)).
Furthermore, now it follows from the theory of *order-embed*, the theory of *embedding* and the equations [ORDER CHECK DIFFL]u = [EMBEDDING ARGS]u and [BASE]u = [EMBEDDING BASE]u (by definition) that
(a) for every i with $1 \leq i \leq k - 1$, [EMBEDDING EMBED ... DEEPER ... ARGS]u (with DEEPER occurring $i - 1$ times in the term) baselist-corresponds to the list of those elements of the list of complements of [EMBEDDING EMBED ... DEEPER ... VERB]u = *verb* (with DEEPER occurring $i - 1$ times in the term) by which the list of complements of *verb* and [EMBEDDING EMBED ... DEEPER ... VERB]u (with DEEPER occurring i times in the term) differ (claim (2), part (a)),
(b) [EMBEDDING EMBED ... DEEPER ... ARGS]u (with DEEPER occurring $k - 1$ times in the term) baselist-corresponds to the list of complements of [EMBEDDING EMBED ... DEEPER ... VERB]u (with DEEPER occurring $k - 1$ times in the term) (claim (2), part (b)), and
(c) for every i with $1 \leq i \leq k - 1$, the length of the list of complements of [EMBEDDING EMBED ... DEEPER ... VERB]u (with DEEPER occurring $i - 1$ times in the term) is greater than or equal to the length of the list of complements of [EMBEDDING EMBED ... DEEPER ... VERB]u (with DEEPER occurring i times in the term), which is claim (3), part (c).

Hence the proposition holds.

■

With the theory and proof of *ws2* it has been shown how an object that encodes the relationship between verbs and their native arguments in a sentence can be defined on the basis of the syntactic assumptions of section 3.1.

Now that we have painted a detailed picture of the technical aspects of a formalization of the idea of an auxiliary structure *embed*, it is time to step back and consider the results. To see the general situation it is useful to recall our overall architecture for the formalization of LP principles as it is described in section 4.1. (4.38) presents a phrase in which the currently specified parts of the LP module are indicated.

(4.38) *The structure of a phrase*

$$
\begin{bmatrix}
\textit{phrase} & \\
\text{PHON} & \boxed{2}\ \textit{list-of-phons} \\
\text{BUILD-PHON} & \begin{bmatrix} \textit{build-phon} & \\ \text{PERM ORIGIN} & \boxed{1}\ \textit{list-of-signs} \\ \text{SIGNPHON PHONS} & \boxed{2}\ \textit{list-of-phons} \end{bmatrix} \\
\text{LP} & \begin{bmatrix} \textit{lp-hc} & \\ \text{PHON} & \boxed{2}\ \textit{list-of-phons} \\ \text{WS2} & \begin{bmatrix} \textit{ws2} & \\ \text{FIND-VERBS} & \textit{find-verbs} \\ \text{ORDER} & \textit{order-embed} \\ \text{EMBEDDING} & \begin{bmatrix} \textit{embedding} & \\ \text{EMBED} & \boxed{3}\ \textit{embed} \end{bmatrix} \\ \text{BASE} & \boxed{1}\ \textit{list-of-signs} \end{bmatrix} \\ \text{EMBED} & \boxed{3}\ \textit{embed} \end{bmatrix} \\
\text{DTRS} & \begin{bmatrix} \textit{head-comp-struc} & \\ \text{HEAD-DTR} & \textit{sign} \\ \text{COMP-DTRS} & \textit{list-of-signs} \\ \text{ALL-DTRS} & \boxed{1}\ \textit{list-of-signs} \end{bmatrix}
\end{bmatrix}
$$

Although the connection between the *ws2* object and other parts of *lp-hc* and *phrase* have not yet been formally established, it is clearly visible what will happen in the theories of *lp-hc* and *phrase*. The structure-sharing defined by the latter theories is indicated by the tags of (4.38).

The value of BUILD-PHON constrains the phonology of the phrase to permutations of the signs which are on the ALL-DTRS list. This is why the ALL-DTRS list is sometimes called the permutation domain of a phrase. The value of LP, *lp-hc*, additionally constrains the permutations which are specified by *build-phon* to those permutations which comply with the LP principles. *ws2* is a part of *lp-hc* which is sensitive to the specific syntactic analysis the LP module is combined with. This is one of the characteristics of objects such as *ws2* that specify the interface between the LP principles and other parts of the linguistic structure. The sensitivity to the underlying syntactic analysis also marks an important difference between objects like *ws2* and the LP principles, which are independent of the syntactic analysis. The linguistically meaningful part of the theory of *ws2* consists in the claim that the information encoded in *embed* objects is in fact necessary to write adequate LP principles for German. As we have seen, *embed* objects reflect the hierarchical structure of the verbal complex and match the verbs with their native arguments. Native arguments are arguments whose SYNSEM value is not inherited by raising.

We have shown that all of this information is available even if the underlying syntactic structure of sentences is not hierarchical and the COMPS list of verbs contains synsem objects instead of signs. This does not mean, though, that these distinctions are entirely irrelevant for the LP module. For example, a more articulated syntactic structure of the verbal complex could contribute to a simplification of the theory of *ws2*, because the level of embedding of the verbs could directly be derived from the syntactic tree. Signs on the COMPS list would make the whole theory of *lsignsynsems* unnecessary.

4.4 A first LP principle (*lpp2*)

In the previous sections we have presented some prerequisites for a precise formalization of LP rules in HPSG. As a result it is now possible to give a formal explication for the type of statements linguists usually want to make about linearization regularities. Sections 4.4 and 4.5 contain two rules one might want to establish for constituent order phenomena in the German mittelfeld. As in the previous sections our choice for the particular principles we formalize was mainly motivated by expository reasons. We will discuss two rules for pronouns that relate to the value of EMBED to demonstrate the use of the information about the level of embedding in coherent constructions that was captured by the theory of *ws2* in section 4.3. It will be shown that one can start out with a rather intuitive notion of the rule that is to be added to the HPSG principles of constituent order and proceed step by step to arrive at formal theories that can be proved to deliver what is required by the linguist.

The structure of the present section is as follows: in 4.4.1 an informal statement of the LP principle is given together with an outline of how it is going to be formalized. In sections 4.4.2 and 4.4.3 the subparts of the LP principle are presented and discussed in detail. 4.4.4 consists of the theory of the sort *lpp2* and some concluding remarks.

4.4.1 Presenting an *lpp2* object

Let us first consider the following informal statement about the order of pronouns in the mittelfeld:

(4.39) *A first LP principle*

if a *phrase* x has signs i and j such that

i and j are on the same level of embedding and
i is a subject pronoun and
j is a non-subject pronoun

then i's PHON precedes j's PHON on x's PHON

The LP-rule in (4.39) deals with the relative order of a subject pronoun and non-subject pronouns of the same level of embedding, where the notion 'level of embedding' refers to the concept of embedding we have formalized in the previous section, 4.3. For simple cases with a single verb (4.39) says that a subject pronoun has to precede every other pronoun that is an argument of the verb. For more complicated cases, i.e. cases where there appear pronouns at different levels of embedding, the LP principle in (4.39) looks at every level separately. We have chosen to discuss the principle in (4.39) here, because it involves the need to identify a subject pronoun. The information about subjecthood is, however, not encoded into the sign that functions as subject, as there is no attribute IS-SUBJECT specified on signs whose value would indicate the subject function. But, there is a SUBJ attribute defined on verbs which makes it possible to tell which sign is the subject of which verb. Thus, information on subjecthood has to be gained from the context within which a sign appears, it cannot be directly seen from the concerning sign itself. It is mainly for this purpose that the VERB attribute is defined for *embed* objects

As stated in section 4.1, we intend to formalize the linearization regularity expressed in (4.39) as an HPSG principle. This principle will be put into the LPP2 value of the *lp* object sketched in subsection 4.2. (4.40) gives an overview of the attributes appropriate for the sort *lpp2*.

(4.40) *Appropriateness Conditions for the sort lpp2*

lpp2	PHON	*list-of-phons*
	EMBED	*embed*

lpp2_0	PHON	*list-of-phons*
	EMBED	*embed0*
lpp2_1	PHON	*list-of-phons*
	EMBED	*embed1*
	FIND-S/NS	*find-s/ns*
	TEST	*prec*
	REC	*lpp2*

Before we take a closer look at the attributes of *lpp2_1* and the new sorts, let us draw the general picture of what is going to happen in the theory of *lpp2*. The structure of the sort *lpp2* is parallel with the one given for an arbitrary recursive LP principle in (4.9) of section 4.1.2. Thus, technically speaking, *lpp2* encodes a recursive LP principle.

It is already evident from the informal statement in (4.39) that there are two things to be done. First, the pronouns for each level of embedding have to be identified and second, for all pronouns of a given level of embedding the constraint on their phonological order has to be applied. Given the setup of the logic of HPSG the most convenient way to accomplish this task is to recursively search for pronouns at each level of embedding, put them into lists and then apply the constraints on ordering. With the appropriateness conditions of (4.40) these tasks are distributed over the values of the attributes FIND-S/NS, TEST and REC.

This short characterization of the inner properties of the sort *lpp2* indicates its relation to the general architecture of an LP principle described in 4.1.2. As has been noticed, we are dealing with an LP principle which involves recursive precedence checking as it was exemplified in (4.9). In the present case, checking proceeds along the structure of embedding, i.e. the EMBED value. It can be seen from the appropriateness conditions in (4.40), how closely the *lpp2* object follows the construction of the involved *embed* object: As long as there is yet a further level of embedding (i.e. the EMBED value of *lpp2* is of sort *embed1*), there is a further step of recursion, indicated by the REC specification on *lpp2*. At the very last level of embedding an *embed0* is the value of *lpp2*'s EMBED attribute and there is no REC attribute specified. In (4.41) an example *lpp2* object is given that shows how the EMBED structure is used to monitor the recursive structure of an *lpp2* object. A precise formulation will be stated in the theory of *lpp2* in section 4.4.4.

(4.41) *The recursive structure of an lpp2 object*

$$
\left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{lpp2\_1}} \\
\text{PHON} & \boxed{1}\ \textit{list-of-phons} \\
\text{EMBED} & \left[\begin{array}{ll}
\text{VERB} & \textit{sign} \\
\text{ARGS} & \textit{list-of-signs} \\
\text{DEEPER}\ \boxed{2} & \left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{embed1}} \\
\multicolumn{2}{l}{\ldots} \\
\text{DEEPER}\ \boxed{3} & \left[\begin{array}{l}
\textit{embed1} \\
\ldots \\
\text{DEEPER}\ \boxed{4}\ \textit{embed0}
\end{array}\right]
\end{array}\right]
\end{array}\right] \\
\ldots & \\
\text{REC} & \left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{lpp2\_1}} \\
\text{PHON} & \boxed{1}\ \textit{list-of-phons} \\
\text{EMBED} & \boxed{2}\ \textit{embed1} \\
\multicolumn{2}{l}{\ldots} \\
\text{REC} & \left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{lpp2\_1}} \\
\text{PHON} & \boxed{1}\ \textit{list-of-phons} \\
\text{EMBED} & \boxed{3}\ \textit{embed1} \\
\multicolumn{2}{l}{\ldots} \\
\text{REC} & \left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{lpp2\_0}} \\
\text{PHON} & \boxed{1}\ \textit{list-of-phons} \\
\text{EMBED} & \boxed{4}\ \textit{embed0}
\end{array}\right]
\end{array}\right]
\end{array}\right]
\end{array}\right]
$$

The AVM in (4.41) makes clear that the EMBED DEEPER value at a given level of recursion is identical with the EMBED value one level down, i.e. with the REC EMBED value. Thus, the EMBED DEEPER value of the *lpp2_1* object that appears as overall object in the example, is identical to the *embed* value of the *lpp2* object that is the REC value of the 'matrix' *lpp2* object. Furthermore, there is a covariation between the species of an *lpp2* object and its EMBED value. If the EMBED value is of sort *embed0*, the *lpp2* object is of species *lpp2_1*. If the EMBED value is of sort *embed1*, the *lpp2* object is of species *lpp2_1*. Thus, the recursion within an *lpp2* object is closely tied to the recursive structure of its EMBED value.

In (4.41) the function of the attribute REC has been illustrated which is similar to the *rec* attribute discussed at the example sort *lpp-recursive* in (4.9). The attributes FIND-S/NS and TEST, too, stand in a close relation the the example attributes used in (4.7).They are responsible for the precedence checking. The attribute FIND-S/NS fulfills the function of FIND-A/B of (4.7) and (4.9), i.e. it singles out in lists the phonologies of the subject pronouns and the non-subject pronouns at a given level of embedding. The TEST attribute corresponds to the PREC attribute in (4.7). There it is checked that the phonology of the subject pronoun precedes the phonology of the non-subject pronouns on the PHON list of *lpp2*.

The following sections contain a discussion of the sorts introduced in (4.40). In 4.4.2 the value of the attribute FIND-S/NS is described, and in 4.4.3 the sort *prec* is defined. In 4.4.4 the theory of *lpp2* can finally be stated.

4.4.2 The sort *find-s/ns*

The first new sort used in *lpp2* is *find-s/ns*. Its task is to identify the phonologies of the subject and non-subject pronouns of a given level of embedding. In order to fulfill this task, an object of the sort *find-s/ns* is specified for some attributes. These are given in (4.42).

(4.42) *Appropriateness Conditions for the sort find-s/ns*

find-s/ns	VERB	*sign*
	ARGS	*list-of-signs*
	SU	*signphon*
	NSU	*signphon*
	FIND-PRON	*findpron*
	SEP-PRON	*separate-pron*

The idea of *find-s/ns* is quite simple. VERB and ARGS are ensured to take the verb and the arguments list of the current level of embedding. as their values. In FIND-PRON all pronominal arguments of that level of embedding are collected. In SEP-PRONS the subject pronoun is singled out from the others. We have to use a list-valued feature even for the subject pronoun since in the general case there is the possibility that the subject is not pronominal or there might even be no subject at all. The theory of *separate-pron* thus results in two (possibly empty) lists of signs that are supposed to contain pronominal signs with the specified properties. Those two lists are passed on to the SU SIGNS list and the

NSU SIGNS list. The latter two lists exist by virtue of the sort *signphon* which was introduced in (2.8), and which occurs in the appropriateness conditions of the sort *find-s/ns* as value of SU and NSU. The sort *signphon* converts lists of signs into lists of phons (and vice versa) and is used in this particular case to produce a list with the phonology of the subject (if there is a pronominal one) and a list with the phonology of those pronouns that are not subject. This is exactly the information that is needed as input for the sort *prec* in the theory of *lpp2* in (4.72).

The attribute FIND-PRON takes values of the sort *find-s/ns* which has not yet been defined. Like the sort *find-verbs* in (4.19) it is an example of how to find elements with a particular property on a list and to collect them in an other list. But *find-pron* is much simpler than the sort discussed in (4.19). We define *find-pron* as following:

(4.43) *Appropriateness Conditions for the sort find-pron*

find-pron	ARGS	*list-of-signs*		
	PRONS	*list-of-signs*		
	find-pron0	ARGS	*elist*	
		PRONS	*elist*	
	find-pron1	ARGS	*nelist-of-signs*	
		PRONS	*list-of-signs*	
		REC	*find-pron*	

As can be seen in (4.43), there are only two lists occurring in a *find-pron* object. The PRONS list contains exactly all the pronouns which occur on the ARGS list. This idea can now directly be defined, since *find-pron* does not contain further new sorts. In (4.44) the theory of this sort is given.

(4.44) *Theory of find-pron*

$$
\begin{array}{ll}
(:\sim \textit{find-pron1} & \\
\wedge \text{ :ARGS FIRST SYNSEM LOC CONTENT} \sim \textit{pron}) & \\
\rightarrow & (\text{:PRONS FIRST} \approx \text{:ARGS FIRST} \\
& \wedge \text{ :PRONS REST} \approx \text{:REC PRONS} \\
& \wedge \text{ :REC ARGS} \approx \text{:ARGS REST})
\end{array}
$$

as AVM:

$$\begin{bmatrix} \textit{find-pron1} \\ \text{ARGS FIRST SYNSEM LOC CONTENT } \textit{pron} \end{bmatrix} \Longrightarrow$$

$$\begin{bmatrix} \textit{find-pron1} & \\ \text{ARGS} & \begin{bmatrix} \textit{nelist-of-signs} \\ \text{FIRST } \boxed{2}\ \textit{sign} \\ \text{REST } \boxed{1}\ \textit{list-of-signs} \end{bmatrix} \\ \text{PRONS} & < \boxed{2}\ \textit{sign} \mid \boxed{3}\ \textit{list-of-signs} > \\ \text{REC} & \begin{bmatrix} \textit{find-pron} \\ \text{ARGS } \boxed{1}\ \textit{list-of-signs} \\ \text{PRONS } \boxed{3}\ \textit{list-of-signs} \end{bmatrix} \end{bmatrix}$$

$$\begin{array}{ll} (:\sim \textit{find-pron1} & \\ \wedge \neg(:\text{ARGS FIRST SYNSEM LOC CONTENT} \sim \textit{pron})) & \\ \rightarrow & (:\text{PRONS} \approx :\text{REC PRONS} \\ & \wedge :\text{REC ARGS} \approx :\text{ARGS REST}) \end{array}$$

As AVM:

$$\begin{bmatrix} \textit{find-pron1} \\ \text{ARGS FIRST SYNSEM LOC CONTENT} \neg\ \textit{pron} \end{bmatrix} \Longrightarrow$$

$$\begin{bmatrix} \textit{find-pron1} & \\ \text{ARGS} & \begin{bmatrix} \textit{nelist-of-signs} \\ \text{FIRST } \textit{sign} \\ \text{REST } \boxed{1}\ \textit{list-of-signs} \end{bmatrix} \\ \text{PRONS} & \boxed{3}\ \textit{list-of-signs} \\ \text{REC} & \begin{bmatrix} \textit{find-pron} \\ \text{ARGS } \boxed{1}\ \textit{list-of-signs} \\ \text{PRONS } \boxed{3}\ \textit{list-of-signs} \end{bmatrix} \end{bmatrix}$$

Before we are going to prove a crucial property of the objects of the sort *find-pron*, we introduce a definition which will help to formulate the proposition more naturally. Def 4 defines the notion of "pronoun" within the HPSG structure.

Def 4 *Pronouns*

p **is a pronoun** iff

$S([\text{SYNSEM LOC CONTENT}]p) \sqsubseteq \textit{pron}$

Definition Def 4 together with definition Def 2 from section 4.3 enable us to state a proposition to the theory of *find-prons* in a straightforward way. To recall, in Def 2 the notion 'element of a list' has been defined. The intuitive content of the following proposition is that all pronouns of the ARGS list are collected in the PRONS list.

PROPOSITION[43]
For every u with $\mathrm{S}(u) \sqsubseteq$ *find-pron*, it is the case that $[\textsc{prons}]u$ consists exactly of those objects o which are pronouns and elements of $[\textsc{args}]u$.

PROOF
(by induction on the length of $[\textsc{args}]u$)

Base:

For each u with $\mathrm{S}(u) \sqsubseteq$ *find-pron*, if $\mathrm{S}([\textsc{args}]u) =$ *elist*,

then $\mathrm{S}(u) =$ *find-pron0* and therefore $\mathrm{S}([\textsc{prons}]u) =$ *elist*. As both lists are empty, it trivially holds that $[\textsc{prons}]u$ consists exactly of the objects which are pronouns and elements of $[\textsc{args}]u$.

Hypothesis:

Assume that we have shown for every u with $\mathrm{S}(u) \sqsubseteq$ *find-pron* that if $[\textsc{args}]u = l$ of length n and $[\textsc{prons}]u = h$, then h consists exactly of those objects which are pronouns and elements of l.

Step:

For each u with $\mathrm{S}(u) \sqsubseteq$ *find-pron*, if $[\textsc{args}]u = l'$ is of length $n+1$,

then $\mathrm{S}(u) =$ *find-pron1*. Let $[\textsc{args first}]u = o$ and $[\textsc{prons}]u = h'$. Two cases have to be distinguished:

Case 1: o is a pronoun

Then (by Def 4) $\mathrm{S}([\textsc{synsem loc content}]o) \sqsubseteq$ *pron*. We know that $[\textsc{prons first}]u = o$, thus o is an element of h'. Since $[\textsc{rec args}]u = [\textsc{args rest}]u = l$ and l is of length n, our hypothesis applies and guarantees that $[\textsc{rec prons}]u$ consists exactly of those objects which are elements of $[\textsc{rec args}]u$ and pronouns. As l and l' only differ in o, and as o is a pronoun and an element of l', the proposition holds for u.

Case 2: o is not a pronoun

Then (by Def 4) it is not the case that $\mathrm{S}([\textsc{synsem loc content}]o) \sqsubseteq$ *pron*. Hence we know that $[\textsc{prons}]u = [\textsc{rec prons}]u$. Since $[\textsc{rec args}]u = [\textsc{args rest}]u = l$ and l is of length n, our hypothesis applies and guarantees that $[\textsc{rec prons}]u$ consists exactly of those objects which are elements of $[\textsc{rec args}]u$ and pronouns. Therefore o is not an element of $[\textsc{rec prons}]u$ and thus not an element of $[\textsc{prons}]u$ either. As o is not a pronoun and as nothing is changed from $[\textsc{rec prons}]u$ to $[\textsc{prons}]u$, the proposition holds for u.

[43]Throughout the rest of chapter 4 all lists we are dealing with are assumed to be finite, even if this is not explicitly stated in the definitions, propositions or proofs.

By induction the proposition follows.

■

The sort *find-pron* is illustrated with an example in (4.45). In this example just the phonology of the signs on the ARGS list is given. The signs with the phonologies *er* (*he*) and *ihr* (*to-her*) are pronouns whereas the sign with the PHON value *das Buch* (*the book*) is not. The pronouns among the elements of the ARGS list are collected in the *prons* list. It can be seen from the AVM in (4.45) that it is checked at every level of recursion whether the ARGS FIRST element is a pronoun, i.e. is to be added to the PRONS list or not.

(4.45) *An example find-pron object*

$$\left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{find-pron1}} \\
\textsc{args} & \left\langle \boxed{1}\left[\textsc{phon}\ <\textit{er}>\right], \boxed{2}\left[\textsc{phon}\ <\textit{ihr}>\right], \boxed{3}\left[\textsc{phon}\ <\textit{das, Buch}>\right]\right\rangle \\
\textsc{prons} & \left\langle \boxed{1}, \boxed{2} \right\rangle \\
\textsc{rec} & \left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{find-prons1}} \\
\textsc{args} & \left\langle \boxed{2}, \boxed{3} \right\rangle \\
\textsc{prons} & \left\langle \boxed{2} \right\rangle \\
\textsc{rec} & \left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{find-prons1}} \\
\textsc{args} & \left\langle \boxed{3}\left[\textsc{phon}\ \textit{a book}\right]\right\rangle \\
\textsc{prons} & \textit{elist} \\
\textsc{rec} & \left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{find-verbs0}} \\
\textsc{args} & \textit{elist} \\
\textsc{prons} & \textit{elist}
\end{array}\right]
\end{array}\right]
\end{array}\right]
\end{array}\right]$$

After having illustrated how the sort *find-pron* works, we can look at the way it is used in the overall architecture of the LP principle. There, its function is to supply a list of all pronouns that occur as arguments of a verb at some level of embedding. The sort *find-pron* appears as value of the value of the attribute FIND-PRON on a *find-s/ns* object. Within the theory of *find-s/ns* there will a link between the ARGS value and the FIND-PRON ARGS value. Furthermore the list of pronouns identified within the attribute FIND-PRON it given to another attribute of the sort *find-s/ns*, namely SEP-PRON. The relations between the FIND-PRON attribute and the other attributes of a *find-s/ns* object are demonstrated in the AVM in (4.46) using the concrete example from (4.45).

(4.46) *The interactions involving the* FIND-PRON *attribute*

$$\begin{bmatrix} \textit{find-s/ns} & \\ \text{VERB} & \textit{sign} \\ \text{ARGS} & \boxed{4} \left\langle \boxed{1} \left[\text{PHON} <\textit{er}>\right], \boxed{2} \left[\text{PHON} <\textit{ihr}>\right], \boxed{3} \left[\text{PHON} <\textit{das, Buch}>\right] \right\rangle \\ \text{SU} & \textit{signphon} \\ \text{NSU} & \textit{signphon} \\ \text{FIND-PRON} & \begin{bmatrix} \textit{find-pron} & \\ \text{ARGS} & \boxed{4} \left\langle \boxed{1}, \boxed{2}, \boxed{3} \right\rangle \\ \text{PRONS} & \boxed{5} \left\langle \boxed{1}, \boxed{2} \right\rangle \end{bmatrix} \\ \text{SEP-PRON} & \begin{bmatrix} \textit{separate-pron} & \\ \text{PRONS} & \boxed{5} \\ \ldots & \end{bmatrix} \end{bmatrix}$$

The next new sort to be discussed, *separate-pron*, is concerned with dividing the list of pronouns into two lists. The first one, SU, will consist of the subject pronoun (if there is one) and the second one, NSU, contains the pronouns that are not subjects (if there are any). The attributes appropriate for this sort are specified in (4.47).

(4.47) *Appropriateness Conditions for the sort separate-pron*

separate-pron	PRONS	*list-of-signs*
	VERB	*sign*
	SU	*list-of-signs*
	NSU	*list-of-signs*

separate-pron0	PRONS	*elist*
	VERB	*sign*
	SU	*elist*
	NSU	*elist*
separate-pron1	PRONS	*nelist-of-signs*
	VERB	*sign*
	SU	*nelist-of-signs*
	NSU	*list-of-signs*
	REC	*separate-pron*
separate-pron2	PRONS	*nelist-of-signs*
	VERB	*sign*
	SU	*list-of-signs*
	NSU	*nelist-of-signs*
	REC	*separate-pron*

Just like the sort *find-pron*, *separate-pron* 'looks for' elements of a given list with a certain property and collects them on an other list. The case of *separate-pron* is, however, a little more complicated in various respects: First, a subject pronoun can only be identified as such if one looks at the verb whose subject it is. Therefore an attribute VERB is included in the sort *separate-pron*. Second, as it is not only the subject that is needed for the precedence checking but the non-subjects as well, two lists are built up: the SU list for the subject and the NSU list for the other elements of the ARGS list. Since there are no new sorts within (4.47), the theory of *separate-pron* can be formalized and proved:

(4.48) *Theory of separate-pron*

$$
\begin{array}{ll}
\multicolumn{2}{l}{:\sim \textit{separate-pron1}} \\
\rightarrow & (\textsc{:verb synsem loc cat val subj first} \approx \textsc{:prons first synsem} \\
& \wedge\ \textsc{:su first} \approx \textsc{:prons first} \\
& \wedge\ \textsc{:su rest} \sim \textit{elist} \\
& \wedge\ \textsc{:nsu} \approx \textsc{:rec nsu} \\
& \wedge\ \textsc{:prons rest} \approx \textsc{:rec prons})
\end{array}
$$

as AVM:

$$
\begin{bmatrix} \textit{separate-pron1} \end{bmatrix} \Longrightarrow
$$

$$
\begin{bmatrix}
\textit{separate-pron1} & \\
\textsc{prons} & \begin{bmatrix} \textit{nelist-of-signs} & \\ \textsc{first}\ \boxed{2} & \begin{bmatrix} \textit{sign} \\ \textsc{synsem}\ \boxed{1}\ \textit{synsem} \end{bmatrix} \\ \textsc{rest} & \boxed{4}\ \textit{list-of-signs} \end{bmatrix} \\
\textsc{verb} & \textsc{synsem loc cat val subj first}\ \boxed{1}\ \textit{synsem} \\
\textsc{su} & <\boxed{2}\ \textit{sign}> \\
\textsc{nsu} & \boxed{3}\ \textit{list-of-signs} \\
\textsc{rec} & \begin{bmatrix} \textit{separate-pron} & \\ \textsc{prons} & \boxed{4}\ \textit{list-of-signs} \\ \textsc{nsu} & \boxed{3}\ \textit{list-of-signs} \end{bmatrix}
\end{bmatrix}
$$

$$
\begin{array}{ll}
\multicolumn{2}{l}{:\sim \textit{separate-pron2}} \\
\rightarrow & ((\neg(\textsc{:verb synsem loc cat val subj first} \approx \textsc{:prons first synsem}) \\
& \quad \vee\ \textsc{:verb synsem loc cat val subj} \sim \textit{elist}) \\
& \wedge\ \textsc{:su} \approx \textsc{:rec su} \\
& \wedge\ \textsc{:nsu first} \approx \textsc{:prons first} \\
& \wedge\ \textsc{:nsu rest} \approx \textsc{rec nsu} \\
& \wedge\ \textsc{:prons rest} \approx \textsc{:rec prons})
\end{array}
$$

as AVM (second disjunct only):

$$
\begin{bmatrix} \textit{separate-pron2} \end{bmatrix} \Longrightarrow
$$

$$
\begin{bmatrix}
\textit{separate-pron2} & \\
\textsc{prons} & \langle \boxed{2}\ \textit{sign} \mid \boxed{4}\ \textit{list-of-signs} \rangle \\
\textsc{verb} & \textsc{synsem loc cat val subj}\ \textit{elist} \\
\textsc{su} & \boxed{1}\ \textit{list-of-signs} \\
\textsc{nsu} & \langle \boxed{2}\ \textit{sign} \mid \boxed{3}\ \textit{list-of-signs} \rangle \\
\textsc{rec} & \begin{bmatrix} \textsc{prons} & \boxed{4}\ \textit{list-of-signs} \\ \textsc{su} & \boxed{1}\ \textit{list-of-signs} \\ \textsc{nsu} & \boxed{3}\ \textit{list-of-signs} \end{bmatrix}
\end{bmatrix}
$$

Before we come to the proposition and its proof, we have to introduce another definition. (5) defines the notion 'subject' in terms of HPSG attributes. According to Def 5 a subject of a verb v is the *synsem* object that is the first element of the SUBJ list of v.

Def 5 *Subject*

For any objects s and v:

s **is the subject of** v iff

1. S([SYNSEM LOC CAT HEAD]v) = *verb* and
2. the path SYNSEM LOC CAT VAL SUBJ FIRST is defined on v
3. [SYNSEM LOC CAT VAL SUBJ FIRST]$v = s$

This definition is used in the following proposition which states that within the sort *separate-pron* the PRONS list divided into the SU and the NSU lists in the appropriate way. The proposition is followed directly by its proof.

PROPOSITION
For each u with $S(u) \sqsubseteq$ *separate-pron* it holds that
for each v which is an element of [PRONS]u:

1. [SU]u consists exactly of those objects v which are elements of [PRONS]u and [SYNSEM]v is the subject of [VERB]u.

2. [NSU]u consists exactly of those objects v which are elements of [PRONS]u and not the subject of [VERB]u.

PROOF
(by induction on the length of [PRONS]u, parallel for both parts of the proposition)

Base:

For each u with $\mathrm{S}(u) \sqsubseteq$ *separate-pron*, if $\mathrm{S}([\text{PRONS}]u) =$ *elist*,

then $\mathrm{S}(u) =$ *separate-pron0*, therefore $\mathrm{S}([\text{SU}]u) =$ *elist* and $\mathrm{S}([\text{NSU}]u) =$ *elist*. As all three lists are empty the proposition trivially holds.

Hypothesis:

Assume that we have shown for each u with $\mathrm{S}(u) \sqsubseteq$ *separate-pron* that if $[\text{PRONS}]u = l$ of length n $[\text{SU}]u = h$ and $[\text{NSU}]u = k$, then:

(i) h consists exactly of those objects v which are elements of l and [SYNSEM]v is the subject of [VERB]u.

(ii) k consists exactly of those objects v which are elements of l and not the subject of [VERB]u.

Step:

For each u with $\mathrm{S}(u) \sqsubseteq$ *separate-pron*, if $[\text{PRONS}]u = l$' of length $n + 1$,

then we have to distinguish two cases.

Case 1: $\mathrm{S}(u) =$ *separate-pron1*

Let $[\text{PRONS FIRST}]u = o$, $[\text{SU}]u = h'$ and $[\text{NSU}]u = k'$. Then we know that $[\text{SYNSEM}]o = [\text{VERB SYNSEM LOC CAT VAL SUBJ FIRST}]u$, thus (by Def 5) o is the subject of [VERB]u. With $[\text{SU FIRST}]u = [\text{PRONS FIRST}]u$ we know that o is an element of h'. Since $[\text{REC PRONS}]u = [\text{PRONS REST}]u$ $= l$ of length n, our hypothesis applies to $[\text{REC SU}]u = h$ and $[\text{REC NSU}]u$ $= k$. As o is not on l it is by hypothesis neither on h nor on k.

(a) Since h consists exactly of those elements of l which are the subject of [REC VERB]u and because $[\text{REC VERB}]u = [\text{VERB}]u$, all elements of h are the subject of [VERB]u as well. Since l' differs from l only in o, h'consists exactly of those objects which are elements of l' and the subject of [VERB]u. Thus part (i) of consequent of the proposition holds for the present case.

(b) Since $k' = [\text{NSU}]u = [\text{REC NSU}]u = k$ we know that o is not an element of k'. As k consists only of those objects which are elements of l and not the subject of $[\text{REC VERB}]u = [\text{VERB}]u$ and as $[\text{NSU}]u = k$ and o is the subject of [VERB]u part (ii) of consequent of the proposition holds for u in the present case.

Case 2: S(u) = *separate-pron2*

Let [PRONS FIRST]$u = o$, [SU]$u = h'$ and [NSU]$u = k'$. Then we know that either it is not the case that [SYNSEM]o = [VERB SYNSEM LOC CAT VAL SUBJ FIRST]u, or S([VERB SYNSEM LOC CAT VAL SUBJ]u) = *elist*, hence (by Def 5) o is not the subject of [VERB]u. Then we know that [FIRST]k' = [PRONS FIRST]$u = o$ and thus that o is an element of k'. Since [REC PRONS]u = [PRONS REST]$u = l$ of length n, our hypothesis applies to [REC SU]$u = h$ and [REC NSU]$u = k$. As o is not on l it is by hypothesis neither on h nor on k.

(a) Since h consists exactly of those elements of l which are the subject of [REC VERB]u and as [REC VERB]u = [VERB]u, all elements of h are the subject of [VERB]u as well. Since l' differs from l only in o, and h' = [SU]u = [REC SU]$u = h$ and [SYNSEM]o is not the subject of [VERB]u, h' consists exactly of those objects which are elements of l' and the subject of [VERB]u. Thus part (i) of consequent of the proposition holds for u in the present case.

(b) Since [REST]k' = [NSU REST]u = [REC NSU]$u = k$ and [VERB]u = [REC VERB]u, all elements of [REC PRONS]u which are not the subject of [REC VERB]u are elements of [PRONS]u. Furthermore, they are not the subject of [VERB]u either. With o being on k' part (ii) of consequent of the proposition holds for u in the present case.

Having treated all cases, the proposition holds for u in the induction step.

By induction the consequent of the entire proposition follows.

■

The newly defined sort can be illustrated in a concrete example. Let's assume a *separate-pron* object with a PRONS list that consists of the German pronouns *ich* (*I*), *ihn* (*him*) and *dir* (*to-you*) and a VERB which has got the synsem value of *ich* on its SUBJ list. The corresponding *separate pronouns* object then looks as in (4.49).

(4.49) *An example separate-pron object*

$$
\begin{bmatrix}
\textit{separate-pron1} & \\
\text{PRONS} & \left\langle \boxed{1} \begin{bmatrix} \text{PHON} & <\textit{ich}> \\ \text{SYNSEM} & \boxed{1'} \\ \ldots & \end{bmatrix}, \boxed{2} \begin{bmatrix}\text{PHON} <\textit{ihn}>\end{bmatrix}, \boxed{3} \begin{bmatrix}\text{PHON} <\textit{dir}>\end{bmatrix} \right\rangle \\
\text{VERB} & \boxed{4} \begin{bmatrix} \textit{sign} & \\ \text{PHON} & <\textit{zeige}> \\ \text{SYNSEM LOC CAT VAL SUBJ FIRST} & \boxed{1'}\ \textit{synsem} \end{bmatrix} \\
\text{SU} & < \boxed{1}\ \textit{sign} > \\
\text{NSU} & < \boxed{2}\ \textit{sign},\ \boxed{3}\ \textit{sign} > \\
\text{REC} & \begin{bmatrix}
\textit{separate-pron2} & \\
\text{PRONS} & \langle \boxed{2}\ \textit{sign}, \boxed{3}\ \textit{sign} \rangle \\
\text{VERB} & \boxed{4} \\
\text{SU} & \textit{elist} \\
\text{NSU} & < \boxed{2}, \boxed{3} > \\
\text{REC} & \begin{bmatrix}
\textit{separate-pron2} & \\
\text{PRONS} & < \boxed{3}\ \textit{sign} > \\
\text{VERB} & \boxed{4}\ \textit{sign} \\
\text{SU} & \textit{elist} \\
\text{NSU} & < \boxed{3}\ \textit{sign} > \\
\text{REC} & \begin{bmatrix}
\textit{separate-pron0} & \\
\text{PRONS} & \textit{elist} \\
\text{VERB} & \boxed{4}\ \textit{sign} \\
\text{SU} & \textit{elist} \\
\text{NSU} & \textit{elist}
\end{bmatrix}
\end{bmatrix}
\end{bmatrix}
\end{bmatrix}
$$

Having defined the sorts *find-pron* and *separate-pron*, we can now turn to the theory of the sort *find-s/ns*. This sort coordinates the interaction between its attributes to guarantee that (i) the pronouns are singled out from the ARGS list, (ii) the subject pronoun — if present — is stuck onto the SU list and (iii) the rest of the pronouns can be found on theNSU lists. This coordinating task of the sort FIND-S/NS is accomplished by its theory, given in (4.50).

(4.50) *Theory of find-s/ns*

$$
\begin{aligned}
{:}{\sim}\textit{find-s/ns} \quad \rightarrow \quad & (\text{:FIND-PRON ARGS} \approx \text{:ARGS} \\
& \wedge\ \text{:SEP-PRON PRONS} \approx \text{:FIND-PRON PRONS} \\
& \wedge\ \text{:SEP-PRON VERB} \approx \text{:VERB} \\
& \wedge\ \text{:SU SIGNS} \approx \text{:SEP-PRON SU} \\
& \wedge\ \text{:NSU SIGNS} \approx \text{:SEP-PRON NSU})
\end{aligned}
$$

as AVM:

$$
\begin{bmatrix} \textit{find-s/ns} \end{bmatrix} \Longrightarrow
\begin{bmatrix}
\textit{find-s/ns} & \\
\textsc{verb} & \boxed{3}\ \textit{sign} \\
\textsc{args} & \boxed{1}\ \textit{list-of-signs} \\
\textsc{su} & \begin{bmatrix} \textit{signphon} \\ \textsc{signs}\ \boxed{4}\ \textit{list-of-signs} \end{bmatrix} \\
\textsc{nsu} & \begin{bmatrix} \textit{signphon} \\ \textsc{signs}\ \boxed{5}\ \textit{list-of-signs} \end{bmatrix} \\
\textsc{find-pron} & \begin{bmatrix} \textit{find-pron} & \\ \textsc{args} & \boxed{1}\ \textit{list-of-signs} \\ \textsc{prons} & \boxed{2}\ \textit{list-of-signs} \end{bmatrix} \\
\textsc{sep-pron} & \begin{bmatrix} \textit{separate-pron} & \\ \textsc{prons} & \boxed{2}\ \textit{list-of-signs} \\ \textsc{verb} & \boxed{3}\ \textit{sign} \\ \textsc{su} & \boxed{4}\ \textit{list-of-signs} \\ \textsc{nsu} & \boxed{5}\ \textit{list-of-signs} \end{bmatrix}
\end{bmatrix}
$$

The relevant property of the sort *find-s/ns* is that the phonologies of subject and nonsubject pronouns of the ARGS are collected in the SU PHONS and the NSU PHONS lists respectively. We state this relation between the attribute values of a *find-s/ns* object more formally in the subsequent proposition which is followed by the proof.

PROPOSITION
For each u with $\mathrm{S}(u) = \textit{find-s/ns}$, it holds that

1. [SU PHONS]u consists exactly of the phonology of those objects o which are pronouns, elements of [ARGS]u and the subject of [VERB]u.

2. [NSU PHONS]u consists exactly of the phonology of those objects o which are pronouns, elements of [ARGS]u and not the subject of [VERB]u.

PROOF
(via the structure of the type *find-s/ns*)

For each u with $\mathrm{S}(u) = \textit{find-s/ns}$,

we know that [ARGS]u = [FIND-PRON ARGS]u. As S([FIND-PRON]u) $\sqsubseteq$ *find-pron* it follows by the properties of this sort that [FIND-PRON PRONS]u consists exactly of those objects which are elements of [ARGS]u and pronouns.

Furthermore [SEP-PRON VERB]u = [VERB]u and [SEP-PRON PRONS]u = [FIND-PRON PRONS]u. As S([SEP-PRON]u) $\sqsubseteq$ *separate-pron*, we know that [SEP-PRON SU]u consists exactly of those elements of [SEP-PRON PRONS]u which are pronouns and the subject of [VERB]u. As [SEP-PRON SU]u consists exactly of the subject pronouns of [SEP-PRON PRONS]u and thus of [FIND-PRON PRONS]u, and [FIND-PRON PRONS]u consists exactly of the pronouns of [FIND-PRON ARGS]u and thus of [ARGS]u, we know that [SEP-PRON SU]u consists exactly of the subject pronouns of [ARGS]u. We can further conclude that [SEP-PRON NSU]u consists exactly of those objects which are elements of [ARGS]u, pronouns and not the subject of [VERB]u.

From the theory of *find-s/ns* we know that [SEP-PRON SU]u = [SU SIGNS]u. Because of S([SU]u) $\sqsubseteq$ *signphon* we know from (2.11) that [SU PHONS]u is exactly a concatenation of the PHON values on the [SU SIGNS]u list. Thus part (i) of the proposition holds for u. As [SEP-PRON NSU]u = [NSU SIGNS]u and S([NSU]u) $\sqsubseteq$ *signphon*, part (ii) of the proposition follows as well.

■

The interaction of the different attributes of *find-s/ns* has become evident by the AVM given together with the theory of the sort in (4.50). It might nevertheless be instructive to look at a more concrete example. For this purpose, the example in (4.45) and (4.46) is reconsidered here.

(4.51) *A complete example for a find-s/ns object*

$$\begin{bmatrix}
\textit{find-s/ns} & \\
\text{VERB} & \boxed{1}\begin{bmatrix}\text{PHON} & \langle\textit{zeigt}\rangle \\ \text{SYNSEM LOC CAT VAL SUBJ FIRST} & \boxed{2'}\end{bmatrix} \\
\text{ARGS} & \left\langle \boxed{2}\begin{bmatrix}\text{PHON} & \boxed{2''}\ \langle\textit{er}\rangle \\ \text{SYNSEM} & \boxed{2'}\end{bmatrix}, \boxed{3}\begin{bmatrix}\text{PHON} & \boxed{3''}\ \langle\textit{ihr}\rangle\end{bmatrix}, \boxed{4}\begin{bmatrix}\text{PHON} & \boxed{4''}\ \langle\textit{das, Buch}\rangle\end{bmatrix} \right\rangle \\
\text{SU} & \begin{bmatrix}\textit{signphon} & \\ \text{SIGNS} & \langle \boxed{2} \rangle \\ \text{PHONS} & \boxed{2''}\end{bmatrix} \\
\text{NSU} & \begin{bmatrix}\textit{signphon} & \\ \text{SIGNS} & \langle \boxed{3} \rangle \\ \text{PHONS} & \boxed{3''}\end{bmatrix} \\
\text{FIND-PRON} & \begin{bmatrix}\textit{find-pron} & \\ \text{ARGS} & \langle \boxed{2}, \boxed{3}, \boxed{4} \rangle \\ \text{PRONS} & \langle \boxed{2}, \boxed{3} \rangle\end{bmatrix} \\
\text{SEP-PRON} & \begin{bmatrix}\textit{separate-pron} & \\ \text{PRONS} & \langle \boxed{2}, \boxed{3} \rangle \\ \text{VERB} & \boxed{1} \\ \text{SU} & \langle \boxed{2} \rangle \\ \text{NSU} & \langle \boxed{3} \rangle\end{bmatrix}
\end{bmatrix}$$

The internal structure of the sort *find-s/ns* is now defined. Objects of the sort *find-s/ns* appear within the LP component as value of the attribute FIND-S/NS on *lpp2_1* objects. It should therefore be emphasized which parts of the FIND-S/NS value will be needed in the other attributes of the *lpp2_1* object. Intuitively the LP principle we are heading for in the present section says that at any level of embedding the phonology of a subject pronoun precedes the phonology of non-subject pronouns of the same level of embedding. In the formalization given so far, it has been shown how to identify the phonologies of the relevant signs at a given a verb and a list of *signs*. Thus, it has to be guaranteed that the VERB value and the ARGS value of the FIND-S/NS are identical with the VERB and ARGS values on the EMBED value. By doing so, the search mechanisms inside the *find-s/ns* object are ensured to work on elements of the same level of embedding. Furthermore, only the lists tagged as $\boxed{2''}$ and $\boxed{3''}$ in (4.51) are needed for precedence checking in the TEST attribute of the *lpp2_1* object. In (4.52) the relations between the *lpp2_1* object and its FIND-S/NS attribute are indicated.

(4.52) *The interactions involving the* FIND-S/NS *attribute*

$$
\begin{bmatrix}
\textit{lpp2\_1} & \\
\text{PHON} & \textit{list-of-phons} \\
\text{EMBED} & \begin{bmatrix} \textit{embed1} & \\ \text{VERB} & \boxed{2}\ \textit{sign} \\ \text{ARGS} & \boxed{3}\ \textit{list-of-signs} \\ \text{DEEPER} & \textit{embed} \end{bmatrix} \\
\text{FIND-S/NS} & \begin{bmatrix} \textit{find-s/ns} & \\ \text{VERB} & \boxed{2}\ \textit{sign} \\ \text{ARGS} & \boxed{3}\ \textit{list-of-signs} \\ \text{SU} & \begin{bmatrix} \textit{signphon} & \\ \text{PHONS} & \boxed{4}\ \textit{list-of-phons} \end{bmatrix} \\ \text{NSU} & \begin{bmatrix} \textit{signphon} & \\ \text{PHONS} & \boxed{5}\ \textit{list-of-phons} \end{bmatrix} \\ \ldots & \end{bmatrix} \\
\text{TEST} & \begin{bmatrix} \textit{prec} & \\ \ldots & \\ \text{A} & \boxed{4}\ \textit{list-of-phons} \\ \text{B} & \boxed{5}\ \textit{list-of-phons} \\ \ldots & \end{bmatrix} \\
\text{REC} & \textit{lpp2}
\end{bmatrix}
$$

The lists with the phonologies of the subject and non-subject pronouns, i.e. the tags $\boxed{4}$ and $\boxed{5}$ in (4.52) are token identical with the values of attributes of *lpp2_1*'s TEST value where precedence checking is finally done. In 4.4.3 the sort that occurs as value of this attribute is discussed.

4.4.3 The sort *prec*

At this point the pronouns that are subject to the LP principle (4.39) have been identified and their phonology is available on two lists. What we need now is a theory to constrain the order of these pronouns in the desired way. This is done by the theory of *prec*. Objects of sort *prec* are defined as value of the attribute TEST of *lpp2_1*. The important attributes that mediate the results of *prec* are the attributes C, A and B. Their value is a *list-of-phons*. The theories that we are going to discuss guarantee that each element of A precede each element of B on C.

The real work is done in two more theories, the theory of *alist* and the theory of *blist*. *alist* and *blist* are values of the attributes ARG1 and ARG2 respectively as can be seen in the definition of *prec*:

(4.53) *Appropriateness Conditions for the sort prec*

prec	C	*list-of-phons*
	A	*list-of-phons*
	B	*list-of-phons*
	ARG1	*alist*
	ARG2	*blist*

alist and *blist* are rather nontrivial sorts. Within ARG1 the shortest prefix of the C-list that contains all elements of the A-list is identified. In ARG2 it is checked whether this prefix of the C-list contains any members of the B-list (which is not allowed).

Before the theory of the sort *prec* can be given in (4.65), the newly introduced sorts *alist* and *blist* are to be defined. First, *alist* is addressed. In (4.54), we give the appropriateness conditions for this sort.

(4.54) *Appropriateness Conditions for the sort alist*

alist	BIG	*list-of-phons*	
	SMALL	*list-of-phons*	
	PREF	*list-of-phons*	
	alist0	BIG	*list-of-phons*
		SMALL	*elist*
		PREF	*elist*
	alist1	BIG	*nelist-of-phons*
		SMALL	*nelist-of-phons*
		PREF	*nelist-of-phons*
		TEST	*in-and-out*
		REC	*alist*

The idea of the subsorts *alist0* and *alist1* of *alist* is the following: If an empty SMALL list is checked, the empty prefix will do (*alist0*). If the SMALL list is non-empty, *alist1* applies and we start to construct a prefix of the BIG list that contains every element on the SMALL list and recursively subtract every element of the A-list from the SMALL list as soon as it is found on the BIG list. Once the SMALL list is empty, we have found the smallest prefix of the BIG list that contains all elements of the SMALL list. To give a brief outlook, in *blist* it will be checked whether that prefix also contains elements of the B-list.

To formalize this idea we need another auxiliary sort which is called *in-and-out*. It is the sort of the value of an *alist1*'s TEST attribute. As will be seen later, this sort will be useful for the formalization of the sort *blist* as well. Any *in-and-out* object has three attributes, SET, ELE and NSET. Two of them, SET and NSET, have the value *list-of-phons*, ELE is *phon*-valued. *in-and-out* checks whether a given element ELE is on the SET list but not on the NSET list. The two lists only differ by the presence or absence of that element. If the element does not occur on the first list, the sequence of elements on SET and NSET is identical. Thus *in-and-out* differs from *take-out* in (2.5) and (2.6) only in cases in which the ELE-value does not occur on the SET list. In (4.55) the appropriateness conditions of the sort *in-and-out* are given.

(4.55) *Appropriateness Conditions for the sort in-and-out*

in-and-out	SET	*list-of-phons*		
	ELE	*phon*		
	NSET	*list-of-phons*		
	in-and-out0		SET	*elist*
			ELE	*phon*
			NSET	*elist*
	in-and-out1		SET	*nelist-of-phons*
			ELE	*phon*
			NSET	*list-of-phons*
	in-and-out2		SET	*nelist-of-phons*
			ELE	*phon*
			NSET	*list-of-phons*
			REC	*in-and-out*

The sort *in-and-out* comes in three species. Each of these treat a special case. Generally speaking, in an object of sort *in-and-out* the SET list is recursively worked through until it is empty or until the ELE value appears as SET FIRST value at an *in-and-out* object. In the first case, one will find an object of species *in-and-out0*, and an object of species *in-and-out1* in the second case. If neither of them is the case for a given *in-and-out* object, it is of species *in-and-out2* and introduces a new recursion step in its REC value. As (4.55) does not introduce any new sort, we can give the theory of *in-and-out* directly.

(4.56) *Theory of in-and-out*

$$:\sim \textit{in-and-out1} \quad \rightarrow \quad \begin{array}{l}(:\text{SET FIRST} \approx :\text{ELE} \\ \wedge\ :\text{SET REST} \approx :\text{NSET})\end{array}$$

as AVM:

$$\begin{bmatrix}\textit{in-and-out1}\end{bmatrix} \Longrightarrow \begin{bmatrix}\textit{in-and-out1} & \\ \text{SET} & < \boxed{1}\ \textit{phon} \mid \boxed{2}\ \textit{list-of-phons} > \\ \text{ELE} & \boxed{1}\ \textit{phon} \\ \text{NSET} & \boxed{2}\ \textit{list-of-phons}\end{bmatrix}$$

$$:\sim \textit{in-and-out2} \quad \rightarrow \quad \begin{array}{l}(\neg(:\text{SET FIRST} \approx :\text{ELE}) \\ \wedge\ :\text{NSET FIRST} \approx :\text{SET FIRST} \\ \wedge\ :\text{REC SET} \approx :\text{SET REST} \\ \wedge\ :\text{REC ELE} \approx :\text{ELE} \\ \wedge\ :\text{REC NSET} \approx :\text{NSET REST})\end{array}$$

as AVM:

$$\begin{bmatrix}\textit{in-and-out2}\end{bmatrix} \Longrightarrow \begin{bmatrix}\textit{in-and-out2} & \\ \text{SET} & < \boxed{1}\ \textit{phon} \mid \boxed{2}\ \textit{list-of-phons} > \\ \text{ELE} & \boxed{3}\ \textit{phon} \\ \text{NSET} & < \boxed{1} \mid \boxed{4}\ \textit{list-of-phons} > \\ \text{REC} & \begin{bmatrix}\textit{in-and-out} & \\ \text{SET} & \boxed{2}\ \textit{list-of-phons} \\ \text{ELE} & \boxed{3}\ \textit{phon} \\ \text{NSET} & \boxed{4}\ \textit{list-of-phons}\end{bmatrix}\end{bmatrix}$$

$$\& \ \boxed{1} \neq \boxed{3}$$

In the proposition and the proof for (4.56) we employ two definitions. Def 6 is a specialized variant of membership on a list (cf. Def 2) which refers to the first occurrence of an element on a list. In Def 7 this definition is used to introduce a new notion, 'first-*o* difference'. First-*o* difference basically defines the relation of the SET and the NSET list of an *in-and-out* object with respect to the ELE value.

Def 6 *first occurrence on a list*

For any object o and any list $l = < o_1, \ldots, o_n >$,
$o_i, 1 \leq i \leq n$, **is the first occurrence of o on l**, iff

1. o is an element of l and
2. $o_i = o$ and
3. there is no $j, 1 \leq j \leq n$, with $j < i$ such that $o_j = o$.

Def 7 *first-o difference*

For any object o and any two lists l, h, with $l = < o_1, \ldots, o_n >$
l **is first-o different from** h iff

if o is an element of l and o_i is the first occurrence of o in l

then $h = < o_1, \ldots, o_{i-1}, o_{i+1}, \ldots, o_n >$

else $h = < o_1, \ldots, o_n >$

Def 7 expresses exactly the relation needed to state the next proposition. It will be shown that the attributes of an *in-and-out* are such that if the ELE value appears on the SET list, then the NSET list has all the elements of the SET list on it in exactly the same order except for the first occurrence of ELE. If the ELE value does not occur on the SET list, then the *nset* and the SET list have got exactly the same elements in exactly the same order. But, of course, they need not be identical lists.

PROPOSITION
For each u with $\mathrm{S}(u) \sqsubseteq$ *in-and-out*, it is the case that $[\mathrm{SET}]u$ is first-$[\mathrm{ELE}]u$ different from $[\mathrm{NSET}]u$.

PROOF

The two cases in the definition of first-o different will be treated separately in case 1 and case 2 respectively.

Case 1: $[\mathrm{ELE}]u$ is an element of $[\mathrm{SET}]u$

(by induction on the position of the first occurrence of $[\mathrm{ELE}]u$ in $[\mathrm{SET}]u$)

Base:

For each u with $\mathrm{S}(u) \sqsubseteq$ *in-and-out*, if $[\mathrm{ELE}]u = e$ and $[\mathrm{SET}]u = < o_1, \ldots, o_n >$ with $o_1 = e$, then [SET FIRST]u = e, thus S([SET]u) = *nelist-of-phons*. Therefore S(u) = *in-and-out1*. Then we know that [SET REST]u = [NSET]u = $< o_2, \ldots, o_m >$. Hence [SET]u is first-[ELE]u different from [NSET]u.

Hypothesis:

Assume that we have shown for each u with S(u) $\sqsubseteq$ *in-and-out* that if [ELE]$u = e$ and [SET]$u = < o_1, \ldots, o_n, \ldots, o_m >$, where o_n is the first occurrence of e in [SET]u, then [SET]u is first-[ELE]u different from [NSET]u.

Step:

For each u with S(u) $\sqsubseteq$ *in-and-out*, if [ELE]u = e and [SET]u = $< o_1, \ldots, o_{n+1}, \ldots, o_m >$, where o_{n+1} is the first occurrence of e in [SET]u.

then we know that S([SET]u) = *nelist-of-phons* and [SET FIRST]$u \neq$ [ELE]u = e. Therefore S(u) = *in-and-out2*. On [REC SET]u the first occurrence of e is on the n-th position. Thus our hypothesis applies to [REC SET]u. As [REC ELE]u = [ELE]$u = e$, we know that [REC SET]u is first-[ELE]u different from [REC NSET]u, where [REC NSET]$u = < o_2, \ldots, o_n, o_{n+2}, \ldots, o_m >$. But [NSET FIRST]u = [SET FIRST]u and [NSET REST]u = [REC NSET]u, thus [NSET]$u = < o_1, o_2, \ldots, o_n, o_{n+2}, \ldots, o_m >$. We see that [SET]$u$ is first-[ELE]u different from [NSET]u.

By induction the proposition follows for u in case 1.

Case 2: [ELE]u is not an element of [SET]u

(by induction on the length of [SET]u)

Base:

For each u with S(u) $\sqsubseteq$ *in-and-out*, if S([SET]u) = *elist*,

then S(u) = *in-and-out0* and S([NSET]u) = *elist*. As both lists are empty, it holds trivially that [SET]u is first-[ELE]u different from [NSET]u.

Hypothesis:

Assume that we have shown for each u with S(u) $\sqsubseteq$ *in-and-out* that if [ELE]$u = e$ and [SET]$u = l$ of length n, then [SET]u is first-e different from [NSET]u.

Step:

For each u with S(u) $\sqsubseteq$ *in-and-out*, if [ELE]u = e and [SET]u = $< o_1, \ldots, o_{n+1} >$ of length $n + 1$,

then, as we are studying the case in which e is not an element of [SET]u, it holds that [SET FIRST]$u \neq$ [ELE]u, and S([SET]u) = *nelist-of-phons*. Thus we know that S(u) = *in-and-out2*. As [REC ELE]u = [ELE]u = e

and [REC SET]u = [SET REST]$u = l$ of length n, our hypothesis applies to [REC SET]u and [REC NSET]u. We know that e is not an element of [REC SET]u. Therefore we conclude that [REC NSET]$u = < o_2, \ldots, o_{n+1} >$. As [NSET FIRST]$u$ = [SET FIRST]u and [NSET REST]u = [REC NSET]u, [NSET]$u = < o_1, \ldots, o_{n+1} >$. Thus [SET]$u$ is first-[ELE]u different from [SET]u.

By induction the proposition follows for u in case 2.
Having treated all possible cases, the proposition is shown for u. ■

The sort *in-and-out* is of crucial importance for the correct working of the precedence checking. Therefore it is worth looking at some hypothetical *in-and-out* objects.

(4.57) *Some example in-and-out objects*

(where a,b, ... are objects sort *phon*)

1. ELE occurs in SET:

$$\begin{bmatrix} \textit{in-and-out2} & \\ \text{SET} & <a,\ b,\ c,\ b,\ d> \\ \text{ELE} & b \\ \text{NSET} & <a,\ c,\ b,\ d> \\ \text{REC} & \begin{bmatrix} \textit{in-and-out1} & \\ \text{SET} & <b,\ c,\ b,\ d> \\ \text{ELE} & b \\ \text{NSET} & <c,\ b,\ d> \end{bmatrix} \end{bmatrix}$$

2. ELE does not occur in SET:

$$\begin{bmatrix} \textit{in-and-out2} & \\ \text{SET} & <a,\ b,\ c> \\ \text{ELE} & e \\ \text{NSET} & <a,\ b,\ c> \\ \text{REC} & \begin{bmatrix} \textit{in-and-out2} & \\ \text{SET} & <b,\ c> \\ \text{ELE} & e \\ \text{NSET} & <b,\ c> \\ \text{REC} & \begin{bmatrix} \textit{in-and-out2} & \\ \text{SET} & <c> \\ \text{ELE} & e \\ \text{NSET} & <c> \\ \text{REC} & \begin{bmatrix} \textit{in-and-out0} & \\ \text{SET} & \textit{elist} \\ \text{ELE} & e \\ \text{NSET} & \textit{elist} \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Example (4.57-1) illustrates the case, where the ELE value is on the SET list. As can be seen, only the first occurrence of the ELE value, b, is removed. Its second occurrence in SET is taken over into the NSET list. In this case, *in-and-out* defines exactly the same relation on two lists and an element as *take-out*, defined in (2.6). It is, however, a precondition of *take-out* that the ELE value actually occurs on the SET list. This precondition is not necessarily met by an *in-and-out* object, as can be seen in (4.57-2). In this case the NSET and the SET list have identical elements in identical order.

Within the sorts *alist* and *blist* the properties of *in-and-out* will be exploited in three ways. First, to extract an element from a list, like in (4.57-1). Second, to leave a list unchanged if a given element is not on that list, as illustrated in (4.57-2). Third to check that a given element is not on a particular list. The third use needs some further comment. If it is known that the SET and the NSET attribute of an *in-and-out* object consist of identical elements in the same order, then it can be concluded that the ELE element of this same object does not occur in the SET list. A special case where the two list valued attributes have identical elements in the same order occurs if the values of these lists are token identical. Thus, it follows from the proposition that if the SET and the NSET value of an *in-and-out* object are token identical, the ELE element is not a member of the SET list. The relevant constellation is shown in (4.58)

(4.58) *Checking non-membership via an in-and-out object*

$$\begin{bmatrix} \textit{in-and-out2} & \\ \text{SET} & \boxed{1}\ \textit{list-of-phons} \\ \text{ELE} & e \\ \text{NSET} & \boxed{1}\ \textit{list-of-phons} \\ \text{REC} & \textit{in-and-out} \end{bmatrix}$$

implies that e is not in $\boxed{1}$.

Using the auxiliary sort *in-and-out* we can define the theory of the sort *alist*. The purpose of this sort will be to find the shortest initial segment (prefix) of the BIG list that contains all the elements of the SMALL list.

(4.59) *Theory of alist*

$$\begin{array}{lll} :\sim \textit{alist1} & \rightarrow & (:\text{TEST SET} \approx :\text{SMALL} \\ & & \wedge\ :\text{TEST ELE} \approx :\text{BIG FIRST} \\ & & \wedge\ :\text{PREF FIRST} \approx :\text{BIG FIRST} \\ & & \wedge\ :\text{PREF REST} \approx :\text{REC PREF} \\ & & \wedge\ :\text{REC BIG} \approx :\text{BIG REST} \\ & & \wedge\ :\text{REC SMALL} \approx :\text{TEST NSET}) \end{array}$$

as AVM:

$$\begin{bmatrix} \textit{alist1} & \end{bmatrix} \Longrightarrow \begin{bmatrix} \textit{alist1} & \\ \textsc{big} & <\boxed{2}\ \textit{phon}|\ \boxed{4}\ \textit{list-of-phons}> \\ \textsc{small} & \boxed{1}\ \textit{list-of-phons} \\ \textsc{pref} & <\boxed{2}\ \textit{phon} \mid \boxed{3}\ \textit{list-of-phons} > \\ \textsc{test} & \begin{bmatrix} \textit{in-and-out} & \\ \textsc{set} & \boxed{1}\ \textit{list-of-phons} \\ \textsc{ele} & \boxed{2}\ \textit{phon} \\ \textsc{nset} & \boxed{5}\ \textit{list-of-phons} \end{bmatrix} \\ \textsc{rec} & \begin{bmatrix} \textit{alist} & \\ \textsc{big} & \boxed{4}\ \textit{list-of-phons} \\ \textsc{small} & \boxed{5}\ \textit{list-of-phons} \\ \textsc{pref} & \boxed{3}\ \textit{list-of-phons} \end{bmatrix} \end{bmatrix}$$

The intuitive idea behind the definition given above is expressed more formally in the following proposition.

PROPOSITION
For each object u with $\mathrm{S}(u) \sqsubseteq$ *alist*, if all elements on $[\textsc{small}]u$ are on $[\textsc{big}]u$ then is the case that $[\textsc{pref}]u$ is the smallest prefix of $[\textsc{big}]u$ that contains all elements of $[\textsc{small}]u$.

PROOF
(via induction on the length of $[\textsc{pref}]u$)

Base:

For each u with $\mathrm{S}(u) \sqsubseteq$ *alist*, if $\mathrm{S}([\textsc{pref}]u) =$ *elist*,

then $\mathrm{S}(u) =$ *alist0* and thus $\mathrm{S}([\textsc{small}]u) =$ *elist*. As both lists are empty, $[\textsc{pref}]u$ is trivially the smallest prefix of $[\textsc{big}]u$ which contains all the elements of $[\textsc{small}]u$.

Hypothesis:

Assume we have shown for each u with $\mathrm{S}(u) \sqsubseteq$ *alist* and $[\textsc{pref}]u = p$ of length n that if all elements on $[\textsc{small}]u$ are on $[\textsc{big}]u$ then $[\textsc{pref}]u$ is the smallest prefix of $[\textsc{big}]u$ which contains all elements of $[\textsc{small}]u$.

Step:

For each u with $\mathrm{S}(u) \sqsubseteq$ *alist*, if $[\textsc{pref}]u = p'$ of length $n+1$,

then $\mathrm{S}(u) =$ *alist1* and $[\textsc{test set}]u = [\textsc{small}]u$, $[\textsc{test ele}]u = [\textsc{big first}]u$ and $\mathrm{S}([\textsc{test}]u) \sqsubseteq$ *in-and-out*. By the properties of the sort *in-and-out* (4.55), (4.56) we know that $[\textsc{small}]u$ is first-$[\textsc{big first}]u$ different from $[\textsc{test nset}]u$. Considering the definition of first-o difference (Def 7) we have to distinguish two cases.

Case 1: [BIG FIRST]u is an element of [SMALL]u

We then know that [TEST NSET]u consists exactly of the elements of [SMALL]u with [BIG FIRST]u removed. Furthermore, [PREF FIRST]u = [BIG FIRST]u and [PREF REST]u = [REC PREF]u. Then [REC PREF]u is of length n and our hypothesis applies.

Hence [REC PREF]u is the shortest prefix of [REC BIG]u which contains all elements of [REC SMALL]u. As [REC BIG]u = [BIG REST]u and [REC SMALL]u = [TEST NSET]u, and as [REC SMALL]u is first-[BIG FIRST]u different from [SMALL]u, we know that [PREF]u is the shortest prefix of [BIG]u which contains all elements of [SMALL]u.

Case 2: [BIG FIRST]u is not an element of [SMALL]u

We then know that [TEST NSET]u and [SMALL]u consist of exactly the same elements. Furthermore, [PREF FIRST]u = [BIG FIRST]u and [PREF REST]u = [REC PREF]u. Then [REC PREF]u is of length n and our hypothesis applies.

Hence [REC PREF]u is the shortest prefix of [REC BIG]u which contains all elements of [REC SMALL]u. As [BIG FIRST]u is not an element of [SMALL]u but precedes all other elements of [BIG]u, it must be an element of [PREF]u. With [BIG REST]u = [REC BIG]u and [PREF REST]u = [REC PREF]u, [PREF]u is the shortest prefix of [BIG]u which contains all elements of [SMALL]u.

As we have treated all cases, the induction step holds for u, and by induction the proposition is shown.

■

The proposition for the theory of *alist* relies on the precondition that every element that is in [SMALL]u is contained in [BIG]u. It is met for all objects of sort *alist* that occur in our architecture, because they are exclusively used as value of the attribute ARG1 of *prec*. The theory of *phrase* as stated below ensures that the precondition is met. For a better understanding of how the sort *alist* works, an example is given in (4.60).

(4.60) *An example alist object*

(Tags are used to enable later reference to parts of the AVM)

$$
\left[\begin{array}{ll}
\textit{alist1} & \\
\text{BIG} & <a, b, c, d, e> \\
\text{SMALL} & <c, a> \\
\text{PREF} & <a, b, c> \\
\text{TEST} & \boxed{a} \left[\begin{array}{ll} \textit{in-and-out} & \\ \text{SET} & <c, a> \\ \text{ELE} & a \\ \text{NSET} & <c> \end{array}\right] \\
\text{REC} & \left[\begin{array}{ll}
\textit{alist1} & \\
\text{BIG} & <b, c, d, e> \\
\text{SMALL} & <c> \\
\text{PREF} & <b, c> \\
\text{TEST} & \boxed{b} \left[\begin{array}{ll} \textit{in-and-out} & \\ \text{SET} & <c> \\ \text{ELE} & b \\ \text{NSET} & <c> \end{array}\right] \\
\text{REC} & \left[\begin{array}{ll}
\textit{alist1} & \\
\text{BIG} & <c, d, e> \\
\text{SMALL} & <c> \\
\text{PREF} & <c> \\
\text{TEST} & \boxed{c} \left[\begin{array}{ll} \textit{in-and-out} & \\ \text{SET} & <c> \\ \text{ELE} & c \\ \text{NSET} & \textit{elist} \end{array}\right] \\
\text{REC} & \left[\begin{array}{ll} \textit{alist0} & \\ \text{BIG} & <d, e> \\ \text{SMALL} & \textit{elist} \\ \text{PREF} & \textit{elist} \end{array}\right]
\end{array}\right]
\end{array}\right]
\end{array}\right]
$$

The AVM in (4.60) illustrates the use of the sort *in-and-out* within the TEST attribute. In the case of the *in-and-out* objects marked with the tags $\boxed{a}$ and $\boxed{c}$, the first element of the BIG list occurs in the SMALL list. Thus, it can be removed from the list of elements which are to be included in the prefix. In the case of the *in-and-out* object tagged as $\boxed{b}$, the first element of the BIG list cannot be found on the SMALL list. Therefore the list of elements that are to be in the prefix cannot be shortened. As can be seen, the theory of *alist* heavily relies on the properties of *in-and-out* as illustrated in (4.57).

In the architecture developed in this paper, an object of sort *alist* always occurs as the value of the ARG1 attribute of a *prec* object. In (4.61) the interactions of the ARG1 attribute with other parts of the *prec* object are shown.

(4.61) *The interactions involving the* ARGS1 *attribute:*

$$\begin{bmatrix} \textit{prec} & \\ \textsc{c} & \boxed{1} <a,\ b,\ c,\ d,\ e> \\ \textsc{a} & \boxed{2} <c,\ a> \\ \textsc{b} & \textit{list-of-phons} \\ \textsc{arg1} & \begin{bmatrix} \textit{alist} & \\ \textsc{big} & \boxed{1} \\ \textsc{small} & \boxed{2} \\ \textsc{pref} & \boxed{3} <a,\ b,\ c> \end{bmatrix} \\ \textsc{arg2} & \begin{bmatrix} \textit{blist} \\ \textsc{big}\ \boxed{3} \\ \ldots \end{bmatrix} \end{bmatrix}$$

As can be seen from (4.61), the attribute ARG1 is used to identify the smallest prefix of the C list that contains all elements of the A list. This prefix is furthermore used in the ARG2 value, where it is checked, that no element of the B list occurs in it.

Let us now turn to the sort *blist*. This sort basically has two list-valued attributes BIG and SMALL. An object of sort *blist* has the property that none of the elements of the SMALL list occurs on the BIG list. The appropriateness conditions are given in (4.62), directly followed by the theory in (4.63).

(4.62) *Appropriateness Conditions for the sort blist*

blist	BIG	*list-of-phons*	
	SMALL	*list-of-phons*	
blist0	BIG	*list-of-phons*	
	SMALL	*elist*	
blist1	BIG	*list-of-phons*	
	SMALL	*nelist-of-phons*	
	TEST	*in-and-out*	
	REC	*blist*	

The sort *blist* uses the sort *in-and-out* within its TEST attribute. As this sort has already been discussed at length, the theory of *blist* can be stated directly.

(4.63) *Theory of blist*

$$\begin{array}{lll} :\sim \textit{blist1} & \rightarrow & (:\textsc{test set} \approx :\textsc{big} \\ & & \wedge\ :\textsc{test ele} \approx :\textsc{small first} \\ & & \wedge\ :\textsc{test nset} \approx :\textsc{test set} \\ & & \wedge\ :\textsc{rec big} \approx :\textsc{big} \\ & & \wedge\ :\textsc{rec small} \approx :\textsc{small rest}) \end{array}$$

as AVM:

$$\begin{bmatrix} \textit{blist1} \end{bmatrix} \Longrightarrow \begin{bmatrix} \textit{blist1} & \\ \textsc{big} & \boxed{1}\ \textit{list-of-phons} \\ \textsc{small} & < \boxed{2}\ \textit{phon} \mid \boxed{3}\ \textit{list-of-phons} > \\ \textsc{test} & \begin{bmatrix} \textit{in-and-out} & \\ \textsc{set} & \boxed{1}\ \textit{list-of-phons} \\ \textsc{ele} & \boxed{2}\ \textit{phon} \\ \textsc{nset} & \boxed{1}\ \textit{list-of-phons} \end{bmatrix} \\ \textsc{rec} & \begin{bmatrix} \textit{alist} & \\ \textsc{big} & \boxed{1}\ \textit{list-of-phons} \\ \textsc{small} & \boxed{3}\ \textit{list-of-phons} \end{bmatrix} \end{bmatrix}$$

The theory of *blist1* presented in (4.63) works by recursively checking that no element of the SMALL list is on the BIG list. To accomplish this the conjunct :TEST NSET $\approx$:TEST SET makes use of the properties of the sort *in-and-out* illustrated in (4.58). Thus it is checked that the BIG FIRST element does not occur on the SMALL list. An alternative to using *in-and-out* in such a peculiar way would be to define a new sort *not-element-of* specifically for that purpose. We will not do that and prove the following proposition instead:

PROPOSITION
For each u with $\mathrm{S}(u) \sqsubseteq$ *blist*, it holds that no v which is an element of $[\textsc{small}]u$ is an element of $[\textsc{big}]u$.

PROOF
(by induction on the length of $[\textsc{small}]u$)

Base:

For each u with $\mathrm{S}(u) \sqsubseteq$ *blist*, if $\mathrm{S}([\textsc{small}]u) =$ *elist*,

then $\mathrm{S}(u) =$ *blist0*. As $[\textsc{small}]u$ is empty, it trivially holds that no element of $[\textsc{small}]u$ is an element of $[\textsc{big}]u$.

Hypothesis:

Assume that we have shown for each u with $\mathrm{S}(u) \sqsubseteq$ *blist* that if $[\textsc{big}]u = k$ and $[\textsc{small}]u = l$ of length n, then no element of l is an element of k.

Step:

For each u with $\mathrm{S}(u) \sqsubseteq$ *blist*, if $[\textsc{small}]u = l'$ of length $n+1$ and $[\textsc{small first}]u = e$,

then $\mathrm{S}(u) =$ *blist1*. Then we know that $[\textsc{test set}]u = [\textsc{big}]u$, $[\textsc{test ele}]u = e$. Because $\mathrm{S}([\textsc{test}]u) \sqsubseteq$ *in-and-out*, $[\textsc{test set}]u$ is first-e

different from [TEST NSET]u. With [TEST NSET]u = [TEST SET]u we know that both lists are identical. This is only possible if e is not an element of [TEST SET]u. Thus e is not an element of [BIG]u either.

Since [REC BIG]u = [BIG]u and [REC SMALL]u = [SMALL REST]u of length n, our hypothesis applies. Hence we know that no element of [SMALL REST]u is an element of [BIG]u. As e = [SMALL FIRST]u is not an element of [BIG]u either, it follows that the consequent of the proposition holds for u.

By induction, the proposition follows.

■

The way the sort *blist* works can best be illustrated by a simple example. In (4.64) it is shown how all the elements of the SMALL list are checked not to be members of the BIG list (which bears the tag $\boxed{3}$). The actual checking is done within the two *in-and-out* objects. Both of them have identical SET and NSET values, as is enforced by the theory of *blist1*. This 'activates' the non-member property of the sort *in-and-out* presented in (4.58).

(4.64) *An example blist object*

$$\begin{bmatrix} \textit{blist1} & \\ \text{BIG} & \boxed{3}\ <a, b, c> \\ \text{SMALL} & <d, e> \\ \text{TEST} & \begin{bmatrix} \textit{in-and-out} & \\ \text{SET} & \boxed{3} \\ \text{ELE} & d \\ \text{NSET} & \boxed{3} \end{bmatrix} \\ \text{REC} & \begin{bmatrix} \textit{blist1} & \\ \text{BIG} & \boxed{3} \\ \text{SMALL} & <e> \\ \text{TEST} & \begin{bmatrix} \textit{in-and-out} & \\ \text{SET} & \boxed{3} \\ \text{ELE} & e \\ \text{NSET} & \boxed{3} \end{bmatrix} \\ \text{REC} & \begin{bmatrix} \textit{blist0} & \\ \text{BIG} & \boxed{3} \\ \text{SMALL} & \textit{elist} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

At this point all auxiliary sorts which are needed for the theory of precedence, i.e. the sort *prec*, have been defined. Since the main work to ensure precedence is done in the values *alist* and *blist* of the attributes

ARG1 and ARG2 of a *prec* object, the only thing left to be done for the theory of *prec* is to establish structure sharing between the list-valued attributes A, B and C and the values of certain attributes of *alist* and *blist*. The goal is to make sure that each element of the A-list precedes each element of the B-list on the C-list. This is defined by theory (4.65).

(4.65) *Theory of prec*

$$
\begin{array}{lll}
{:}{\sim}\mathit{prec} & \rightarrow & ({:}\textsc{c} \approx {:}\textsc{arg1 big} \\
 & & \wedge\ {:}\textsc{a} \approx {:}\textsc{arg1 small} \\
 & & \wedge\ {:}\textsc{b} \approx {:}\textsc{arg2 small} \\
 & & \wedge\ {:}\textsc{arg1 pref} \approx {:}\textsc{arg2 big})
\end{array}
$$

as AVM:

$$
\begin{bmatrix} \mathit{prec} \end{bmatrix}
\Longrightarrow
\begin{bmatrix}
\mathit{prec} & \\
\textsc{c} & \boxed{1}\ \mathit{list\text{-}of\text{-}phons} \\
\textsc{a} & \boxed{2}\ \mathit{list\text{-}of\text{-}phons} \\
\textsc{b} & \boxed{3}\ \mathit{list\text{-}of\text{-}phons} \\
\textsc{arg1} & \begin{bmatrix} \textsc{big} & \boxed{1}\ \mathit{list\text{-}of\text{-}phons} \\ \textsc{small} & \boxed{2}\ \mathit{list\text{-}of\text{-}phons} \\ \textsc{pref} & \boxed{4}\ \mathit{list\text{-}of\text{-}phons} \end{bmatrix} \\
\textsc{arg2} & \begin{bmatrix} \textsc{big} & \boxed{4}\ \mathit{list\text{-}of\text{-}phons} \\ \textsc{small} & \boxed{3}\ \mathit{list\text{-}of\text{-}phons} \end{bmatrix}
\end{bmatrix}
$$

In order to state the proposition for (4.65) naturally, we introduce one more definition. Def 8 states what it means for two lists to precede each other on a third list.

Def 8 *Precedence (lists)*

Given a list $l = < o_1, \ldots, o_n >$ and two lists $h = < a_1, \ldots, a_m >$ and $h' = < a'_1, \ldots, a'_{m'} >$ then

(all of) h **precedes (all of)** h' **on** l iff

there is an $i, 1 \leq i \leq n$ such that all elements of h are on $< o_1, \ldots, o_i >$ and all elements of h' are on $< o_{i+1}, \ldots, o_n >$.

Intuitively, Def 8 says that a list h precedes another list h' on a third list l, if list l can be divided into two parts, the first of which contains all the elements of h, and a second all the elements of h'. If it is known that all elements of h and h' actually occur on l and there are no identical elements on both lists h and h', h precedes h' on l if and only if it can be shown that there is a prefix of l which contains all elements of h but none of h'. This idea is implemented into the sort *prec* and exploited in the proof of the following proposition.

PROPOSITION

For each u with $\mathrm{S}(u) = prec$, if all elements of $[\textsc{a}]u$ and $[\textsc{b}]u$ are on $[\textsc{c}]u$ then all v which are elements of $[\textsc{a}]u$ precede all w which are elements of $[\textsc{b}]u$ on $[\textsc{c}]u$.

PROOF

For each u with $\mathrm{S}(u) = prec$, if all elements of $[\textsc{a}]u$ and $[\textsc{b}]u$ are on $[\textsc{c}]u$,

then we know, as $\mathrm{S}([\textsc{arg}1]u) \sqsubseteq alist$, from $[\textsc{c}]u = [\textsc{arg1 big}]u$ and $[\textsc{a}]u = [\textsc{arg1 small}]u$ together with the precondition that all elements of $[\textsc{arg1 small}]u$ are elements of $[\textsc{arg1 big}]u$. Thus we can use the result proved for the theory of *alist*. By this it is shown that $[\textsc{arg1 pref}]u$ is the smallest prefix of $[\textsc{c}]u$ which contains all elements of $[\textsc{a}]u$. Since $\mathrm{S}([\textsc{arg}2]u) \sqsubseteq blist$ and $[\textsc{arg1 pref}]u = [\textsc{arg2 big}]u$ and $[\textsc{b}]u = [\textsc{arg2 small}]u$, it follows that no element of $[\textsc{b}]u$ is an element of $[\textsc{arg1 pref}]u$.

Hence for $[\textsc{c}]u = < c_1, \ldots, c_n >$, there is a smallest $i, 1 \leq i \leq n$, with $[\textsc{arg1 pref}]u = p = < c_1, \ldots, c_i >$ such that all elements of $[\textsc{a}]u$ are elements of p. As no element of $[\textsc{b}]u$ is an element of p, but, by premise, all elements of $[\textsc{b}]u$ are elements of $[\textsc{c}]u$, it follows directly that all of $[\textsc{a}]u$ precedes all of $[\textsc{b}]u$ on $[\textsc{c}]u$.

Thus the proposition holds. ■

The sort *prec* can be illustrated with an example object. In (4.66) it is shown how the *alist* object in (4.60) and the *blist* object from (4.64) are used within a *prec* object.

(4.66) *An example prec object*

$$\left[\begin{array}{ll}
\multicolumn{2}{l}{prec} \\
\textsc{c} & \boxed{1} <a,\ b,\ c,\ d,\ e> \\
\textsc{a} & \boxed{2} <c,\ a> \\
\textsc{b} & \boxed{3} <e,\ d> \\
\textsc{arg1} & \boxed{4} \left[\begin{array}{ll}
\multicolumn{2}{l}{alist} \\
\textsc{big} & \boxed{1} \\
\textsc{small} & \boxed{2} \\
\textsc{pref} & \boxed{6} <a,\ b,\ c>
\end{array}\right] \\
\textsc{arg2} & \boxed{5} \left[\begin{array}{ll}
\multicolumn{2}{l}{blist} \\
\textsc{big} & \boxed{6} \\
\textsc{small} & \boxed{3}
\end{array}\right]
\end{array}\right]$$

where $\boxed{4}$ *is the AVM in (4.60),*
and $\boxed{5}$ *is the AVM in (4.64).*

The way precedence is checked within the sort *prec* posits certain restrictions on the possible content of the A and B lists. In particular, there is no well-formed *prec* object which has got a token identical element on its A and its B list. It is, however, allowed to have multiple occurrences of token identical elements on either one of the two lists. In (4.67) the different cases are illustrated. We only give the three lists C, A and B and the smallest prefix of C which contains all the elements of A. *Slanted* lower case letters are used for the objects on the lists, thus identical letters stand for token identical objects.

(4.67) *Possible and impossible lists on prec objects*

1. Identical elements on A:

C	<*a, b, a, c, d*>
A	<*a, a*>
B	<*c, d*>
PREF	<*a, b, a*>

2. Identical elements on A and B:

C	<*a, b, a, c, d, c, e*>
A	<*a, a*>
B	<*c, d, c*>
PREF	<*a, b, a*>

3. Elements occurring both on A and B:

C	<*a, b, c, d, c, e*>
A	<*a, c*>
B	<*c, d*>
PREF	<*a, b, c*>

Precedence checking in a *prec* object proceeds by verifying that the PREF value contains no elements that are on the B list. Therefore, the lists A, B, and C of (4.67-1, 2) are allowed to occur in a *prec* object. The lists in (4.67-(3)), however, is excluded, as the object *c* is contained in both B and A, and thus in PREF. Nevertheless the lists in (3) stand in a precedence relation to each other as it was defined in Def 8. Following the definition given there, there is a prefix of C, such that all elements of A are on that prefix and that all elements of B are on the rest of the list. According to this, C would be partitioned into the sublists <*a, b, c*> and <*d, c, e*>.

The consequence of the definition of the sort *prec* to exclude cases as (4.67-(3)) might seem counterintuitive. Therefore we shall first sketch an alternative setup for this sort, and then argue why we do not expect identical objects on the PHON list at all. As a third point, it will be demonstrated in some detail how the identities occurring in (4.67) can all be excluded in the first place.

If one thinks that the sort *prec* should be an exact implementation of the property of list precedence as stated in Def 8, the following modifications have to be made. In order to be able to distinguish between the definition of precedence given so far and the alternative to be sketched next, the new approach is encoded in a sort *prec′*. The sort *prec′* is specified for the same attributes as *prec* in (4.53). The sorts of the values of the attributes ARG1 and ARG2, however, differ. Let's call the sort appropriate for the ARG1 attribute of a *prec′* object *alist′*.

The sort *alist′* is defined similar to the sort *alist* given in (4.54) and (4.59). The sort *alist′*, however, has an attribute SUFF instead of the attribute PREF. On the species *alist′0* the SUFF value will be token identical to the BIG value. On the species *alist′1*, the SUFF value will be structure shared with the REC SUFF value. This guarantees that at any stage of recursion, the SUFF value is the longest suffix of the BIG such that all elements of the A list occur on the C list before the first element of the SUFF list. In a sense, the SUFF list is exactly the part of the BIG list that we were not interested in case of the sort *alist*.

Assuming the sort *alist′* defined the way sketched above, value of the attribute ARG2 of a *prec′* object will be of a sort *subset′* which is like *subset* in (4.25) and (4.26) but takes attributes with *list-of-phons* as values instead of *list-of-synsem*[44] The theory of *prec′* then had to state that the ARG1 SUFF list be identical to the ARG2 BIG list, and that the B list value appear on ARG2 SMALL as well. By this, it would be checked that all elements of the B list have an occurrence on the C list after the occurrences of all the elements on the A list.

The definition of *prec′* allows the lists of (4.67-3) to occur on a *prec′* object as well. In (4.68) these lists are put into such a redefined *prec* object.

[44]In the sort *subset* there is an attribute DIFF which is not needed in the present context, but whose existence is of no harm here.

(4.68) *An example of a prec′ object*

$$
\begin{bmatrix}
\textit{prec}' & \\
\textsc{c} & \boxed{1} < a,\ b,\ c,\ d,\ c,\ e > \\
\textsc{a} & \boxed{2} < a,\ c > \\
\textsc{b} & \boxed{3} < c,\ d > \\
\textsc{arg1} & \begin{bmatrix} \textit{alist}' & \\ \textsc{big} & \boxed{1} < a,\ b,\ c,\ d,\ c,\ e > \\ \textsc{small} & \boxed{2} < a,\ c > \\ \textsc{suff} & \boxed{4} < d,\ c,\ e > \end{bmatrix} \\
\textsc{arg2} & \begin{bmatrix} \textit{subset}' & \\ \textsc{big} & \boxed{4} < d,\ c,\ e > \\ \textsc{small} & \boxed{3} < c,\ d > \\ \ldots & \end{bmatrix}
\end{bmatrix}
$$

If the sort *prec′* is used instead of the sort *prec*, saying that a list h precedes a list h' on a list l is equivalent to saying that there is a *prec* object o with [C]$= l$, [A]$= h$ and [B]$= h'$. We shall not give a proposition and a proof for the sort *prec′*, as the purpose of presenting the sort is to sketch an alternative to the way precedence checking is done with the sort *prec*.

The original sort *prec* in (4.53) and (4.65) differs from its modified variant *prec′* in (4.68) only in cases where structure sharing occurs among the elements of the overall PHON list. It is, however, not evident that such structure sharing would be of any use in linguistic theory. One may even be inclined to argue for explicit distinctness of the elements on a *sign*'s PHON list. The argument roughly runs as follows: The PHON list encodes the temporal ordering of the phonetic realization of its elements. In an utterance it is not possible to realize the same part of the utterance at two different moments in time. Therefore the elements that are realized are all distinct as the realizations are all distinct. Thus there are no identities on the PHON list. This argument presupposes that the moment of realization of an element of the PHON list is somehow encoded in the element itself and furthermore reflected in the ordering on the PHON list. Under such assumption, the elements on the PHON list necessarily have to be all distinct. Unfortunately it is far from clear whether the assumption made above is tenable, as little is known about the internal content of the elements on the PHON list, i.e. the objects of sort *phon*.

As a consequence of this brief discussion it can be said that a linguist has to chose between the sorts *prec* and *prec′* depending on his assumptions about the PHON list. If he wants to exploit structure sharing on the PHON list in a systematic way, then he has to use the sort *prec′* inside the LP component. If he does not want to generally exclude identities on the PHON list, but does not make any systematic use of them in his grammar, he can use any of the sorts *prec* or *prec'*. Thereby he should be aware of the slight difference between the properties of the two sorts. If the linguist has a conception of the PHON list in mind that excludes identical elements on the PHON list altogether, he, too, can use any of the sorts *prec* or *prec'* as they behave exactly the same way in the case of distinct elements on the PHON list. The linguist then, however, has to introduce some mechanism to guarantee that no identical elements can ever occur on a *phon* list.[45]

A completely different approach to the definition of a sort that checks a precedence relation between the elements of lists can be taken which avoids the problem of identities on the PHON list altogether. Whereas the sorts *prec* and *prec'* have been based on the notion of 'element of a list', the third way of defining precedence takes the idea of an 'initial segment of a list' as its basics.

An initial segment of a list is a prefix of that list. It is the particularity of the notion 'initial segment' that one normally needs the information about the last element of the prefix. Thus, for a given list $l = < u_1, \ldots, u_n >$ there are n different initial segments:

$< u_1 >, < u_1, \ldots, u_2 >, \ldots, < u_1, \ldots, u_{n-1} >, < u_1, \ldots, u_n >$

So, instead of referring to the element u_i of list l, one can talk about the initial segment of l that ends with u_i, i.e. the prefix $< u_1, \ldots, u_i >$. The advantage of having initial segments becomes clear in cases where identical elements appear on a list. Assume the list l given in (4.69-1). There are two occurrences of the object a on l. Nevertheless the initial segments associated with the two occurrences differ, as can be seen in (4.69-2).

(4.69) *Example for initial segments:*

1. *List:*
   $l =< a, b, a, c >$
2. *Initial segments:*
   $< a >$
   $< a, b >$
   $< a, b, a >$

[45] Any structure sharing among elements of a particular list can be excluded by adding a junk slot whose value is of a sort *no-dubbles* that implements a restriction that any element of the given list has only one occurrence on that list. Such a sort can be written by using the sort *in-and-out* (with possibly changed sorts on its attributes) as value of one of its attributes.

$$< a, b, a, c >$$

Using initial segments, precedence checking proceeds the following way: Each element e on the lists A and B is associated with a prefix of the C list that has e as its last element, i.e. with an initial segment. Then, precedence is checked by verifying that every single prefix associated with an element of the A list is an initial part of every prefix associated with an element of the B list. This procedure can be illustrated at a simple example in (4.70) below.

(4.70) *Precedence checking with initial segments*

1. *Example lists:*

   C <a, b, a, a, c, a>
   A <a, a, b>
   B <a, c>

2. *Prefixes associable with the object a:*

   list: <a, b, a, a, c, a>
   element: a
   prefixes: <a>
   <a, b, a>
   <a, b, a, a>
   <a, b, a, a, a>

3. *Prefixes associated with the lists* A *and* B*:*

   A: <a>
   <a, b>
   <a, b, a>

   B: <a, b, a, a>
   <a, b, a, a, c>

   or:

   A: <a>
   <a, b>
   <a, b, a>

   B: <a, b, a, a, c>
   <a, b, a, a, c, a>

   or:

   A: <a>
   <a, b>
   <a, b, a, a>

   B: <a, b, a, a, c>
   <a, b, a, a, c, a>

or:

A: <*a*, *b*>
<*a*, *b*, *a*>
<*a*, *b*, *a*, *a*>

B: <*a*, *b*, *a*, *a*, *c*>
<*a*, *b*, *a*, *a*, *c*, *a*>

⋮

Assume the lists A, B and C given in (4.70-1). There are four possible prefixes of the C list ending in *a*, one ending in *b* and one ending in *c*. The prefixes ending in *a* are listed in (4.70-2). Thus any occurrence of an element can be distinguished from any other occurrence of the same element by the associated prefix.

Precedence is then checked by collecting a prefix to every element of the A list and every element on the B list, as is done in (4.70-3). As can be seen there, every list within the A group is a prefix of every list within the B group. It should be noticed that there are several possible associations of prefixes with the elements on the A and B lists. Some of them are given in (4.70-3)

There is, however, a complication with this approach to the sort *prec*. It has to be assured that the prefixes to the elements of the A and B lists are indeed different. If this is not guaranteed, the following constellation is possible:

(4.71) *A complication of the initial segment approach*

1. Example lists:
   C <*a*, *b*, *c*, *a*, *a*, *a*>
   A <*a*, *a*, *b*>
   B <*a*, *c*>
2. Prefixes associated to the lists A and B:
   A: <*a*>
   <*a*>
   <*a*, *b*>

   B: <*a*, *b*, *c*>
   <*a*, *b*, *c*, *a*, *a*>

In (4.71-2) the two occurrences of *a* on the A list have each been associated with a prefix of the C list ending in *a*, but with the same prefix. As a consequence, the lists A, B and C of (4.71-1) would be allowed in a precedence checking object. Nevertheless, it is of course not the case that the A list precedes the B list on the C list. Example (4.71) illustrates that the initial segment approach cannot fully do without excluding a sort of identity.

The approach seemed very attractive at first sight, because it is neutral with respect to identical elements on the PHON list and it is expresses the intuitions about the close connection between an element on the PHON list and its position on that list. It loses, however, some of its appeal once its formalization is considered seriously. These remarks should be sufficient to demonstrate that an initial segment approach to precedence checking can be taken in principle, but that it is in no way simpler than the one taken in this thesis by defining the sort *prec* in (4.53) and (4.65).[46]

In a somewhat lengthy discussion of the precedence checking sort we explored some alternative formulations of a central part of our LP component. As the sort *prec* will occur within every LP principle, it has been worth looking at its properties in detail, which is best be done by comparison to other approaches. We first considered a variant of the sort *prec* that is able to handle identities on the PHON list in an intuitively convincing way. In a second step, the initial segment approach has been sketchily presented. Its conceptual advantages and complications have been mentioned. Thus, closing section 4.4.3, it is clear that there are many ways to encode a precedence checking procedure in a sort. It depends on particular assumptions on the PHON list which one is the best to chose. The sort *prec* has been defined in the present section with most formal care.

Objects of the sort *prec* that have been introduced in this section appears as values of the attribute TEST of the sort *lpp2_1* in (4.40). As all subcomponents of the sort *lpp2* are now defined, the theory of this sort can be given next in section 4.4.4.

4.4.4 The theory of *lpp2*

The theories of *find-s/ns* and *prec* of its attributes FIND-S/NS and TEST are the main building blocks for the theory of *lpp2*, as we have seen in the discussion of the appropriateness conditions (4.40) for *lpp2*. Since both of them are now defined by the theories in (4.50) and (4.65) respectively, they can be employed in the theory of *lpp2*. The theory given in (4.72) is a formalization of the LP principle stated in (4.39). According to this principle, it has to be checked that the phonology of subject pronouns

[46] As can be seen in the different possibilities given in (4.70-3), the initial segment approach is not deterministic - in contrast to the other two. This, however, would cause no problem for the present approach, as no procedural aspects are involved. So, leaving the above mentioned complications aside, the initial segment approach stresses the fact that we are working within a declarative system.

precedes the phonology of nonsubject pronouns if they appear at the same level of embedding.

(4.72) *Theory of lpp2*

$$
\begin{array}{lll}
{:}{\sim}\mathit{lpp2\_1} & \rightarrow & (\text{:EMBED VERB} \approx \text{:FIND-S/NS VERB} \\
 & & \wedge\ \text{:EMBED ARGS} \approx \text{:FIND-S/NS ARGS} \\
 & & \wedge\ \text{:TEST C} \approx \text{:PHON} \\
 & & \wedge\ \text{:TEST A} \approx \text{:FIND-S/NS SU PHONS} \\
 & & \wedge\ \text{:TEST B} \approx \text{:FIND-S/NS NSU PHONS} \\
 & & \wedge\ \text{:REC PHON} \approx \text{:PHON} \\
 & & \wedge\ \text{:REC EMBED} \approx \text{:EMBED DEEPER})
\end{array}
$$

as AVM (including tags from other theories):

$$
\begin{bmatrix} \textit{lpp2\_1} \end{bmatrix} \Longrightarrow
$$

$$
\begin{bmatrix}
\textit{lpp2\_1} & \\
\text{PHON} & \boxed{1}\ \textit{list-of-phons} \\
\text{EMBED} & \begin{bmatrix} \textit{embed} & \\ \text{VERB} & \boxed{2}\ \textit{sign} \\ \text{ARGS} & \boxed{3}\ \textit{list-of-signs} \\ \text{DEEPER} & \boxed{4}\ \textit{embed} \end{bmatrix} \\
\text{FIND-S/NS} & \begin{bmatrix}
\textit{find-s/ns} & \\
\text{VERB} & \boxed{2}\ \textit{sign} \\
\text{ARGS} & \boxed{3}\ \textit{list-of-signs} \\
\text{SU} & \begin{bmatrix} \textit{signphon} & \\ \text{SIGNS} & \boxed{5}\ \textit{list-of-signs} \\ \text{PHONS} & \boxed{6}\ \textit{list-of-phons} \end{bmatrix} \\
\text{NSU} & \begin{bmatrix} \textit{signphon} & \\ \text{SIGNS} & \boxed{7}\ \textit{list-of-signs} \\ \text{PHONS} & \boxed{8}\ \textit{list-of-phons} \end{bmatrix} \\
\text{FIND-PRON} & \begin{bmatrix} \textit{find-pron} & \\ \text{ARGS} & \boxed{3}\ \textit{list-of-signs} \\ \text{PRONS} & \boxed{9}\ \textit{list-of-signs} \end{bmatrix} \\
\text{SEP-PRON} & \begin{bmatrix} \textit{separate-pron} & \\ \text{PRONS} & \boxed{9}\ \textit{list-of-signs} \\ \text{VERB} & \boxed{2}\ \textit{sign} \\ \text{SU} & \boxed{5}\ \textit{list-of-signs} \\ \text{NSU} & \boxed{7}\ \textit{list-of-signs} \end{bmatrix}
\end{bmatrix} \\
\text{TEST} & \begin{bmatrix} \textit{prec} & \\ \text{C} & \boxed{1}\ \textit{list-of-phons} \\ \text{A} & \boxed{6}\ \textit{list-of-phons} \\ \text{B} & \boxed{8}\ \textit{list-of-phons} \end{bmatrix} \\
\text{REC} & \begin{bmatrix} \textit{lpp2} & \\ \text{PHON} & \boxed{1}\ \textit{list-of-phons} \\ \text{EMBED} & \boxed{4}\ \textit{embed} \end{bmatrix}
\end{bmatrix}
$$

The sort definition in (4.72) indicates that the precedence checking occurs along the structure of the EMBED value. At any level, the value of the attribute FIND-S/NS collects the phonologies of the subject and the nonsubject pronouns in separate lists. Within the TEST value the actual precedence checking is done. The REC value indicates that this way of checking continues for the next level of embedding as well.

For the proposition and the proof of (4.72) we need some definitions. These definitions mainly serve to express relations among pronouns and *embed* objects with respect to their level of embedding. Thus Def 9 states the idea of being a 'current pronoun'. Using this definition and the recursive structure of an *embed* object, the notion 'deeper pronoun' is defined in Def 10. These two definitions are then taken together to express what it means to be a pronoun in within an *embed* object (Def 11) Finally, Def 12 addresses the concept of two pronouns appearing at the same level of embedding in an *embed* object.

Let u be any object with $\mathrm{S}(u) \sqsubseteq$ *embed*. Assume that p is a pronoun.

Def 9 *Current Pronoun of*

p **is a current pronoun of** u iff

p is an element of [ARGS]u.

Def 10 *Deeper Pronoun of*

p **is a deeper pronoun of** u iff

1. [DEEPER ARGS]$u = u'$ and p is a current pronoun of u' or
2. [DEEPER]$u = u'$ and p is a deeper pronoun of u'.

Def 11 *Pronoun of*

p **is a pronoun of** u iff

1. p is a current pronoun of u or
2. p is a deeper pronoun of u.

Def 12 *Same Level Pronouns of*

Let p, q be pronouns.

p and q **are same level pronouns of** u iff

1. p and q are current pronouns or
2. $[\text{DEEPER}]u = u'$ and p and q are same level pronouns of u'.

With these definitions, we turn to a proposition which expresses the precedence relation among same level subject and nonsubject pronouns. As the sort *lpp2* is recursive, the prove is done by induction, furthermore it refers to the properties proved for the sorts *find-s/ns* and *prec*.

PROPOSITION
For each u with $\mathrm{S}(u) \sqsubseteq$ *lpp2*, if $[\text{PHON}]u$ consists exactly of the PHON values of the *signs* occurring within $[\text{EMBED}]u$,[47] it holds that

if there is a v, such that v is a pronoun of $[\text{EMBED}]u$ and $[\text{SYNSEM}]v$ is the subject of $[\text{EMBED VERB}]u$

then for every v' such that v and v' are same level pronouns of $[\text{EMBED}]u$ and $v' \neq$ v it holds that $[\text{PHON}]v$ precedes $[\text{PHON}]v'$ on $[\text{PHON}]u$.

PROOF
(by induction on the depth of embedding)

Base:

For each u, such that $\mathrm{S}(u) \sqsubseteq$ *lpp2*, if u has no levels of embedding,

then $\mathrm{S}(u) =$ *lpp2_0* and $\mathrm{S}([\text{EMBED}]u) =$ *embed0*. Thus $\mathrm{S}([\text{EMBED ARGS}]u)$ $=$ *elist* and the path EMBED DEEPER is not defined on u. Therefore, there are no current pronouns of u and no deeper pronouns of u, either. Thus, the proposition holds trivially.

Hypothesis:

Assume that we have shown for each u with $\mathrm{S}(u) \sqsubseteq$ *lpp2*, if u has n levels of embedding then it holds that

[47] This precondition is important to be able to use the result proved for the sort *prec* in (4.65) in the induction step. For the sake of legibility, however, it will not be repeated in the hypothesis.

if there is a v, such that v is a pronoun of [EMBED]u and [SYNSEM]v is the subject of [EMBED VERB]u

then for every v' such that v and v' are same level pronouns of [EMBED]u and $v' \neq$ v it holds that [PHON]v precedes [PHON]v' on [PHON]u.

Step:

For each u with $S(u) \sqsubseteq$ *lpp2*, if u is of $n+1$ levels of embedding,

then $S(u) =$ *lpp2_1*, therefore $S([\text{EMBED}]u) =$ *embed1*. Because S([FIND-S/NS]u) $\sqsubseteq$ *find-s/ns*, [EMBED VERB]u = [FIND-S/NS VERB]u and [EMBED ARGS]u = [FIND-S/NS ARGS]u, we can apply result proved in (4.50). Thus we know with [TEST A]u = [FIND-S/NS SU PHONS]u that [TEST A]u consists exactly of the phonology of those elements of [EMBED ARGS]u which are pronouns and the subject of [EMBED VERB]u. Furthermore, [TEST B]u = [FIND-S/NS NSU PHONS]u tells us that [TEST B]u consists of the phonology of exactly those elements of [EMBED ARGS]u which are pronouns and not the subject of [EMBED VERB]u.

As S([TEST]u) = *prec* and [TEST C]u = [PHON]u, the precondition of proposition (4.65) follows from the precondition of the present proposition, we can use the result shown in the proposition for the sort *prec*. It follows that all of [FIND-S/NS SU PHONS]u precedes all of [FIND-S/NS NSU PHONS]u on [PHON]u. Thus, the proposition holds for u with respect to the current pronouns of [EMBED]u.

It is stated in the theory of *lpp2_1* that [REC EMBED]u = [EMBED DEEPER]u. Therefore all pronouns of [REC EMBED]u are pronouns of [EMBED]u as well. Because [REC]u is only of n levels of embedding, our hypothesis applies [REC]u. Therefore to all pronouns of [REC EMBED]u, and thus to all pronouns of [EMBED DEEPER]u stand in the relation expressed in the proposition. Then the proposition holds for u with respect to all deeper pronouns of [EMBED]u.

As we have shown the proposition for u with respect to all current and all deeper pronouns of [EMBED]u, we have shown it for u with respect to all pronouns of [EMBED]u.

Thus, the proposition holds by induction. ■

With (4.72) and the proof of its properties a formal LP principle for the informal LP rule presented at the outset in (4.39) has been described. A closer look at the components or modules of *lpp2* reveals that it is easy to construct other LP principles that were often discussed in the literature by minor modifications which only involve some of the sorts that occur in the descriptions. This will be illustrated with another LP principle in section 4.5, whose formulation will purely rely on sorts already given.

4.5 A second LP principle (*lpp3*)

The LP principle in (4.72) has only been sensitive to pronouns of the same level of embedding. With the second LP principle of this paper we want to investigate a case in which the ordered pronouns are of different levels of embedding. (4.73) shows an informal version of such an LP-principle:

(4.73) *A second LP principle*

> if a *phrase* x has daughters i and j such that
>
> i and j are pronouns and
> i is less deeply embedded than j
>
> then i's PHON precedes j's PHON on x's PHON

This principle will be encoded in the sort *lpp3*. Objects of this sort occur as value of the attribute LPP3 of an *lp-hc* object. The appropriateness conditions of this sort are given in (4.74).

(4.74) *Appropriateness Conditions for the sort lpp3*

lpp3	PHON	*list-of-phons*	
	EMBED	*embed*	
	CUR-PRONS	*signphon*	
	DEEP-PRONS	*signphon*	
	lpp3_0	PHON	*list-of-phons*
		EMBED	*embed0*
		CUR-PRONS	*signphon0*
		DEEP-PRONS	*signphon0*
	lpp3_1	PHON	*list-of-phons*
		EMBED	*embed1*
		CUR-PRONS	*signphon*
		DEEP-PRONS	*signphon*
		FIND-CUR	*find-pron*
		FIND-DEEP	*append-of-signs*
		TEST	*prec*
		REC	*lpp3*

In CUR-PRONS the pronominal arguments of the current level of embedding are collected, while in DEEP-PRONS the pronominal arguments of the levels of embedding deeper than the current one are stored. TEST fulfills the function discussed for the previous LP principle. The theory of its value *prec* checks if the CUR-PRONS PHONS-elements precede the DEEP-PRONS PHONS-elements on the overall PHON list. The sort *append-of-signs* is defined as in King (1992), i.e., exactly as the sort *append-of-phons* we used in (2.9) and (2.10) but with *list-of-signs* objects instead of *list-of-phon* objects. Every other sort that occurs in (4.74) has been used before. For this reason we can define the theory of *lpp3* without a previous discussion of auxiliary sorts.

(4.75) *Theory of lpp3*

$$
\begin{array}{lll}
{:}{\sim}\mathit{lpp3\_1} & \rightarrow & (\text{:FIND-CUR ARGS} \approx \text{:EMBED ARGS} \\
 & & \wedge\ \text{:CUR-PRONS SIGNS} \approx \text{:FIND-CUR PRONS} \\
 & & \wedge\ \text{:DEEP-PRONS SIGNS} \approx \text{:FIND-DEEP RES} \\
 & & \wedge\ \text{:REC CUR-PRONS SIGNS} \approx \text{:FIND-DEEP LEFT} \\
 & & \wedge\ \text{:REC DEEP-PRONS SIGNS} \approx \text{:FIND-DEEP RIGHT} \\
 & & \wedge\ \text{:TEST C} \approx \text{:PHON} \\
 & & \wedge\ \text{:TEST A} \approx \text{:CUR-PRONS PHONS} \\
 & & \wedge\ \text{:TEST B} \approx \text{:DEEP-PRONS PHONS} \\
 & & \wedge\ \text{:REC PHON} \approx \text{:PHON} \\
 & & \wedge\ \text{:REC EMBED} \approx \text{:EMBED DEEPER})
\end{array}
$$

as AVM:

$$
\begin{bmatrix} \mathit{lpp3\_1} \end{bmatrix} \Longrightarrow
\begin{bmatrix}
\mathit{lpp3\_1} & \\
\text{PHON} & \boxed{1} \\
\text{EMBED} & \begin{bmatrix} \text{VERB} & \\ \text{ARGS} & \boxed{2} \\ \text{DEEPER} & \boxed{3} \end{bmatrix} \\
\text{CUR-PRONS} & \begin{bmatrix} \text{SIGNS} & \boxed{4} \\ \text{PHONS} & \boxed{5} \end{bmatrix} \\
\text{DEEP-PRONS} & \begin{bmatrix} \text{SIGNS} & \boxed{6} \\ \text{PHONS} & \boxed{7} \end{bmatrix} \\
\text{FIND-CUR} & \begin{bmatrix} \text{ARGS} & \boxed{2} \\ \text{PRONS} & \boxed{4} \end{bmatrix} \\
\text{FIND-DEEP} & \begin{bmatrix} \text{LEFT} & \boxed{8} \\ \text{RIGHT} & \boxed{9} \\ \text{RES} & \boxed{6} \end{bmatrix} \\
\text{TEST} & \begin{bmatrix} \text{C} & \boxed{1} \\ \text{A} & \boxed{5} \\ \text{B} & \boxed{7} \end{bmatrix} \\
\text{REC} & \begin{bmatrix} \text{PHON} & \boxed{1} \\ \text{EMBED} & \boxed{3} \\ \text{CUR-PRONS} & \begin{bmatrix} \text{SIGNS} & \boxed{8} \end{bmatrix} \\ \text{DEEP-PRONS} & \begin{bmatrix} \text{SIGNS} & \boxed{9} \end{bmatrix} \end{bmatrix}
\end{bmatrix}
$$

(4.75) is already the full-fledged LP principle for the LP rule which was informally put forward in (4.73). Before we formulate the proposition for the theory of *lpp3* we introduce some more terminology. As the LP principle to be formulated makes reference to pronouns of different levels of embedding, Def 13 defines what it means for one pronoun to be more deeply embedded than an other pronoun on an *embed* object.

For the following definition, let u be an object with $\mathrm{S}(u) \sqsubseteq$ *embed*.

Def 13 *Deeper Pronoun than*

Let p and q be pronouns.

p **is a deeper pronoun of** u **than** q iff

1. q is a current pronoun of u and p is a deeper pronoun of u or
2. $[\textsc{deeper}]u = u'$ and p is a deeper pronoun of u' than q.

This definition enables us to state the relevant property of the sort *lpp3* in a natural way. The proof consists of two parts and there are two lemmata shown which make the main proof more elegant. The lemmata guarantee that the intended elements are collected in the attributes CUR-PRONS and DEEP-PRONS of an *lpp3* object.

PROPOSITION
For each u with $\mathrm{S}(u) \sqsubseteq$ *lpp3*, if p, q are pronouns of $[\textsc{embed}]u$, then it holds that

if p is a deeper pronoun of $[\textsc{embed}]u$ than q
then $[\textsc{phon}]q$ precedes $[\textsc{phon}]p$ on $[\textsc{phon}]u$.

Before we prove the proposition, we have to provide the lemmata 1 and 2 that will be used in the proof:

Lemma 1

For each u with $\mathrm{S}(u) \sqsubseteq$ *lpp3*, it holds that the list $[\textsc{cur-prons signs}]u$ consists exactly of the current pronouns of $[\textsc{embed}]u$.

PROOF
Two cases have to be distinguished:

Case1: $\mathrm{S}(u) =$ *lpp3_0*

Then $\mathrm{S}([\textsc{embed}]u) =$ *embed0*. Therefore there are no current pronouns of $[\textsc{embed}]u$. Because $\mathrm{S}([\textsc{cur-prons}]u) =$ *signphon0*, we know that $\mathrm{S}([\textsc{cur-prons signs}]u) =$ *elist*. Thus the proposition holds for u.

Case2: $\mathrm{S}(u) =$ *lpp3_1*

Then $\mathrm{S}([\textsc{embed}]u) =$ *embed1*. As $\mathrm{S}([\textsc{find-cur}]u) <$ *find-pron*, $[\textsc{embed args}]u = [\textsc{find-cur args}]u$ and $[\textsc{cur-prons signs}]u = [\textsc{find-cur}$

PRONS]u, we know from the property of the sort *find-pron* that [CUR-PRONS SIGNS]u consists exactly of the pronouns that are elements of [EMBED-ARGS]u. Thus, [CUR-PRONS SIGNS]u consists exactly of the current pronouns of [EMBED]u. Thus the proposition holds for u.

As there are no other varieties of *lpp3*, the proposition holds.

■

Lemma 2

For each u with $\mathrm{S}(u) \sqsubseteq$ *lpp3*, it holds that the list [DEEP-PRONS SIGNS]u consists exactly of the deeper pronouns of [EMBED]u.

PROOF
(by induction on the level of recursion)

Base:

For each u with $\mathrm{S}(u) \sqsubseteq$ *lpp3*, if u has no recursion,

then $\mathrm{S}(u) =$ *lpp3_0* and thus $\mathrm{S}([\text{EMBED}]u) =$ *embed0*. As [EMBED]u is not defined for the attribute DEEPER, there are no deeper pronouns of [EMBED]u. By the appropriateness conditions for *lpp3_0* we know that $\mathrm{S}([\text{DEEP-PRONS}]u) =$ *signphon0*, hence $\mathrm{S}([\text{DEEP-PRONS SIGNS}]u) =$ *elist*. Since there exist no deeper pronouns of [EMBED]u and the [DEEP-PRONS SIGNS]u list is empty, the proposition holds for u.

Hypothesis:

Assume that we have shown for each u with $\mathrm{S}(u) \sqsubseteq$ *lpp3*, if u has n levels of recursion then [DEEP-PRONS SIGNS]u consists exactly of the deeper pronouns of [EMBED]u.

Step:

For each u with $\mathrm{S}(u) \sqsubseteq$ *lpp3* if u has $n+1$ levels of recursion

then $\mathrm{S}(u) =$ *lpp3_1*, and thus $\mathrm{S}([\text{EMBED}]u) =$ *embed1*. As [REC EMBED]u = [EMBED DEEPER]u, we know that all pronouns of [REC EMBED]u are deeper pronouns of [EMBED]u. [REC]u has only n levels of embedding, thus our hypothesis applies and we know that [REC DEEP-PRONS SIGNS]u consists exactly of the deeper pronouns of [EMBED DEEPER]u. By lemma (1) we know that [REC CUR-PRONS SIGNS]u consists exactly of the current pronouns of [EMBED DEEPER]u.

As S([FIND-DEEP]u) $\sqsubseteq$ *append-of-signs* and [FIND-DEEP LEFT]u = [REC CUR-PRONS SIGNS]u, [FIND-DEEP RIGHT]u = [REC DEEP-PRONS SIGNS]u and [FIND-DEEP RES]u = [DEEP-PRONS SIGNS]u. Thus [DEEP-PRONS SIGNS]u consists exactly of the current and the deeper pronouns of [EMBED DEEPER]u. By definition Def 10 we know that [DEEP-PRONS SIGNS]u consists exactly of the deeper pronouns of [EMBED]u. Thus, the proposition holds for u.

By induction, the proposition holds.

■

Having shown the two lemmata, we can now prove proposition (4.74).

PROOF
(by induction on the level of embedding)

Base:

For each u with S(u) $\sqsubseteq$ *lpp3*, if u has no recursion

then S(u) = *lpp3_0* and S([EMBED]u) = *embed0*. Therefore there are no pronouns at all, and thus no pronouns deeper than others. Therefore the proposition trivially holds for u.

Hypothesis:

Assume that we have shown for each u with S(u) $\sqsubseteq$ *lpp3*, if u has n levels of recursion then for any pronouns p, q of [EMBED]u it holds that

if p is a deeper pronoun of [EMBED]u
then q, then [PHON]q precedes [PHON]p on [PHON]u.

Step:

For each u with S(u) $\sqsubseteq$ *lpp3*, if u has $n+1$ levels of recursion

then S(u) = *lpp3_1* and S([EMBED]u) = *embed1*. By lemma 1 we know that exactly all current pronouns of [EMBED]u are gathered in [CUR-PRONS SIGNS]u, and by lemma 2 it follows that exactly all deeper pronouns of [EMBED]u make up [DEEP-PRONS SIGNS]u. As [CUR-PRONS]u and [DEEP-PRONS]u are of sort *signphon*, [CUR-PRONS PHONS]u and [DEEP-PRONS PHONS]u are the corresponding PHON values to [CUR-PRONS SIGNS]u and [DEEP-PRONS SIGNS]u respectively. As S([TEST]u) = *prec* and [PHON]u = [TEST C]u, [CUR-PRONS PHONS]u = [TEST A]u and [DEEP-PRONS PHONS]u = [TEST B]u, we know that all of [CUR-PRONS PHONS]u precedes all of [DEEP-PRONS PHONS]u on [PHON]u. Therefore the required precedence relation holds for all current pronouns of [EMBED]u with respect to all deeper pronouns of [EMBED]u.

Since [REC]u is only of n levels of embedding, our hypothesis applies to [REC]u. Thus, all pronouns of [REC EMBED]u fulfill the precedence requirement expressed in the proposition. As [REC EMBED]u = [EMBED DEEPER]u the precedence relation also holds for any deeper pronoun q of [EMBED]u and an arbitrary other pronoun p of [EMBED]u which is deeper than q.

The proposition follows by induction.

■

In the present section a second LP principle concerning the order of pronouns in the mittelfeld was presented. Just as the LP principle discussed in 4.4, the encoding of the LP principle informally stated in (4.73) makes use of the information about the structure of embedding in coherent constructions that was provided in objects of the sort *embed*. It is particularly interesting that to encode the LP principle in (4.73) only one additional sort, *lpp3*, has been needed. In the following section, the theory of *lp-hc* is given that makes crucial use of the properties proved for the sorts *lpp2* and *lpp3* defined in sections 4.4 and 4.5.

4.6 The theory of *lp-hc*

In the previous sections some crucial components of a theory of constituent order in HPSG were formalized. It is still necessary to combine these components in the theory of *lp* objects and in the theory of *phrase*. The task of the theory of *lp* will be to connect the values of the external interface attributes, the auxiliary structure attributes and the LP principle attributes in the way outlined in section 4.1.2. The values of the LP principle attributes constrain the possible order of the phonological elements on the PHON list of a phrase.

The theory of *phrase* has to fulfill two tasks. First, it establishes the link between the phonology of the entire phrase and the *build-phon* object and that between the ALL-DTRS list and the *build-phon* object. Thus it is made sure that the phonology of the phrase really is some permutation of the phonology of the daughters of that phrase. Whether this only concerns immediate daughters of the phrase or some more daughters that are dominated by that phrase depends on how the ALL-DTRS list is constructed. In case of the 'flat' analysis of a German sentence outlined in section 3.1 the ALL-DTRS list only contains immediate daughters of a *phrase*. The second task the theory of *phrase* has to fulfill for the LP component is to connect the PHON list of the phrase to the PHON list of the *lp* object and to provide the ALL-DTRS list as input of the theory of *ws2*, which constructs the information of the structure of embedding in coherent constructions.

In the present section we shall only be concerned with the theory of *lp*, and in particular with the theory of the species *lp-hc*. The connections between attribute values that are expressed in the theory of *phrase*, will be considered in the next chapter 5. As all subparts of an *lp-hc* have been introduced, the theory of *lp-hc* can be given in (4.76).

(4.76) *Theory of lp-hc*

$$
\begin{array}{lll}
{:}{\sim}\mathit{lp\text{-}hc} & \rightarrow & (\text{:EMBED} \approx \text{:WS2 EMBEDDING EMBED} \\
 & & \wedge\ \text{:LPP2 PHON} \approx \text{:PHON} \\
 & & \wedge\ \text{:LPP2 EMBED} \approx \text{:EMBED} \\
 & & \wedge\ \text{:LPP3 PHON} \approx \text{:PHON} \\
 & & \wedge\ \text{:LPP3 EMBED} \approx \text{:EMBED})
\end{array}
$$

as AVM:

$$\begin{bmatrix} \textit{lp-hc} \end{bmatrix} \Longrightarrow \begin{bmatrix} \textit{lp-hc} & \\ \textsc{phon} & \boxed{1} \\ \textsc{ws2} & \begin{bmatrix}\textsc{embedding embed}\ \boxed{2}\end{bmatrix} \\ \textsc{embed} & \boxed{2} \\ \textsc{lpp2} & \begin{bmatrix} \textsc{phon} & \boxed{1} \\ \textsc{embed} & \boxed{2} \end{bmatrix} \\ \textsc{lpp3} & \begin{bmatrix} \textsc{phon} & \boxed{1} \\ \textsc{embed} & \boxed{2} \end{bmatrix} \end{bmatrix}$$

As usual, we need some terminological definitions for the proposition and the proof. The following definitions will make use of the notion 'verb' as it has been defined in Def 1 in section 4.3. There, a *sign* was called a verb if an only if it has a HEAD value of sort *verb*. In Def 14 an argument of a verb is defined as an element of the verb's COMPS list. Def 15 finally is needed to identify native arguments of a verb. These are the arguments that are not inherited by argument extraction but genuine elements of the verbs COMPS list. The notions have a parallel in the terminology 'pronoun of' and 'current pronoun of' an *embed* object as defined in (Def 11 and Def 9). In the latter definitions, an pronoun of an *embed* object is any pronoun that occurs in an ARGS list of this object at some level of embedding. A current pronoun was defined as being a pronoun which is an element of the ARGS list of the *embed* object itself.

In the following definitions let v be a verb (cf. Def 1).

Def 14 *Argument*

u **is an argument of** v iff

[SYNSEM]u is an element of [SYNSEM LOC CAT VAL COMPS]v.

Def 15 *Native Argument*

u **is a native argument of** v iff

1. u is an argument of v and
2. there is no v' which is a verb and an argument of v such that u is an argument of v'.

Using these definitions, we can state a proposition which links the properties of the COMPS list of a verb to those of the position of this verb within the EMBED value. As the LP principles are implemented as referring to the PHON and the EMBED attribute of an *lp-hc*, this relation guarantees that the LP principles work correctly. The proposition has two parts, one for each LP principle.

PROPOSITION
Let [PHON]u be some permutation of the PHON values of the elements on [BASE]u.
For each u, S(u) = *lp-hc*, if [PHON]u is some permutation of the PHON values of the elements on [WS2 BASE]u then it holds that
for each v which is the value of [EMBED {…DEEPER …} VERB]u at some level of recursion, it is the case that

1. if there is a w, such that w is a pronoun and [SYNSEM]w is the subject of v,

   then for each w', such that w' is a pronoun and a native argument of v:
   if $w' \neq w$,
   then [PHON]w precedes [PHON]w' on [PHON]u.

2. for each pair p, q such that p and q are pronouns and arguments of v it holds that

   if p is a native argument of v and q is a not a native argument of v,

   then [PHON]p precedes [PHON]q on [PHON]u.

The proof relies on the fact that at a given level of embedding ([EMBED {…DEEPER}]u), the elements on the ARGS list of that level of embedding are exactly the native arguments of the VERB value of the same level of embedding. This property follows from different parts of the grammar. These are: Firstly, the assumptions on argument inheritance made in chapter 3, namely that all elements of the COMPS list of a coherently embedded verb appear on the matrix COMPS list. Secondly, the way the ALL-DTRS value is constructed; we have to rely in particular on the fact that all complements and the head are elements of the ALL-DTRS list whose value is given as input to the *ws2* object inside the LP component. Thirdly, the properties of the sort *ws2* as given in section 4.3. In the following we assume this correspondence between the elements on the EMBED structure and the notion '(native) arguments' to be given.

PROOF

For each u with S(u) = *lp-hc*, if [PHON]u is some permutation of the PHON values of the elements on [WS2 BASE]u …

Part 1:

...then [PHON]u = [LPP2 PHON]u and [WS2 EMBEDDING EMBED]u = [EMBED]u = [LPP2 EMBED]u. Thus we know for each verb v which occurs at some level of embedding as [EMBED {... DEEPER ...} VERB]$u = v$ that all its native arguments appear in [EMBED {... DEEPER ...} ARGS]u at the same level of embedding. By [PHON]u = [LPP2 PHON]u and the precondition that the phonologies of all elements on [WS2 BASE]u, it is ensured that the phonology of the arguments of the embed structure is contained in the PHON list of *lpp2* objects. Since S([LPP2]u) $\sqsubseteq$ *lpp2*, we know by the properties of this sort (4.72) that for each o if o is a pronoun, a native argument of v and [SYNSEM]o is the subject of v, then [PHON]o precedes [PHON]o' on [PHON]u for every o' which is a pronominal native argument of v but [SYNSEM]o' is not the subject of v. Thus, the first part of the proposition holds.

Part 2:

...then [PHON]u = [LPP2 PHON]u and [WS2 EMBEDDING EMBED]u = [EMBED]u = [LPP2 EMBED]u. Thus we know for each verb v which occurs at some level of embedding as [EMBED {... DEEPER ...} VERB]$u = v$ that all its native arguments appear in [EMBED {... DEEPER ...} ARGS]u at the same level of embedding.

By [PHON]u = [LPP3 PHON]u and the precondition on [PHON]u it is ensured that the phonology of the arguments of the embed structure is contained in the PHON list of [LPP3]u. Since S([LPP3]u) $\sqsubseteq$*lpp3*, we know that if there are any p and q which are pronominal arguments of v then they are pronouns of [LPP3 EMBED]u. We know further that if p is a native argument of v, then it is a current pronoun of [EMBED {... DEEPER ...}]$u = e$, and any non-native pronominal argument q of v is a deeper pronoun of e. With the properties of *lpp3* we know that [PHON]p precedes [PHON]q on [LPP3 PHON]u, which means that [PHON]p precedes [PHON]q on [PHON]u as well. Thus the second part of the proposition holds.

With all its parts being shown, the proposition follows. ■

With this proof it is shown that *lp-hc* incorporates the two LP principles stated informally in (4.39) and (4.72). Thus there are two relations expressed that must hold between the PHON value and the EMBED value of an *lp-hc* object. In chapter 5 the nesting of an *lp-hc* object as the value of an LP attribute on a *phrase* will be established. At this last step, it will finally hold that the LP principles in (4.39) and (4.72) restrict the PHON value of a *phrase* with respect to the set of all permutations of the PHON values on the *phrase*'s daughters.

Before we turn to the relevant parts of the theory of *phrase*, however, we shall explore some possible extensions of the *lp-hc* object. This is done in section 4.7, where we show how further auxiliary structures can be constructed, and how LP rules proposed in the literature can be encoded as LP principles in the present system.

4.7 Extending the *lp-hc* object

Now that we have precisely defined parts of an *lp-hc* object, it might seem that the formal apparatus needed for the formulation of two LP principles is much larger than one would expect intuitively. In the present section, we show that once a collection of sorts is defined, they can be used to state new LP principles as well. Thus, the main work needs only be done once.

To illustrate this point, we shall extract a set of linearization regularities from the literature and demonstrate how these regularities can be expressed as LP principles by basically reusing the sorts defined above. The LP rules considered in this section are taken from Pollard and Sag (1987), Uszkoreit (1986a), Pollard et al. (1994), Jacobs (1988) and Reape (1994). The selection from the literature has been done arbitrarily and we shall not comment on the empirical value of these LP rules. The purpose of the presentation lies in the aspects of formalization. It will be shown how easily new LP principles can be added to the species *lp-hc*, thus giving some justification for the formal efforts needed in the preceding sections. Furthermore, the examples will demonstrate that the present framework is strong enough to capture any LP rule that has been proposed in the literature. And, as a consequence of this, it should become evident that the grammar writer can stick to her favored set of LP rules and, nevertheless, adopt the LP component outlined in this thesis. By doing so, she steps from mostly only intuitively interpretable LP rules to a system fully integrated into the grammar.

In (4.77), some of the currently assumed precedence regularities for German are given:[48] The LP rules in (4.77) are listed in the order they are discussed in this section. In (4.78) we repeat an LP rule introduced by Pollard and Sag (1987) that makes reference to the relative position of elements on the obliqueness hierarchy. This LP rule, too, will be formalized in the course of the present section.

(4.77) *Some LP rules for German*

(LP-1) An (uninverted) V follows any other V that appears on its COMPS list.

(LP-2) Personal pronouns precede nonpronominal constituents.

(LP-3) The agent precedes the theme.

[48]The precedence rules collected in (4.77) are restricted to the ones that would apply in an *lp-hc* object. Within such an object, however, it is not only the placement of nominal constituents which is to be taken care of, but of verbal elements as well. Thus, the scope of the LP rules in (4.77) goes beyond the domain traditionally called mittelfeld.

(LP-4) The agent precedes the goal.

(LP-5) The goal precedes the theme.

(LP-6) NPs precede (uninverted) V.

(LP-7) An inverted V precedes everything else in the current domain.

(LP-8) scope including elements precede included elements.

(4.78) *An LP rule for English*

(LP-$\ll$) less oblique complements precede more oblique ones.

The first LP rule, together with (LP-6), and (LP-7), has to do with verb placement. (LP-1) accounts for the order of verbs in a verbal complex, where higher verbs are supposed to follow deeper verbs. (LP-6) orders all verbs which are part of the verbal complex, namely the uninverted verbs, after the nominal elements. By means of this LP rule, the verbal complex appears at the end of a sentence. (LP-7), finally, is used to treat fronted verbs. These verbs bear a INV + specification and their phonology is realized before the phonologies of the other nominal and verbal elements. As they stand, (LP-1), (LP-6) and (LP-7) are certainly in no way empirically adequate[49]. However, they are extracted from Uszkoreit (1986a, p. 3, number (3)), Reape (1994, p. 315, his (1) - (3) respectively) and Pollard et al. (1994, p. 5, number (2)).

The rules (LP-3), (LP-4), (LP-5) and (LP-2) deal with the relative order of nominal constituents in the mittelfeld. The first three of them make reference to thematic roles, the last one is sensitive to the species of the CONTENT value of the NP, pronominal or not. (LP-3) – (LP-5) are cited from Uszkoreit (1986a, p. 9, his (OP1) - (OP3)). Variations on these rules can be found in Jacobs (1988, p. 19, his (P1) - (P2), and (P7) on p. 23). (LP-2) is also taken from Uszkoreit (1986a, p. 10, his (OP5)). Similar statements can be found in Jacobs (1988, p. 20, his (P3)) and Frank (n.d., p. 6).

The last LP rule in (4.77), (LP-8), is a rephrasing of a linearization principle given by Jacobs (1988, p. 20, his (P6)), but not mentioned in the other papers. According to this LP rule, the order of scopal elements in the mittelfeld is governed by the scope inclusion relation that holds between them.

[49]Hinrichs and Nakazawa (1994) give a system which is able to capture more verb placement phenomena than the system in (4.77). They achieve a broader empirical coverage by introducing a new boolean feature FLIP (Hinrichs and Nakazawa 1994, pp. 23 - 24). Reape (1990) uses a feature DIR(ection of status government) for similar purposes.

In (4.78) an LP rule from (Pollard and Sag 1987) is repeated.[50] Some aspects of this LP rule have already been discussed in section 4.1. Pollard and Sag (1987) use this LP rule to determine the relative order of complements in English.

This brief review of LP rules commonly assumed served to show the kind of regularities stated in the literature. It will next be sketched how these rules can be reconstructed within the present conception of an LP component for HPSG. Section 4.7 is further divided in three parts. In 4.7.1 formalizations of the LP rules (LP-1) – (LP-5) are sketched. In section 4.7.2 we show how to construct another auxiliary structure, viz. the value of BASE. Besides this, the LP rules (LP-6), (LP-7) and (LP-$\ll$) are considered. Section 4.7.3 contains some remarks on what a third auxiliary structure could look like that would be needed to encode an LP principle corresponding to (LP-8). The last section, 4.7.4, resumes the main results of the possible extensions of the sort *lp-hc* discussed in 4.7.

Even though the formalization of LP principles is central in the following sections, the discussion will not be as technical as it has been in the preceding parts of this thesis. As in most cases the sorts introduced in this section, 4.7, make use of sorts already defined elsewhere in this thesis, no further proofs are given.

4.7.1 Some LP principles using the EMBED structure

The first LP rule from (4.77) we want to take a closer look at, is (LP-1). It is repeated in (4.79) for convenience.

(4.79) *(LP-1)*

1. An (uninverted) V follows any other V that appears on its COMPS list.
2. Example sentence:
   ... weil Peter Maria das Lied singen hören will
   because Peter Maria the song sing hear wants
   '... because Peter wants to hear Maria sing the song'

This LP rule is responsible for the presumably canonical order of verbs in German coherent constructions, exemplified in sentence (4.79-2). An informal rephrasing of (LP-1) is very similar in structure to the formulation in (4.73) of the LP principle encoded into an *lp-hc*'s LPP3 attribute.

[50]See example (4.3) for a quote from (Pollard and Sag 1987, p. 174).

(4.80) *Informal statement of (LP-1)*

> if a *phrase* x has daughters i and j such that
>
> i and j are verbs and
> i is uninverted and
> i is less deeply embedded than j
>
> then i's PHON follows j's PHON on x's PHON.

The way (LP-1) is expressed in (4.80) indicates already how we intend to formalize it. There will be an extra attribute appropriate for an *lp-hc* object. We shall call it LPP-VERBS with a value of sort *lpp-verbs*. The value of the LPP-VERBS attribute encodes an LP principle which takes the PHON list and the EMBED structure from other parts of the *lp-hc* object. Precedence checking proceeds along the EMBED structure, just as has been the case within the value of LPP3. At any level of embedding it is checked whether there is an uninverted highest verb and all the deeper verbs are collected. Then it is verified that the phonologies of the deeper verbs precede the phonology of the highest one in the overall phonology.

The shape of an *lpp-verbs* object is defined by the appropriateness conditions given in (4.81)

(4.81) *Appropriateness Conditions for the sort lpp-verbs*

lpp-verbs	PHON	*list-of-phons*
	EMBED	*embed*
	CUR-VERB	*signphon*
	DEEP-VERBS	*signphon*
	FIND-CUR	*list-of-signs*

lpp-verbs0	PHON	*list-of-phons*
	EMBED	*embed0*
	CUR-VERB	*signphon0*
	DEEP-VERBS	*signphon0*
	FIND-CUR	*elist*
lpp-verbs1	PHON	*list-of-phons*
	EMBED	*embed1*
	CUR-VERB	*signphon0*
	DEEP-VERBS	*signphon*
	FIND-CUR	*list-of-signs*
	FIND-DEEP	*append-of-signs*
	REC	*lpp-verbs*

lpp-verbs2	PHON	*list-of-phons*
	EMBED	*embed1*
	CUR-VERB	*signphon1*
	DEEP-VERBS	*signphon*
	FIND-CUR	*list-of-signs*
	FIND-DEEP	*append-of-signs*
	TEST	*prec*
	REC	*lpp-verbs*

A look at the appropriateness conditions of *lpp3* given in (4.74) shows the striking parallels between the two sorts. The main difference between the two sorts lies in the fact that there are just two cases distinguished in *lpp3*, but three cases in *lpp-verbs*: there may be no VERB attribute on the EMBED value, or the EMBED VERB value is an inverted verb or the EMBED VERB value is an uninverted verb.

In (4.82) the theories of *lpp-verbs1* and *lpp-verbs2* are given as AVMs. There will be no formal statement or proof, as the parallelism to *lpp3* is evident. Nevertheless we shall briefly comment on parts of the theories of *lpp-verbs1* and *lpp-verbs2*.

(4.82) *Theory of lpp-verbs (as AVMs)*

Theory of *lpp-verbs1* as AVM:

$$\begin{bmatrix} \textit{lpp-verbs1} \end{bmatrix} \Longrightarrow$$

$$\begin{bmatrix}
\textit{lpp-verbs1} & \\
\text{PHON} & \boxed{1}\ \textit{list-of-phons} \\
\text{EMBED} & \begin{bmatrix} \textit{embed1} & \\ \text{VERB} & \boxed{2} \begin{bmatrix} \textit{sign} \\ \text{SYNSEM LOC CAT HEAD INV} + \end{bmatrix} \\ \text{ARGS} & \textit{list-of-signs} \\ \text{DEEPER} & \boxed{3}\ \textit{embed} \end{bmatrix} \\
\text{CUR-VERB} & \textit{signphon0} \\
\text{DEEP-VERBS} & \begin{bmatrix} \textit{signphon} & \\ \text{PHONS} & \textit{list-of-phons} \\ \text{SIGNS} & \boxed{4}\ \textit{list-of-signs} \end{bmatrix} \\
\text{FIND-CUR} & < \boxed{2}\ \textit{sign} > \\
\text{FIND-DEEP} & \begin{bmatrix} \textit{append-of-signs} & \\ \text{LEFT} & \boxed{5}\ \textit{list-of-signs} \\ \text{RIGHT} & \boxed{6}\ \textit{list-of-signs} \\ \text{RES} & \boxed{4}\ \textit{list-of-signs} \end{bmatrix} \\
\text{REC} & \begin{bmatrix} \textit{lpp-verbs} & \\ \text{PHON} & \boxed{1}\ \textit{list-of-phons} \\ \text{EMBED} & \boxed{3}\ \textit{embed} \\ \text{FIND-CUR} & \boxed{6}\ \textit{list-of-signs} \\ \text{FIND-DEEP RES} & \boxed{5}\ \textit{list-of-signs} \end{bmatrix}
\end{bmatrix}$$

The theory of *lpp-verbs2* as AVM:

$$
\left[\textit{lpp-verbs2}\right] \Longrightarrow
$$

$$
\left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{lpp-verbs2}} \\
\text{PHON} & \boxed{1}\ \textit{list-of-phons} \\
\text{EMBED} & \left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{embed1}} \\
\text{VERB} & \boxed{2} \left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{sign}} \\
\text{PHON} & \boxed{2"}\ \textit{list-of-phons} \\
\multicolumn{2}{l}{\text{SYNSEM LOC CAT HEAD INV -}}
\end{array}\right] \\
\text{ARGS} & \textit{list-of-signs} \\
\text{DEEPER} & \boxed{3}\ \textit{embed}
\end{array}\right] \\
\text{CUR-VERB} & \left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{signphon1}} \\
\text{PHONS} & \boxed{2"}\ \textit{list-of-phons} \\
\text{SIGNS} & \boxed{2}\ \textit{list-of-signs}
\end{array}\right] \\
\text{DEEP-VERBS} & \left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{signphon}} \\
\text{PHONS} & \boxed{4"}\ \textit{list-of-phons} \\
\text{SIGNS} & \boxed{4}\ \textit{list-of-signs}
\end{array}\right] \\
\text{FIND-CUR} & <\ \boxed{2}\ \textit{sign}\ > \\
\text{FIND-DEEP} & \left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{append-of-signs}} \\
\text{LEFT} & \boxed{5}\ \textit{list-of-signs} \\
\text{RIGHT} & \boxed{6}\ \textit{list-of-signs} \\
\text{RES} & \boxed{4}\ \textit{list-of-signs}
\end{array}\right] \\
\text{TEST} & \left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{prec}} \\
\text{C} & \boxed{1}\ \textit{list-of-phons} \\
\text{A} & \boxed{4"}\ \textit{list-of-phons} \\
\text{B} & \boxed{2"}\ \textit{list-of-phons}
\end{array}\right] \\
\text{REC} & \left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{lpp-verbs}} \\
\text{PHON} & \boxed{1}\ \textit{list-of-phons} \\
\text{EMBED} & \boxed{3}\ \textit{embed} \\
\text{FIND-CUR} & \boxed{6}\ \textit{list-of-signs} \\
\text{FIND-DEEP RES} & \boxed{5}\ \textit{list-of-signs}
\end{array}\right]
\end{array}\right]
$$

To explain the AVMs in (4.82), we give some remarks on the theories of the species of *lpp-verbs*. First, if there is no current verb, i.e. we have reached the deepest level of embedding, then nothing has to be done and the recursive precedence checking has come to an end. Thus the object is of species *lpp-verbs0* and no further constraint has to be stated for this species.

Second, there is a current verb, but it is inverted (INV $+$). Then nothing has to be checked either. In order to guarantee a correct collecting of the deeper verbs, the current verb is put as first and only member on the FIND-CUR list. Furthermore, the list containing the former current verb (REC FIND-CUR) is appended to the list of formerly deeper verbs (REC

FIND-DEEP RES) to represent the new deeper verbs (FIND-DEEP RES and DEEP-VERBS SIGNS).

In the third case, the current verb is uninverted. Then the formerly current verb is appended to the formerly deeper verbs the way just described, and, again, the current verb is put as only element on the FIND-CUR list. Furthermore it is now verified that the current verb's phonology follows the phonologies of the deeper verbs on the PHON list.

With these remarks on (LP-1) a first example has been given of how an often stated LP rule can be incorporated into the LP component presented in this thesis. There is just one additional attribute needed on the *lp-hc* object, namely LPP-VERBS, with one additional sort definition for its value. The rest of the work is taken care of by sorts which have already been defined, as *append-of-signs* ((2.9) and (2.10)), *signphon* ((2.8), (2.11)) and *prec* ((4.53) and (4.65)). Furthermore, precedence checking proceeded along the auxiliary structure of sort *embed*.

The next linearization regularities to be treated are (LP-3), (LP-4) and (LP-5) of (4.77). These LP rules are concerned with the relative order of NPs with different thematic roles. In (4.83), they are repeated.

(4.83) *Three LP rules*

(LP-3) The agent precedes the theme

(LP-4) The agent precedes the goal

(LP-5) The goal precedes the theme

The three LP rules in (4.83) are too general as they stand. In the German mittelfeld, they only apply to 'full' NPs, i.e.NPs whose CONTENT value is of sort *nonpronoun*. Application should further be restricted to NPs with the same value for definiteness and focus.[51] For the sake of simplicity in the following treatment of (LP-3) – (LP-5), the restriction to full NPs will be the only one considered.

The three LP rules in (4.83) refer to thematic roles like *agent*, *theme* and *goal*. These roles are, however, unknown in HPSG as it stands, where there are only individual roles in the sense of Dowty (1989), i.e. a verb like *give* has as attributes GIVER, GIVEE and GIVEN in its NUCLEUS value

[51]Although we do not commit ourselves to any judgment about the empirical adequacy of the LP rules in (4.77), we still have to take care of some of the interactions between them. This is done here by restricting the application of precedence checking to elements with rather complex properties. In the system outlined in Uszkoreit (1986a) possible interactions between linearization regularities are handled via disjunctive LP rules. Even though we do not intend to use disjunctive LP principles, we shall show at the end of section 4.7.2 how these can be incorporated into the present framework.

instead of AGENT, GOAL and THEME. But, as shown in Dowty (1989), it is far from clear whether there a direct mapping from individual thematic roles to some standardized set of thematic roles can be defined. Let us assume for the time being that such a mapping has been established, or that the individual thematic role labels like GIVER have been replaced by labels like AGENT etc.[52]

With the treatment of thematic roles so modified it is possible to identify a sign's thematic role by looking at the structure of embedding. In (4.84-1) an example sentence is given together with the EMBED value in (4.84-2) that appears in the LP value for that sentence.

(4.84) *Thematic roles in the* EMBED *structure*

1. *An example:*
   (...daß) Peter Maria das Buch gegeben hat
   (...that) Peter Maria the book given has
   '...that Peter(agent) has given Maria(goal) a book(theme)'

2. *The* LP EMBED *value for the corresponding phrase:*

$$\left[\begin{array}{ll} \multicolumn{2}{l}{\textit{embed1}} \\ \text{VERB} & \left[\begin{array}{ll} \text{PHON} & \langle \textit{hat} \rangle \\ \text{SYNSEM LOC CONTENT NUCLEUS} & \left[\begin{array}{l} \textit{haben} \\ \text{SOA-ARG}\ \boxed{4}\ \textit{psoa} \end{array}\right] \end{array}\right] \\ \text{ARGS} & \langle \boxed{4'} \rangle \\ \text{DEEPER} & \left[\begin{array}{ll} \multicolumn{2}{l}{\textit{embed1}} \\ \text{VERB}\ \boxed{4'} & \left[\begin{array}{ll} \text{PHON} & \langle \textit{gegeben} \rangle \\ \text{SYNSEM LOC CONTENT}\ \boxed{4} & \left[\begin{array}{ll} \multicolumn{2}{l}{\textit{psoa}} \\ \text{NUCLEUS} & \left[\begin{array}{l} \textit{give} \\ \text{AGENT}\ \boxed{1}\ \textit{index} \\ \text{GOAL}\ \boxed{2}\ \textit{index} \\ \text{THEME}\ \boxed{3}\ \textit{index} \end{array}\right] \end{array}\right] \end{array}\right] \\ \text{ARGS} & \langle \boxed{1'}, \boxed{2'}, \boxed{3'} \rangle \end{array}\right] \\ \multicolumn{2}{l}{\text{DEEPER}\ \textit{embed0}} \end{array}\right]$$

where:

$$\boxed{1'} = \left[\begin{array}{ll} \text{PHON} & \langle \textit{Peter} \rangle \\ \text{SYNSEM LOCAL CONTENT} & \left[\begin{array}{l} \textit{npro} \\ \text{INDEX}\ \boxed{1} \end{array}\right] \end{array}\right]$$

[52] The fact that Pollard and Sag (1994) do not use general thematic role attributes could be an indication that the authors do not expect empirical generalizations as the ones in (4.83) to be relevant for linguistic theory.

$$\boxed{2'} = \begin{bmatrix} \text{PHON} & \textit{<Maria>} \\ \text{SYNSEM LOCAL CONTENT} & \begin{bmatrix} npro \\ \text{INDEX}\ \boxed{2} \end{bmatrix} \end{bmatrix}$$

$$\boxed{3'} = \begin{bmatrix} \text{PHON} & \textit{<das, Buch>} \\ \text{SYNSEM LOCAL CONTENT} & \begin{bmatrix} npro \\ \text{INDEX}\ \boxed{3} \end{bmatrix} \end{bmatrix}$$

In (4.84-2) the indices of the elements in the DEEPER ARGS list and those that appear as values on the thematic role attributes of the DEEPER VERB are identical. By the way the LP EMBED value of a phrase is built in LP WS2 (cf. section 4.3), it is guaranteed that there always is structure sharing between the INDEX values of the ARGS list at any level of embedding with some thematic role attribute values of the VERB sign at the same level of embedding. Thus, even without an attribute THEMATIC-ROLE on complements, it is possible to find the thematic role a complement is assigned by looking at the EMBED structure.

The relation between the thematic role attributes in the NUCLEUS value of a verb and the properties of the EMBED structure can be used to state the LP rules (LP-3) – (LP-5) from (4.83) as LP principles. The informal formulation of the three LP rules in (4.85) resembles the structure of our first LP given as (4.39):

(4.85) *Informal statement of (LP-3) - (LP-5)*

(LP-3) If a *phrase* x has signs v, i and j as daughters such that
v, i and j are of the same level of embedding and
v is a verb and
i and j are full NPs and
i's INDEX and v's AGENT values are identical and
j's INDEX and v's THEME values are identical, then

i's PHON precedes j's PHON on x's PHON.

(LP-4) If a *phrase* x has signs v, i and j as daughters such that
v, i and j are of the same level of embedding and
v is a verb and
i and j are full NPs and
i's INDEX and v's AGENT values are identical and
j's INDEX and v's GOAL values are identical, then

i's PHON precedes j's PHON on x's PHON.

(LP-5) If a *phrase* x has signs v, i and j as daughters such that
v, i and j are of the same level of embedding and
v is a verb and

i and j are full NPs and
i's INDEX and v's GOAL values are identical and
j's INDEX and v's THEME values are identical, then
i's PHON precedes j's PHON on x's PHON.

The LP principles in (4.85) are formulated parallel to (4.39). This indicates that they too make use of the auxiliary structure of embedding and check precedence restrictions among elements of the same level of embedding. The three principles can be encoded into three distinct attributes of an *lp-hc*, LPP-AT for (LP-3), LPP-AG for (LP-4) and LPP-GT for (LP-5) whose values are of sort *lpp-at*, *lpp-ag* and *lpp-tg* respectively. As the three LP principles differ only with respect to the thematic roles involved, it is sufficient to look at one of them, (LP-3), more closely.

The sort *lpp-at* is defined exactly like the sort *lpp2* ((4.40) and (4.50)) except for the attribute FIND-S/NS. In its stead, there is an attribute FIND-AG/TH. Thus, there are two species of the sort *lpp-at*, *lpp-at0* and *lpp-at1*. Both species are specified for the attributes PHON and EMBED. Furthermore, *lpp-at1* has the attributes TEST with values of sort *prec*, REC with values of sort *lpp-at* and an attribute FIND-AG/TH.

The value of the attribute FIND-AG/TH is of a new sort *find-ag/th*. In (4.86) the attributes of this sort are given:

(4.86) *Appropriateness Conditions for the sort find-ag/th*

find-ag/th	VERB	*sign*
	ARGS	*list-of-signs*
	AGENT	*signphon*
	THEME	*signphon*
	FIND-NPRO	*find-npro*
	SEP-NPRO	*separate-npro*

Even though some of the attributes of *find-ag/th* are new, their function should already be intuitively clear. Within the value of FIND-NPRO the nonpronominal nominal elements of the ARGS list, i.e. the full NPs, are identified. In SEP-NPRO the agent and the theme of the VERB are selected if they are among the nonpronominals. Their value is guaranteed to be token identical to the AGENT SIGNS and THEME SIGNS specification respectively. Thus, the new sort *find-ag/th* is very much like *find-s/ns*.

Let us have a brief look at the two new sorts used in (4.86) and compare them to the ones needed in the definition of *find-s/ns*. The sort *find-npro* that appears as value of the FIND-NPRO attribute in (4.86) differs from *find-pro* in (4.43) and (4.44) with respect to the elements which are selected from the ARGS list. In the case of *find-npro* the signs with a specification of SYNSEM LOC CONTENT of sort *nonpronoun* are found on a NPROS list. In the definition of the sort *find-pro* the elements with a SYNSEM LOC CONTENT value of sort *pron* were looked for.

Things are slightly different for the value of the attribute SEP-NPRO of a *find-ag/th* object as compared to the SEP-PRONS value in the case of an *find-s/ns* object. Again, a search function is encoded, but now there are four cases to be considered in *sep-npro* instead of the three in case of *separate-pron*.[53] To make the following discussion more concrete, the appropriateness conditions of the sort *separate-npro* are given in (4.87).

(4.87) *Appropriateness Conditions for the sort separate-npro*

separate-npro	NPROS	*list-of-signs*	
	VERB	*sign*	
	AGENT	*list-of-signs*	
	THEME	*list-of-signs*	
	separate-npro0	NPROS	*elist*
		VERB	*sign*
		AGENT	*elist*
		THEME	*elist*
	separate-npro1	NPROS	*nelist-of-signs*
		VERB	*sign*
		AGENT	*list-of-signs*
		THEME	*list-of-signs*
		REC	*separate-npro*
	separate-npro2	NPROS	*nelist-of-signs*
		VERB	*sign*
		AGENT	*nelist-of-signs*
		THEME	*list-of-signs*
		REC	*separate-npro*
	separate-npro3	NPROS	*nelist-of-signs*
		VERB	*sign*
		AGENT	*list-of-signs*
		THEME	*nelist-of-signs*
		REC	*separate-npro*

The four species of *separate-npro* treat one of the following cases each. In the first case, there are no (more) elements on the NPRO list to be considered. In this case, the end of the recursion is reached (*separate-npro0*). In the second case the NPRO FIRST element is neither the agent nor the theme of the verb. Thus it is simply ignored and the next element on the NPRO list is considered (*separate-npro1*). The third case is relevant, if the ARGS FIRST element is the verb's agent which is put on the AGENT list (*separate-npro2*). In the fourth and last case, the NPRO FIRST element is the verb's theme. It is put on the THEME list. These four cases are captured in the theory of *separate-npro* which is sketched by the following AVMs.

[53] cf. the definitions in (4.47) and (4.48) in section 4.4.

(4.88) *Theory of separate-npro in AVMs (partially):*

Theory of *separate-npron1*

$$\left[\textit{separate-npron1}\right] \Longrightarrow$$

$$\left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{separate-npron1}} \\
\text{NPROS} & \left\langle \left[\text{SYNSEM LOC CONTENT INDEX } \boxed{1}\right] \middle| \boxed{2} \right\rangle \\
\text{VERB} & \boxed{3}\left[\text{SYNSEM LOC CONTENT NUCLEUS} \left[\begin{array}{ll} \text{AGENT} & \boxed{4} \\ \text{THEME} & \boxed{5} \end{array}\right]\right] \\
\text{AGENT} & \boxed{6} \\
\text{THEME} & \boxed{7} \\
\text{REC} & \left[\begin{array}{ll} \multicolumn{2}{l}{\textit{separate-npro}} \\ \text{NPROS} & \boxed{2} \\ \text{VERB} & \boxed{3} \\ \text{AGENT} & \boxed{6} \\ \text{THEME} & \boxed{7} \end{array}\right]
\end{array}\right]$$

$\& \; (\boxed{1} \neq \boxed{4}) \; \& \; (\boxed{1} \neq \boxed{5})$

Theory of *separate-npro2*

$$\left[\textit{separate-npron2}\right] \Longrightarrow$$

$$\left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{separate-npron2}} \\
\text{NPROS} & \left\langle \boxed{1}\left[\text{SYNSEM} \ldots \text{INDEX } \boxed{2}\right] \middle| \boxed{3} \right\rangle \\
\text{VERB} & \boxed{4}\left[\text{SYNSEM} \ldots \text{NUCLEUS} \left[\begin{array}{ll} \text{AGENT} & \boxed{2} \\ \text{THEME} & \textit{index} \end{array}\right]\right] \\
\text{AGENT} & < \boxed{1} > \\
\text{THEME} & \boxed{6} \\
\text{REC} & \left[\begin{array}{ll} \multicolumn{2}{l}{\textit{separate-npro}} \\ \text{NPROS} & \boxed{3} \\ \text{VERB} & \boxed{4} \\ \text{AGENT} & <> \\ \text{THEME} & \boxed{6} \end{array}\right]
\end{array}\right]$$

The AVM corresponding to the theory of *separate-npro3* would look similar to the one for *separate-npro2*, but with the values for AGENT and THEME interchanged.

We have seen that the sorts *find-npro* and *separate-npro* are very similar to the sorts *find-pron* and *separate-pron*. This similarity carries over to the theory of the sort *find-ag/th* as compared to the theory of the sort *find-s/ns*. In (4.89) the theories of both sorts are given as AVMs. The AVM for the theory of *find-s/ns* is repeated from (4.50) of section 4.4.2.

(4.89) *Comparing two theories*

Theory of *find-ag/th*

$$\begin{bmatrix}\textit{find-ag/th}\end{bmatrix} \Longrightarrow \begin{bmatrix} \textit{find-ag/th} & \\ \text{VERB} & \boxed{3} \\ \text{ARGS} & \boxed{1} \\ \text{AGENT} & \begin{bmatrix}\textit{signphon} \\ \text{SIGNS}\ \boxed{4}\end{bmatrix} \\ \text{THEME} & \begin{bmatrix}\textit{signphon} \\ \text{SIGNS}\ \boxed{5}\end{bmatrix} \\ \text{FIND-NPRO} & \begin{bmatrix}\textit{find-npro} & \\ \text{ARGS} & \boxed{1} \\ \text{NPROS} & \boxed{2}\end{bmatrix} \\ \text{SEP-NPRO} & \begin{bmatrix}\textit{separate-npro} & \\ \text{NPROS} & \boxed{2} \\ \text{VERB} & \boxed{3} \\ \text{AGENT} & \boxed{4} \\ \text{THEME} & \boxed{5}\end{bmatrix} \end{bmatrix}$$

Theory of *find-s/ns*

$$\begin{bmatrix}\textit{find-s/ns}\end{bmatrix} \Longrightarrow \begin{bmatrix} \textit{find-s/ns} & \\ \text{VERB} & \boxed{3} \\ \text{ARGS} & \boxed{1} \\ \text{SU} & \begin{bmatrix}\textit{signphon} \\ \text{SIGNS}\ \boxed{4}\end{bmatrix} \\ \text{NSU} & \begin{bmatrix}\textit{signphon} \\ \text{SIGNS}\ \boxed{5}\end{bmatrix} \\ \text{FIND-PRON} & \begin{bmatrix}\textit{find-pron} & \\ \text{ARGS} & \boxed{1} \\ \text{NPROS} & \boxed{2}\end{bmatrix} \\ \text{SEP-PRON} & \begin{bmatrix}\textit{separate-pron} & \\ \text{PRONS} & \boxed{2} \\ \text{VERB} & \boxed{3} \\ \text{SU} & \boxed{4} \\ \text{NSU} & \boxed{5}\end{bmatrix} \end{bmatrix}$$

To conclude the discussion of (LP-3), it just remains to give an overview of an entire *lpp-at* object This general perspective is given in the AVM in (4.90), using the example from (4.84-1).

(4.90) *An example lpp-at (considering the second level of embedding only)*

$$
\begin{bmatrix}
\textit{lpp-at1} & \\
\text{PHON} & \boxed{1} \langle \textit{Peter, Maria, das, Buch, gegeben, hat} \rangle \\
\text{EMBED} & \begin{bmatrix}
\textit{embed1} & \\
\text{VERB} & \boxed{2} \begin{bmatrix}
\text{PHON} & \langle \textit{gegeben} \rangle \\
\text{SYNSEM ...NUCLEUS} & \begin{bmatrix} \textit{give} & \\ \text{AGENT} & \boxed{3} \\ \text{GOAL} & \boxed{4} \\ \text{THEME} & \boxed{5} \end{bmatrix}
\end{bmatrix} \\
\text{ARGS} & \langle \boxed{6}, \boxed{7}, \boxed{8} \rangle \\
\text{DEEPER} & \textit{embed0}
\end{bmatrix} \\
\text{FIND-AG/TH} & \begin{bmatrix}
\textit{find-ag/th} & \\
\text{VERB} & \boxed{2} \\
\text{ARGS} & \langle \boxed{6}, \boxed{7}, \boxed{8} \rangle \\
\text{AGENT} & \begin{bmatrix} \textit{signphon} & \\ \text{SIGNS} & \langle \boxed{6} \rangle \\ \text{PHONS} & \boxed{6''} \end{bmatrix} \\
\text{THEME} & \begin{bmatrix} \textit{signphon} & \\ \text{SIGNS} & \langle \boxed{8} \rangle \\ \text{PHONS} & \boxed{8''} \end{bmatrix} \\
\text{FIND-NPRO} & \begin{bmatrix} \textit{find-npro} & \\ \text{ARGS} & \langle \boxed{6}, \boxed{7}, \boxed{8} \rangle \\ \text{NPROS} & \langle \boxed{6}, \boxed{7}, \boxed{8} \rangle \end{bmatrix} \\
\text{SEP-NPRO} & \begin{bmatrix} \textit{separate-npro} & \\ \text{NPROS} & \langle \boxed{6}, \boxed{7}, \boxed{8} \rangle \\ \text{VERB} & \boxed{2} \\ \text{AGENT} & \langle \boxed{6} \rangle \\ \text{THEME} & \langle \boxed{8} \rangle \end{bmatrix}
\end{bmatrix} \\
\text{TEST} & \begin{bmatrix} \textit{prec} & \\ \text{C} & \boxed{1} \\ \text{A} & \boxed{6''} \\ \text{B} & \boxed{8''} \end{bmatrix} \\
\text{REC} & \begin{bmatrix} \textit{lpp-at0} & \\ \text{PHON} & \boxed{1} \\ \text{EMBED} & \textit{embed0} \end{bmatrix}
\end{bmatrix}
$$

where:

$$
\boxed{6} = \begin{bmatrix} \text{PHON} & \boxed{6''} \langle \textit{Peter} \rangle \\ \text{SYNSEM LOCAL CONTENT INDEX} & \boxed{3} \end{bmatrix}
$$

$$
\boxed{7} = \begin{bmatrix} \text{PHON} & \langle \textit{Maria} \rangle \\ \text{SYNSEM LOCAL CONTENT INDEX} & \boxed{4} \end{bmatrix}
$$

$$\boxed{8} = \begin{bmatrix} \textsc{phon} & \boxed{8”} \langle \textit{das, Buch} \rangle \\ \textsc{synsem local content index} & \boxed{5} \end{bmatrix}$$

Example (4.90) illustrates all details we have discussed about the sort *lpp-at*, and thus about the linearization principle stated in (4.85) and in (4.83) as (LP-3). The parallel rules, (LP-4) and (LP-5), would be formalized almost the same way, just changing the names of the thematic roles.

For our formal reconstruction of the LP rules (LP-3) – (LP-5) of (4.77), we had to make some assumption about a possible treatment of thematic roles within HPSG. Without these or similar assumptions, however, it would make no sense to assign these LP rules any relevance within an HPSG theory for German.

The preceding discussion of (LP-1), (LP-3), (LP-4) and (LP-5) treated all LP rules from (4.77) that could be stated with respect to the structure of embedding as it is made available by an *lp-hc*'s WS2 attribute. In section 4.1.2 it has been put forward that there are at least two other auxiliary structures needed for an empirically adequate treatment of precedence regularities. One of these structures is encoded in the BASE list of an *lp-hc* object (4.11). This list orders the complement daughters according to their position on the obliqueness hierarchy. The other auxiliary structure can be found in the SCOPOI attribute of an *lp-hc* object. The value of this attribute gives the scope inclusion relations.

In sections 4.7.2 and 4.7.3 we will sketch how these two auxiliary structures can be constructed from other information available, and show briefly that the remaining LP rules from (4.77) can be expressed by making reference to these structures. We shall first address the construction of the BASE list.

4.7.2 Some LP principles involving the BASE structure

The BASE value of an *lp-hc* object is intended to be a list of *signs* which respects the obliqueness hierarchy in its ordering. In the particular case we are concerned with here, only the head daughter and complement daughters are involved. As we assume that the subject is always present on the COMPS list (cf. chapter 3), the order on the head daughter's COMPS list contains all the information about the obliqueness relation among the complement daughters. There are just two complications left: First, there is no place for the head on the obliqueness hierarchy. Second, following Pollard and Sag (1994) there are just *synsem* objects on the COMPS and the SUBCAT lists. But, as a PHON value is needed for checking precedence, one needs to find the *sign* to any of the *synsem* objects on the COMPS list.

As far as the first point is concerned, there are basically two possibilities. The head could be assigned a particular position on the BASE list, e.g. the first. The second option would be to simply allow the head to occur on any position on the BASE list. The first possibility has the advantage that there is always exactly one BASE list ordering. For this practical reason, we adopt the first solution.

We shall next consider the problem of how to get from the *synsem* objects on the COMPS list to *signs*. The technique used in the case of *build-phon* in chapter 2 offers a straightforward solution. In chapter 2 we had to establish a relation between a list of *signs* and a permutation of the PHON values of these *signs*. For that purpose, the value of the ALL-DTRS list was checked to be a permutation of some other list of *signs*. Furthermore, it was guaranteed that the ordering of the *signs* on this list be parallel to the ordering of the elements of the PHON list.

The task of building the BASE list is practically the same, except that the COMPS list plays the role of the PHON list in chapter 2. Again, some permutation of the ALL-DTRS list is needed. But now this permutation is required to have its elements' SYNSEM values ordered the same way these *synsem* objects occur on the head's COMPS list. This checking has to ignore the head daughter, however.

In the informal presentation of the *lp-hc* object in section 4.2, example (4.11), the attribute WS1 was introduced as the external interface attribute used to insure the proper relation between the ALL-DTRS list and the BASE list. Making use of sorts already defined in this thesis, we can now draw a more concrete picture of the WS1 attribute, which takes as value objects of sort *ws1*. In (4.91) the appropriateness conditions of this sort are given, followed by a sketch of the corresponding theory in the AVM in (4.92).

(4.91) *Appropriateness Conditions for the sort ws1*

ws1	ALL-DTRS	*list-of-signs*
	PERM	*perm*
	SIGNSYNSEM	*signsynsem*
	BASE	*list-of-signs*

ALL-DTRS and BASE take simple lists of signs as values. The attribute ALL-DTRS corresponds to the INPUT attribute on the sort *build-aux-i* introduced in section 4.1.2, example (4.6). BASE corresponds to the attribute OUTPUT. Thus, the sort *ws1* fits the requirements of a sort that can be used as the value of an external interface attribute.

The sort *perm* has already been defined in section 2.2 under (2.4) and (2.7). A *perm* object has two attributes with values of sort *list-of-signs*, where one list is a permutation of the other. The only new sort used in (4.91) is *signsynsem*. It is, however, just a variation of the sort *compsverb* familiar from section 4.3.1, examples (4.22) and (4.23). The sort *compsverb* takes as attributes a VERBL with value of sort *list-of-signs*, and a COMPSL with value of sort *list-of-lists-of-synsems*. The list encoded in the second attribute consists exactly of the COMPS lists of the elements on the first list. In a similar way, the sort *compsverb* is specified for two attributes, SIGNS and SYNSEMS, which take a *list-of-signs* and a *list-of-synsems* as their respective values. The two lists are constructed in a parallel way. 'Parallel' here means that for any i-th element of the list of *signs* it holds that its SYNSEM value is token identical with the i-th element of the list of *synsems*.[54]

With these remarks in mind, the AVM in (4.92) should be sufficient to understand the way the BASE list is thought to be constructed.

(4.92) *Theory of ws1 (as AVM):*

$$\begin{bmatrix} \textit{ws1} \end{bmatrix} \Longrightarrow$$

$$\begin{bmatrix} \textit{ws1} & \\ \textsc{all-dtrs} & \begin{bmatrix} \textit{list-of-signs} \\ \textsc{first}\ \boxed{1} \begin{bmatrix} \textit{sign} \\ \textsc{synsem loc cat val comps}\ \boxed{3} \end{bmatrix} \\ \textsc{rest}\ \boxed{2}\ \textit{list-of-signs} \end{bmatrix} \\ \textsc{perm} & \begin{bmatrix} \textit{perm} & \\ \textsc{origin} & \boxed{2}\ \textit{list-of-signs} \\ \textsc{permuted} & \boxed{4}\ \textit{list-of-signs} \end{bmatrix} \\ \textsc{signsynsem} & \begin{bmatrix} \textit{signsynsem} & \\ \textsc{signs} & \boxed{4}\ \textit{list-of-signs} \\ \textsc{synsems} & \boxed{3}\ \textit{list-of-synsems} \end{bmatrix} \\ \textsc{base} & \begin{bmatrix} \textit{list-of-signs} & \\ \textsc{first} & \boxed{1}\ \textit{sign} \\ \textsc{rest} & \boxed{4}\ \textit{list-of-signs} \end{bmatrix} \end{bmatrix}$$

The sort *ws1* is closely tied to the structural assumptions made in this thesis. As stated in (3.14) of section 3.1, the head daughter appears as first element on the DTRS ALL-DTRS list. This list is structure shared with the LP WS1 ALL-DTRS list. Thus, the sign with tag $\boxed{1}$ in (4.92) is token identical with the phrase's head daughter. In the present assumptions on the BASE list, the head daughter is the first element of the auxiliary structure. Thus the sign $\boxed{1}$ appears as the BASE FIRST element in the theory of *ws1*. The rest of the ALL-DTRS list, the list with the tag $\boxed{2}$, is taken for permutation. An appropriate permutation of this list, tagged

[54]Another, but slightly more complicated example for a sort which encodes two parallel lists is the sort *signphon* in (2.8) and (2.9).

as $\boxed{4}$, forms the REST value of the BASE list. The permutation chosen is the one that stands in a *signsynsem*-relation to the SUBCAT list of the head. Thus, the ordering on the BASE REST list corresponds to the obliqueness hierarchy.[55]

The preceding discussion has demonstrated that it is possible to make a list of signs available within a *phrase*'s LP value whose elements occur in the order of obliqueness. This list, put into the LP BASE attribute, serves as auxiliary structure for the formulation of several LP principles. Looking at the linearization regularities collected in (4.77), we can now formalize the following LP rules: (LP-6), which orders the phonology of NP complements before the one of (uninverted) Vs, and (LP-7), which enforces an inverted V's phonology to precede the phonology of any other element on the ALL-DTRS list. But the BASE list can be exploited to formulate more complicated LP rules, too. As an example the LP rule (LP-$\ll$) form (4.78) can be expressed. These LP rules can be rephrased as informal LP principles.

(4.93) *Informal statement of (LP-6), (LP-7) and an (LP-$\ll$)*

(LP-6) If a *phrase* x has signs i and j as daughters such that
i is an NP and
j is a V with INV- then
i's PHON precedes j's PHON on x's PHON.

(LP-7) If a *phrase* x has signs i and j as daughters such that
i is a V with INV+ then
i's PHON precedes j's PHON on x's PHON.

(LP-$\ll$) If a *phrase* x has signs i and j as daughters such that
i and j are complement daughters of x and
i's SYNSEM precedes j's SYNSEM on the COMPS list of
x's head daughter[56]
i's PHON precedes j's PHON on x's PHON.

Following our overall strategy, each of the LP principles in (4.93) is encoded in the value of a particular LP principle attribute of the sort *lp-hc*. These attributes are called LPP-NP/V, LPP-INV and LPP-$\ll$ which

[55]Under a different structural assumption the construction of the BASE list may be slightly more complicated than outlined in (4.91) and (4.92). If one adopts a binary structure for German as given in section 3.2, the 'pivot' is not the first element on the ALL-DTRS list, where the 'pivot' is the lowest head in the projection. So, the 'pivot' has to be extracted from the ALL-DTRS value. It can be found as it is the element of the ALL-DTRS list with the longest COMPS list.

[56]In a binary structure (cf. section 3.2), the COMPS list of the 'pivot' contains the relevant information

take values of sort *lpp-np/v*, *lpp-inv* and *lpp-*$\ll$ respectively. The sorts *lpp-np/v* and *lpp-inv* involve just one global precedence checking procedure over the BASE list, i.e. they are 'simple' LP principles in the sense of section 4.1.2. In fact, the two LP principles even belong to the 'restricted simple' class as they only need information from the signs themselves. We could therefore give a formalization of (LP-6) and (LP-7) with respect to the ALL-DTRS list instead of the BASE list. As the BASE structure will be needed later anyways and as it contains all the information available on the ALL-DTRS list, the two LP principles are defined with reference to the BASE list.

In the case of *lpp-np/v* the phonologies of all the NPs occurring on the BASE list and the phonologies of all uninverted Vs of that list have to be collected. This can be achieved in two steps:

First, the noun phrases and the uninverted verbs are collected in different lists. For that purpose, attributes are specified on the sort *lpp-np/v* that take as value a search functional sort. An example of such a sort has been defined as *find-pron* in (4.43) and (4.44). In (4.94) the appropriateness conditions for the sort *lpp-np/v* are given together with an AVM indicating the theory of this sort. Within an *lpp-np/v* object there are two search functional sorts, *find-np* and *find-v*, which occur as the values of the attributes FIND-NP and FIND-V.

The second step to arrive at the phonologies whose precedence relations are to be verified consists of extracting the phonologies of the signs collected in the first step. In section 2.3 the sort *signphon* has been defined in (2.8) and (2.11). Its purpose is to extract the phonologies from a list of signs. This very sort is used in the case of *lpp-np/v*. In (4.94), the values of the attributes NP and V are of sort *signphon*. As can be seen from the AVM given, their PHONS values contain exactly the phonologies of the NPs and Vs collected from the BASE value.

Finally, the precedence relation between the elements on the NP PHONS list and those on the V PHONS list is checked with respect to the overall PHON list, tagged as $\boxed{1}$ in (4.94). Precedence checking is, again, put into an attribute, TEST, with value of sort *prec* as discussed in section 4.4.3.

(4.94) *A sketch of the sort lpp-np/v:*

Appropriateness Conditions for the sort *lpp-np/v* :

lpp-np/v	PHON	*list-of-phons*
	BASE	*list-of-signs*
	FIND-NP	*find-np*
	FIND-V	*find-v*
	NP	*signphon*
	V	*signphon*
	TEST	*prec*

Theory of *lpp-np/v* :

$$
\begin{bmatrix} \textit{lpp-np/v} \end{bmatrix}
\Longrightarrow
\begin{bmatrix}
\textit{lpp-np/v} & \\
\textsc{phon} & \boxed{1}\ \textit{list-of-phons} \\
\textsc{base} & \boxed{2}\ \textit{list-of-signs} \\
\textsc{find-np} & \begin{bmatrix} \textit{find-np} & \\ \textsc{base} & \boxed{2}\ \textit{list-of-signs} \\ \textsc{np} & \boxed{3}\ \textit{list-of-signs} \end{bmatrix} \\
\textsc{find-v} & \begin{bmatrix} \textit{find-v} & \\ \textsc{base} & \boxed{2}\ \textit{list-of-signs} \\ \textsc{v} & \boxed{4}\ \textit{list-of-signs} \end{bmatrix} \\
\textsc{np} & \begin{bmatrix} \textit{signphon} & \\ \textsc{signs} & \boxed{3}\ \textit{list-of-signs} \\ \textsc{phons} & \boxed{5}\ \textit{list-of-phons} \end{bmatrix} \\
\textsc{v} & \begin{bmatrix} \textit{signphon} & \\ \textsc{signs} & \boxed{4}\ \textit{list-of-signs} \\ \textsc{phons} & \boxed{6}\ \textit{list-of-phons} \end{bmatrix} \\
\textsc{test} & \begin{bmatrix} \textit{prec} & \\ \textsc{c} & \boxed{1}\ \textit{list-of-phons} \\ \textsc{a} & \boxed{5}\ \textit{list-of-phons} \\ \textsc{b} & \boxed{6}\ \textit{list-of-phons} \end{bmatrix}
\end{bmatrix}
$$

It is clear that the sort *lpp-inv* would be a variant of *lpp-np/v*, where inverted Vs are considered instead of NPs and everything else instead of uninverted Vs.[57]

The LP rules (LP-6) and (LP-7) have been shown to be expressible as cases of 'simple' LP principles. (LP-$\ll$) from (4.93) is only slightly more complicated. This LP rule involves a recursive precedence checking along the BASE list, as discussed informally in example (4.8) of section 4.1.2, and demonstrated formally for the sorts *lpp2*, *lpp3*. Thus, (LP-$\ll$) is an instance of a 'recursive' LP principle.

There is one problem with the formulation of (LP-$\ll$) by using the BASE list, namely that (LP-$\ll$) is relevant only for complements, not to the head of a construction. The head, however, is the first element of the BASE list. One way out would be to introduce a new auxiliary list, which were identical to the BASE REST list as designed so far. In order to keep the number of auxiliary structures in an *lp* object as low

[57]One might as well use a variant of the sort *separate-pron* of (4.47) and (4.48) in the case of *lpp-inv*. The variant needed would do with a single search procedure. Recall that *separate-pron* takes a list and puts all elements of that list which have a certain property on a second list. The remaining elements of the first list are collected in a third list. In the case of *separate-pron* the three list valued attributes are called PRONS, SU and NSU. For the purpose of *lpp-inv*, the inverted verbs are extracted from the BASE list and put into an INVERTED list. For the remaining elements of the BASE list there is an OTHERS list.

as possible, one could also think of a second approach. The idea is to assure that the phonology of the second not that of the first element of the BASE list precedes the phonology of all following elements on that list. Applied recursively along the BASE list, this trick excludes exactly the first element of the original BASE lists from the precedence checking. For the other elements of the list, the precedence checking procedure just outlined provides exactly the effects of (LP-$\ll$). In this second approach the LP principle can be stated as simply as in the first, so there is no need to introduce a new auxiliary attribute on an *lp-hc* object.

To give a more concrete impression of what (LP-$\ll$) looks like in the present framework, the appropriateness conditions for the sort *lpp-*$\ll$ and its species are given in (4.95), followed by the theory of this sort.

(4.95) *A sketch of the sort lpp-*$\ll$*:*

Appropriateness Conditions for the sort *lpp-*$\ll$:

lpp-$\ll$	PHON	*list-of-phons*	
	BASE	*nelist-of-signs*	
	lpp-$\ll$*0*	PHON	*list-of-phons*
		BASE	*nelist-of-signs*
	lpp-$\ll$*1*	PHON	*list-of-phons*
		BASE	*nelist-of-signs*
		COMP1	*signphon*
		COMP2	*signphon*
		TEST	*prec*
		REC	*lpp-*$\ll$

Theory of *lpp-*$\ll$:

$$\begin{bmatrix} \textit{lpp-}\ll\textit{0} \end{bmatrix} \Longrightarrow \begin{bmatrix} \textit{lpp-}\ll\textit{0} \\ \text{PHON } \textit{list-of-phons} \\ \text{BASE } \begin{bmatrix} \textit{nelist-of-signs} \\ \text{FIRST } \textit{sign} \\ \text{REST } \textit{elist} \end{bmatrix} \end{bmatrix}$$

$$
\left[\textit{lpp-}{\ll}\textit{1}\right] \Longrightarrow
$$

$$
\left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{lpp-}{\ll}\textit{1}} \\
\text{PHON} & \boxed{1}\ \textit{list-of-phons} \\
\text{BASE} & \left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{nelist-of-signs}} \\
\text{FIRST} & \textit{sign} \\
\text{REST} & \boxed{2} \left[\begin{array}{l}
\textit{nelist-of-signs} \\
\text{FIRST}\ \boxed{3}\ \textit{sign} \\
\text{REST}\ \boxed{4}\ \textit{nelist-of-signs}
\end{array}\right]
\end{array}\right] \\
\text{COMP1} & \left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{signphon}} \\
\text{SIGNS} & <\ \boxed{3}\ \textit{sign}\ > \\
\text{PHONS} & \boxed{5}\ \textit{list-of-phons}
\end{array}\right] \\
\text{COMP2} & \left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{signphon}} \\
\text{SIGNS} & \boxed{4}\ \textit{nelist-of-signs} \\
\text{PHONS} & \boxed{6}\ \textit{list-of-phons}
\end{array}\right] \\
\text{TEST} & \left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{prec}} \\
\text{C} & \boxed{1}\ \textit{list-of-phons} \\
\text{A} & \boxed{5}\ \textit{list-of-phons} \\
\text{B} & \boxed{6}\ \textit{list-of-phons}
\end{array}\right] \\
\text{REC} & \left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{lpp-}{\ll}} \\
\text{PHON} & \boxed{1}\ \textit{list-of-phons} \\
\text{BASE} & \boxed{2}\ \textit{nelist-of-signs}
\end{array}\right]
\end{array}\right]
$$

As there is no specialized attribute sensitive function involved, the sort *lpp-*$\ll$ could be fully defined in terms of previously given sorts.

The setup of the sort *lpp-*$\ll$ implements the idea of always leaving out the first element of the BASE list for precedence checking. However, except for the first element of the original BASE list, every element of the original list occurs as a second element at some level of recursion. Therefore, it is only the very first element, the head daughter, on whose phonology no linearization restriction is imposed by the attribute LPP-$\ll$. In (4.96) an example *lp-hc* object is shown. To stress the relevant point, only the PHON, BASE and the TEST values are given. It can be seen that even though the sign with the PHON value *a* is the first on the BASE list, *a* may be the last element on the overall PHON value.

(4.96) *Parts of an extended lp-hc*

$$
\begin{bmatrix}
\textit{lp-hc} & \\
\text{PHON} & <b, c, d, a> \\
\ldots & \\
\text{BASE} & \left\langle [\text{PHON} <a>], [\text{PHON} <b>], [\text{PHON} <c>], [\text{PHON} <d>] \right\rangle \\
\text{LPP-}\ll & \begin{bmatrix}
\textit{lpp-}\ll\textit{1} & \\
\text{PHON} <b, c, d, a> & \\
\text{BASE} \left\langle [\text{PHON} <a>], [\text{PHON} <b>], [\text{PHON} <c>], [\text{PHON} <d>] \right\rangle & \\
\ldots & \\
\text{TEST} & \begin{bmatrix} \textit{prec} \\ \text{C} <b, c, d, a> \\ \text{A} <b> \\ \text{B} <c, d> \end{bmatrix} \\
\text{REC} & \begin{bmatrix}
\textit{lpp-}\ll\textit{1} & \\
\text{PHON} <b, c, d, a> & \\
\text{BASE} \left\langle [\text{PHON} <b>], [\text{PHON} <c>], [\text{PHON} <d>] \right\rangle & \\
\ldots & \\
\text{TEST} & \begin{bmatrix} \textit{prec} \\ \text{C} <b, c, d, a> \\ \text{A} <c> \\ \text{B} <d> \end{bmatrix} \\
\text{REC} & \boxed{1}
\end{bmatrix}
\end{bmatrix}
\end{bmatrix}
$$

where tag $\boxed{1}$ is expanded as the following AVM:

$$
\boxed{1} = \begin{bmatrix}
\textit{lpp-}\ll\textit{1} & \\
\text{PHON} <b, c, d, a> & \\
\text{BASE} \left\langle [\text{PHON} <c>], [\text{PHON} <d>] \right\rangle & \\
\ldots & \\
\text{TEST} & \begin{bmatrix} \textit{prec} \\ \text{C} <b, c, d, a> \\ \text{A} <d> \\ \text{B} < > \end{bmatrix} \\
\text{REC} & \begin{bmatrix} \textit{lpp-}\ll\textit{0} \\ \text{PHON} <b, c, d, a> \\ \text{BASE} \left\langle [\text{PHON} <d>] \right\rangle \end{bmatrix}
\end{bmatrix}
$$

In the discussion of the BASE list, it was in fact only (LP-$\ll$) which really made reference to the specific ordering properties imposed on that list. The other LP principles just referred to inherent properties of the signs on that list ((LP-6) and (LP-7)). This is of course due to the fact that we are formalizing LP rules presented within a GPSG or HPSG framework, where there are only restricted possibilities for LP rules (cf. the discussion in chapter 1 and section 4.1). Pollard and Sag's (LP-$\ll$) goes beyond these possibilities, but it does not yet use the entire potential of the present system. Avoiding any commitment to empirical issues, we still want to stress that we expect LP principles for German which refer to the position of elements on the BASE list as well as to inherent properties of these elements.

As a tentative example, we are going to consider a generalization of Lenerz (1977, section 2.3.2) definiteness condition. The definiteness condition says that the phonology of a definite full NP complement may precede the phonology of an indefinite full NP complement. An indefinite on the other hand may only precede a definite if the indefinite is a dative NP and the definite an accusative NP. It may be interesting from an empirical as well as from a technical point of view to try to link the definiteness condition to positions on the COMPS list, abstracting away from case. As the order of elements on the COMPS list is motivated via the binding theory, binding facts have to be considered whenever the ordering on the COMPS list is concerned.

Binding facts in German seem to suggest that it is better to bind a dative anaphora by an accusative NP than the other way. This observation leads to an ordering on the COMPS list where an accusative complement precedes a dative. This ordering is, however, exactly the reverse of the ordering expected from linearization data as discussed in Lenerz (1977).

Just to make a technical point, we shall adopt some suggestions made by Primus (1989). Investigating word order and binding data, she observes a contrast in grammaticality between two groups of bitransitive verbs in German. In order to account for the differences within a GB framework, Primus assumes that binding in German is sensitive to some base generated constituent order as well as to morphological case. According to her, there is an overall tendency that an accusative NP binds a dative anaphora more easily than a dative NP would bind an accusative anaphora. Nevertheless with certain verbs even binding a dative anaphora by an accusative NP yields a not fully grammatical sentence. Primus further notices that it is exactly this group of verbs that seems to exhibit an unmarked complement order where the dative precedes the accusative.

Translating the proposals made in Primus (1989) into HPSG, one may encode the case condition on binding into some general part of the binding theory for German as it holds for all verbs. The difference between the particular verbs can be reflected in the ordering of the elements on the SUBCAT/COMPS list. In this case, for some bitransitive verbs the accusative complement precedes the dative, for others the dative complement precedes the accusative on the COMPS list. It is plausible to encode such an idiosyncratic property of a verb in the COMPS list, as subcategorization properties in general are idiosyncratic. It is furthermore clear that the difference between the two groups of verbs has to be expressed within an attribute whose value is available to the binding theory as the difference is relevant to binding facts. The condition on morphological case, however, is valid for all verbs. Thus, it can be hard-wired into the binding theory for German.

These speculative remarks on binding only serve to give some motivation to adopt a version of (LP-$\ll$) for German which can be understood as a generalized definiteness condition. We assume that for the verbs treated in Lenerz (1977), a dative NP precedes an accusative NP on the COMPS list of the lexical entry. Other bitransitive verbs have exactly the reverse relative ordering of their accusative and dative complements. In both cases, however, there is a correlation between the position on the COMPS list, the definiteness of a complement and the possibilities for linearization. This correlation is expressed as informal LP principle (LP-def) in (4.97).

(4.97) *(LP-def)*

> If a phrase x has two daughters i and j such that
> i is a definite full NP complement and
> j is an indefinite full NP complement and
> i's SYNSEM precedes j's SYNSEM on the COMPS list of x's head daughter
> then i's PHON precedes j's PHON on x's PHON.

This LP principle is responsible for the typical grammaticality pattern found in (Lenerz 1977):

(4.98) *Grammaticality patter for (LP-def)*

1. order on the COMPS list:
   NP_{def}, NP_{indef}
   orders on the PHON list:
   NP_{def}, NP_{indef}
   $*$ NP_{indef}, NP_{def}

2. order on the COMPS list:
   NP_{indef}, NP_{def}
   order on the PHON list:
   NP_{indef}, NP_{def}
   NP_{def}, NP_{indef}

Only one out of the four possible linearizations is ungrammatical according to (LP-def). In (4.98-1) the antecedent of the LP principle given in (4.97) is met, as there are two full NP complements, a definite and an indefinite one, where the definite NP precedes the indefinite on the COMPS list. Thus the phonology of the definite NP has to precede the phonology of the indefinite. This renders the second linearization ungrammatical.

In the case of (4.98-2), the order of the full NP complements does not meat the requirements of the antecedent of (LP-def), as the indefinite NP precedes the definite on the COMPS list. Therefore the LP principle stated in (4.97) is trivially met independent of the linearization. Thus, both possible orderings given in (4.98-2) are grammatical.

The LP principle given as (LP-def) in (4.97) and the grammaticality pattern that follows from it in (4.98) will be the subject of the remainder of the present section. First it will be shown how the LP principle stated as (LP-def) can be captured within the present framework. Then we shall discuss how the grammaticality pattern in (4.98) is accounted for in a different approach to LP, the one taken by Uszkoreit (1986a). Third we show that the mechanisms used in Uszkoreit (1986a) can be integrated into the present framework as well. We further argue that a theory of LP that does without such mechanisms is preferable.

Following our general architecture, the LP principle (LP-def) is encoded as an attribute LPP-DEF.This attribute occurs on *lp-hc* objects and bears values of the sort *lpp-def*. Not going into details, the sort *lpp-def* is very similar to the sort *lpp-*$\ll$ illustrated under (4.95) and (4.96). Both LP principles are recursive and take the BASE list as their auxiliary structure. Furthermore, both LP principles postulate that the precedence relation holding among elements on the BASE list corresponds to the ordering of their phonologies. The only difference between the sorts *lpp-*$\ll$ and *lpp-def* lies in the elements considered for precedence checking. In the case of *lpp-*$\ll$ all elements of the BASE list except for the first have been involved. The sort *lpp-def* only considers full NPs. At every step in the recursion the current BASE list of the *lpp-def* object is a suffix of the original BASE list. If the FIRST value of this suffix is a definite full NP, its phonology is put into the A value of some *prec* object. Then, the phonology of all indefinite full NPs that occur on the REST value of the suffix are collected and the resulting list is put into the B value of the same *prec* object. Within the *prec* object the correct precedence relation is checked. In (4.99) the appropriateness conditions and the theory of *lpp-def* are given.

(4.99) *A sketch of the sort lpp-def:*

Appropriateness Conditions for the sort *lpp-def* :

lpp-def	PHON	*list-of-phons*
	BASE	*elist-of-signs*

lpp-def0	PHON	*list-of-phons*
	BASE	*elist-of-signs*
lpp-def1	PHON	*list-of-phons*
	BASE	*nelist-of-signs*
	FIND-DEF	*find-def*
	FIND-INDEF	*find-indef*
	COMP1	*signphon*
	COMP2	*signphon*
	TEST	*prec*
	REC	*lpp-def*

Theory of *lpp-def* :

$$\begin{bmatrix} \textit{lpp-def1} \end{bmatrix} \Longrightarrow \begin{bmatrix} \textit{lpp-def1} & \\ \text{PHON} & \boxed{1}\ \textit{list-of-phons} \\ \text{BASE} & \begin{bmatrix} \textit{nelist-of-signs} \\ \text{FIRST}\ \boxed{2}\ \textit{sign} \\ \text{REST}\ \boxed{3}\ \textit{list-of-signs} \end{bmatrix} \\ \text{FIND-DEF} & \begin{bmatrix} \textit{find-def} \\ \text{INPUT}\ < \boxed{2}\ \textit{sign} > \\ \text{OUTPUT}\ \boxed{4}\ \textit{list-of-signs} \end{bmatrix} \\ \text{FIND-INDEF} & \begin{bmatrix} \textit{find-indef} \\ \text{INPUT}\ \boxed{3}\ \textit{list-of-signs} \\ \text{OUTPUT}\ \boxed{5}\ \textit{list-of-signs} \end{bmatrix} \\ \text{COMP1} & \begin{bmatrix} \textit{signphon} \\ \text{SIGNS}\ \boxed{4}\ \textit{list-of-signs} \\ \text{PHONS}\ \boxed{6}\ \textit{list-of-phons} \end{bmatrix} \\ \text{COMP2} & \begin{bmatrix} \textit{signphon} \\ \text{SIGNS}\ \boxed{5}\ \textit{list-of-signs} \\ \text{PHONS}\ \boxed{7}\ \textit{list-of-phons} \end{bmatrix} \\ \text{TEST} & \begin{bmatrix} \textit{prec} \\ \text{C}\ \boxed{1}\ \textit{list-of-phons} \\ \text{A}\ \boxed{6}\ \textit{list-of-phons} \\ \text{B}\ \boxed{7}\ \textit{list-of-phons} \end{bmatrix} \\ \text{REC} & \begin{bmatrix} \textit{lpp-def} \\ \text{PHON}\ \boxed{1}\ \textit{list-of-phons} \\ \text{BASE}\ \boxed{3}\ \textit{nelist-of-signs} \end{bmatrix} \end{bmatrix}$$

A comparison to the sort *lpp-*$\ll$ in (4.95) shows that the new sort bears two extra attributes, FIND-DEF and FIND-INDEF, which relativize the checking procedure to elements with certain properties. Thus, the sorts *find-def* and *find-indef* are search functional sorts like *find-pron* (cf. (4.43) and (4.44)) but looking for (in)definite full NPs instead of pronouns. With these remarks in mind, the AVM given in (4.99) is enough to illustrate what the LP principle (LP-def) looks like in the present framework.

After the technical comments, it should be noted that the adoption of (LP-def) has some consequences for the set of LP principles. Thus it might turn out that the rules that made reference to thematic roles ((LP-3) – (LP-5)) can — at least partially — be substituted by LP principles which refer to the ordering of complements, i.e. the BASE list. If such a substitution is possible, (LP-def) clearly covers a broader range of data than traditional approaches.

Uszkoreit (1986a) introduces a theory of LP that is close to the classical GPSG conception as resumed in section 4.1.1. The main difference between the (Uszkoreit 1986a) system and the one from (Gazdar et al. 1985) is that Uszkoreit allows for LP rules that apply disjunctively.[58] This deviation from classical LP rules enables him to account for a certain interaction between linearization regularities. In his system the grammaticality pattern given in (4.98) can be explained under the assumption of two disjunctively combined LP rules. One may assume that the first NP on the COMPS lists in (4.98) bears the thematic role 'goal', the second one is a 'theme'. Then (LP-5) comes into play which orders the goal's phonology before the theme's PHON value. This LP rule correctly licences the first orderings of the examples (4.98-1, 2), but it wrongly rules out the second example in (4.98-2), where a definite theme is allowed to precede an indefinite goal. Thus, there has to be an extra LP rule which orders the phonology of definites before the one of indefinites. We can informally write this LP rule down as 'definite $\prec$ indefinite'[59] This rule, again, is too restrictive, as it allows only for the first example in (4.98-1) and the second one in (4.98-2), excluding the grammatical first sentence of (4.98-2). Thus, the two LP rules have to be understood as being disjunctively combined, i.e. it is enough for a sentence to satisfy one LP rule in order to be grammatical.

[58] In addition to the disjunctive application of LP rules (Uszkoreit 1986a, p. 19 ff.), some weightening is introduced on the disjuncts within a 'complex LP rule' to account for degrees of grammaticality (Uszkoreit 1986a, pp. 24-27). The framework of the present thesis has no well established means to express degrees of grammaticality. Thus, this second aspect of Uszkoreit (1986a) cannot be expressed within the present formalism.

[59] As far as we know, Uszkoreit does not use an LP rule that is sensitive to definiteness properties of the elements to be ordered. The subsequent discussion is therefore rather hypothetic in nature. Nevertheless, such an LP rule can be given in the (Uszkoreit 1986a) system and its interaction with (LP-5) would be as described in the text.

As (LP-def) and a combination of (LP-5) with an 'definite $\prec$ indefinite' rule lead to the same empirical results, it is hard to decide which system is more adequate. We consider it an advantage of the present approach that the pattern in (4.98) can be expressed in a single LP principle which generalizes over thematic roles.[60] But even if it turned out that disjunctively applying linearization regularities are the empirically most adequate tool to describe word order phenomena, the LP component put forward in this thesis would have to be extended only slightly, as we will demonstrate next.

In section 4.7.1, a way to formalize (LP-5) has been presented. This LP principle orders the phonology of a 'goal' before the phonology of a 'theme'. For that purpose a new sort *lpp-gt* has been introduced as the value of an LP principle attribute that uses the EMBED structure.

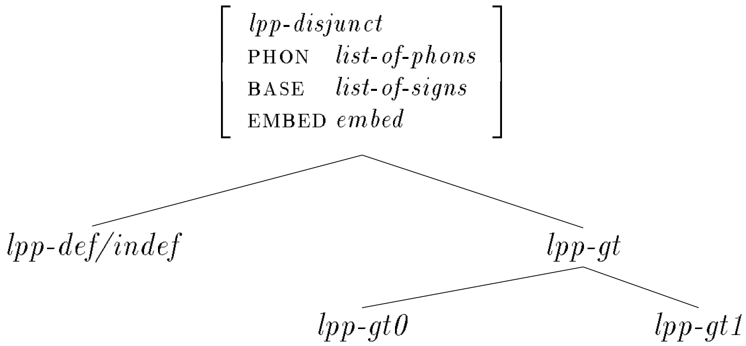
Assume that we have defined a sort *lpp-def/indef*. This sort encodes an LP principle which orders the phonologies of definite full NP complements before the phonologies of indefinite full NP complements.[61] The new sort *lpp-def/indef* encodes a 'simple' LP principle in the terminology of section 4.1.2. The auxiliary structure needed for the encoding is the BASE list.

In order to build a disjunctive LP principle attribute, the two sorts *lpp-gt* and *lpp-def/indef* have to be brought together as subsorts of a new sort *lpp-disjunct*. This sort appears as the value of an LP principle attribute LPP-DISJUNCT. As the sorts *lpp-gt* and *lpp-def/indef* need different auxiliary structures, both structures have to be present at every object of sort *lpp-disjunct*. As they stand so far, the sorts *lpp-def/indef* and *lpp-g/t* have only one auxiliary structure attribute each. In order to appear as subsorts of *lpp-disjunct*, they have to be extended in such a way that both are specified for the attributes BASE and EMBED. This yields the following partition of the sort *lpp-disjunct*.

[60] This is an especially attractive feature as long as there is no explicit theory of thematic roles available in HPSG.

[61] For this LP principle to work, only non-subjects are to be considered. This extra restriction would have to be adopted in the Uszkoreit style of LP rules as well.

(4.100) *The partition of lpp-disjunct*



Except for the additional attribute BASE on *lpp-gt*, this sort remains as defined in section 4.7.1. When an *lpp-disjunct* appears as the value of an LP principle attribute LPP-DISJUNCT, it has to be guaranteed by the theory of *lp* that the values of the PHON attribute and the auxiliary structures BASE and EMBED are structure shared between the LP object and the attributes occurring in the value of the LPP-DISJUNCT attribute. The AVM in (4.101) illustrates these parts of the theory of *lp*.

(4.101) *Parts of the theory of lp-hc involving the* LPP-DISJUNCT *attribute*

$$\begin{bmatrix} \textit{lp-hc} \end{bmatrix} \Longrightarrow \begin{bmatrix} \textit{lp-hc} & \\ \text{PHON} & \boxed{1}\ \textit{list-of-phons} \\ \ldots & \\ \text{BASE} & \boxed{2}\ \textit{list-of-signs} \\ \text{EMBED} & \boxed{3}\ \textit{embed} \\ \ldots & \\ \text{LPP-DISJUNCT} & \begin{bmatrix} \textit{lpp-disjunct} & \\ \text{PHON} & \boxed{1}\ \textit{list-of-phons} \\ \text{BASE} & \boxed{2}\ \textit{list-of-signs} \\ \text{EMBED} & \boxed{3}\ \textit{embed} \end{bmatrix} \end{bmatrix}$$

Given this way, the sort *lpp-disjunct* is an encoding of two disjunctively connected LP principles. The disjunction arises from the requirement that objects always are to be species (cf. section 1.2). Therefore, the value of the LPP-DISJUNCT attribute has to be either an *lpp-def/indef*, an *lpp-gt0* or an *lpp-gt1*, the species that are subsorts of *lpp-disjunct*. We further know that if the value is of species *lpp-def/indef*, the LP principle is met that enforces definites to precede indefinites. If the LPP-DISJUNCT value is a species of *lpp-gt*, on the other hand, (LP-5) is met, i.e. the goal precedes the theme. As a consequence, the two LP principles can be thought of as applying disjunctively.

Looking at the partition of *lpp-disjunct* in (4.100) again, the difference between disjunctive LP principles and those used in the rest of this thesis becomes evident. First, the LP principle attribute LPP-DISJUNCT takes two auxiliary structures, the other LP principles only need one. If more than two linearization regularities are to be put together disjunctively we might even need all the auxiliary structures within a single LP principle attribute. Second, so far the LP principles could be characterized as '(restricted) simple' or 'recursive'. *lpp-disjunct* encodes both, a simple and a recursive LP principle (*lpp-def/indef* versus *lpp-gt*). These points illustrate that introducing disjunctive LP principles into the present framework would not change much in the overall architecture of the LP component; it would, however, lead to a loss in conspicuity.

Summing up, we have discussed three major issues. First, it has been shown how a new auxiliary structure, the BASE list, can be constructed. Second, a few LP principles have been defined with reference to the BASE structure. These LP principles were of decreasing complexity. Third, the LP component of this thesis and the LP system presented in Uszkoreit (1986a) have been compared. We concluded that the Uszkoreit system can be fully integrated into a slightly modified version of the present LP component. Such modifications, however, seem not to be motivated, as the data possibly envisioned in Uszkoreit (1986a) can be accounted for in other ways, too. With this result, we leave the BASE list aside and shortly address the third auxiliary structure.

4.7.3 Comments on the SCOPOI structure

The third auxiliary structure proposed in section 4.2 has been called SCOPOI. Its value will presumably be a list of signs. This list represents the elements of the ALL-DTRS list in their scope inclusion relation. The SCOPOI list will be needed to express linearization regularities such as (LP-8), which enforces scope including elements to precede scope included ones. The discussion in the present section will be rather speculative, indicating that the construction of the SCOPOI structure is less evident than it has been the case for the EMBED and BASE structures.

As we do not intend to address any semantic issues here, we will only sketch how the information needed to build up the SCOPOI list can be gained within a *phrase*. In the following, we adopt the treatment of quantifiers given in Pollard and Sag (1994, section 8.2) without questioning its empirical adequacy. In the (Pollard and Sag 1994) framework, there are basically three attributes responsible for building up the scope relations among the elements of a sentence. These are the QUANTS attribute that occurs inside a SYNSEM LOC CONTENT value of sort *psoa*, and the QSTORE and RETRIEVED attributes which are specified on *signs*.[62] The

[62]According to Pollard and Sag (1994), the value of the QUANTS and the RETRIEVED

relation among the values of the three attributes is taken care of by the SEMANTICS PRINCIPLE (Pollard and Sag 1994, p. 323 and p. 401). Without going into details of the semantic analysis of Pollard and Sag (1994), quantifiers enter the syntactic structure with a nonempty value on their QSTORE attribute. Every element in the QSTORE values of the daughters of a *phrase* either ends up in the *phrase*'s own QSTORE value or in the *phrase*'s RETRIEVED value. In the last case, it also appears on the QUANTS list within the *phrase*'s CONTENT value.

The QUANTS list is a list of quantifiers that orders them according to their scope relations. The QUANTS value of a *phrase* is a concatenation of the QUANTS list of the head daughter and the newly retrieved quantifiers. As the new quantifiers are added to the beginning of the QUANTS list, they have wider scope than the elements that are already present on the head daughter's QUANTS list.

Let us look at a simple example that illustrates the retrieving mechanism of Pollard and Sag (1994).

(4.102) *The scope relations in a phrase*[63]

$$
\begin{bmatrix}
\textit{phrase} & \\
\text{PHON} & \langle \textit{jeder, Student, ein, Gedicht, kennt} \rangle \\
\text{SYNSEM} & \text{LOCAL CONTENT} \begin{bmatrix} \textit{psoa} \\ \text{QUANTS} \ \langle \boxed{2}, \boxed{3} \rangle \\ \text{NUCLEUS} \ \boxed{1} \end{bmatrix} \\
\text{DTRS} & \begin{bmatrix} \textit{head-comp-struc} & \\ \text{H-DTRS} & \boxed{4} \begin{bmatrix} \textit{word} & \\ \text{PHON} & \langle \textit{kennt} \rangle \\ \text{SYNSEM LOCAL CONTENT} & \begin{bmatrix} \textit{psoa} \\ \text{QUANTS} \ \textit{elist-of-quantifiers} \\ \text{NUCLEUS} \ \boxed{1} \end{bmatrix} \end{bmatrix} \\
\text{C-DTRS} & \left\langle \boxed{5} \begin{bmatrix} \textit{phrase} \\ \text{PHON} \ \langle \textit{jeder, Student} \rangle \\ \text{QSTORE} \ \langle \boxed{2} \rangle \end{bmatrix}, \boxed{6} \begin{bmatrix} \textit{phrase} \\ \text{PHON} \ \langle \textit{ein, Gedicht} \rangle \\ \text{QSTORE} \ \langle \boxed{3} \rangle \end{bmatrix} \right\rangle \\
\text{ALL-DTRS} & \langle \boxed{4}, \boxed{5}, \boxed{6} \rangle
\end{bmatrix} \\
\text{QSTORE} & \textit{elist-of-quantifiers} \\
\text{RETRIEVED} & \langle \boxed{2}, \boxed{3} \rangle
\end{bmatrix}
$$

Even though the example in (4.102) is as simple as possible, it illustrates a problem for the present approach. In the ALL-DTRS list, there is

attributes are lists of *quantifiers*, the values of the attribute QSTORE is thought to be a set of *quantifiers*. As sets are not provided by the formalism adopted in this thesis, we assume the values of all three attributes to be of the sort *list-of-quantifiers*. Nevertheless we will not refer to any ordering properties of the values of the attribute QSTORE in order to respect some basic intuitions that underlie the notion of 'set'.

[63]The German sentence in (4.102) is a translation of the example used in Pollard and Sag (1994, p. 324, example (20b)), i.e. the English sentence *every student knows a poem*.

no information about the scope inclusion among the daughters. This information can only be taken from the QUANTS value of the entire *phrase*. The reason for this is that quantifiers only appear on the QUANTS value when they have been retrieved. They can however only be retrieved on a *phrase* with a CONTENT attribute of value *psoa* that 'dominates' the occurrence of the quantifier within the DTRS structure. Thus, the QUANTS list on the head daughter in (4.102) is empty as there are no quantifiers retrieved at the head daughter. On the overall *phrase* in (4.102), the quantifiers tagged as $\boxed{2}$ and $\boxed{3}$ are retrieved and they appear on the QUANTS list of the *phrase* in the order of their relative scope inclusion. In the example, the quantifier associated with *jeder Student* has scope over the quantifier *ein Gedicht*. What is relevant to the interface to the LP component, however, is the information specified on the elements of the ALL-DTRS list. From this information, nothing can be inferred about scope relations.

Further complications arise when the elements of the daughters' QSTOREs are not retrieved locally. For illustration, consider the sentence in (4.103-1) with the scope interpretation informally indicated in (4.103-2).

(4.103) *Nonlocal quantifier retrievement*

1. Ich glaube, daß jeder Student ein Gedicht kennt
   I think that every student a poem knows
   'I think that every student knows a poem'

2. (jeder Student x)[(ein Gedicht y) [ich glaube, daß x y kennt]][64]

The example in (4.103) is more complicated than the one given in (4.102), in that the two quantifiers are only retrieved in the matrix clause. This implies that they do not occur on the QUANTS list of the sentence headed by the verb *kennt*, but in the QUANTS list of the sentence headed by *glaube*. Nevertheless the phonologies corresponding to these quantifiers occur as parts of the phonology of the subordinate clause, i.e. they have to be ordered within the subordinate clause. What is in fact needed is the information of the relative scope of quantifiers as soon as these quantifiers appear in the syntactic structure, no matter where they are retrieved.

[64] The reading given in (4.103-2) is empirically not available. Nevertheless it is predicted by the scope theory given in Pollard and Sag (1994). As we are only concerned with the interface between the scope theory and the LP component, we do not want to propose any changes in the (Pollard and Sag 1994) system.

One might think of a technical answer to this requirement. In the following, we shall give an outline of such a solution.[65] We assume a new attribute ALL-QUANTS present at every *content* object, i.e. at every SYNSEM LOCAL CONTENT value of a sign. The value of the ALL-QUANTS attribute is a *list-of-quantifiers*. Just as on the QUANTS list, the elements on the ALL-QUANTS list are ordered according to their scope relations. Intuitively, the ALL-QUANTS list contains all the quantifiers of a certain domain no matter whether they have been retrieved already or are still to be retrieved. In fact, there may be quantifiers on a *sign*'s ALL-QUANTS list that have not even been introduced syntactically. It is, however, guaranteed that a quantifier will appear on the ALL-QUANTS list of the head daughter of the *const-struc* that introduces the quantifier syntactically.

For the time being we assume the attribute ALL-QUANTS to be present at every species of the sort *content*. This attribute is certainly needed at the species *psoa*, and probably of no use on the species *quantifier*. The attribute, however, does no harm on species where it is not needed empirically. So, further empirical results may motivate a change in the appropriateness conditions of the subsorts of *content*. We consider it an advantage of the present proposal that it is more general than necessary but shows clear options for turning it into a more restrictive theory. The latter point has been our main reason to put the ALL-QUANTS attribute into the CONTENT value instead of defining the new attribute for *signs* in general. If it were localized at every *sign* it would become a hard task to constrain the occurrence of the attribute to sign with particular CONTENT values. Furthermore, we do not expect the occurrence of the attribute ALL-QUANTS to covary with anything else but the CONTENT value of a *sign*.

There are two ways to construct the ALL-QUANTS value of a *sign*, depending on some properties of the sign. Firstly, the ALL-QUANTS value is structure shared with the QUANTS value. This happens in *phrase*s with an empty QSTORE value and a CONTENT value of sort *psoa*. For all other kinds of *signs* nothing is directly said about their ALL-QUANTS value.

Secondly, a *phrase* structure shares its ALL-QUANTS value with some of its daughters. The selected daughters are exactly those to which the first principle for determining the ALL-QUANTS value does not apply. Thus, daughters of the sort *word*, daughters whose CONTENT value is not of sort *psoa* or those with a nonempty QSTORE attribute get their ALL-QUANTS value from the mother.

With these two options a sign may either get the ALL-QUANTS value

[65] The solution put forward in the present section is preliminary in character. It is firmly tied to the requirements of this thesis and to the semantic analysis expressed in Pollard and Sag (1994). Nevertheless, we hope that it will give an idea of the direction where an adequate solution can be found.

by virtue of own specifications (in the first way), or by inheritance from its mother (in the second way). For signs that have a nonempty QSTORE value or that are of sort *word*, but that do not occur as daughters in a structure, nothing is said concerning their ALL-QUANTS value.

The principles governing the value of the ALL-QUANTS attribute have to be stated as parts of the SEMANTICS PRINCIPLE (Pollard and Sag 1994, p. 323 and p. 401). This principle states the relations that hold between the values of the attributes QUANTS, QSTORE and RETRIEVED as they are used in Pollard and Sag (1994). Adding the ALL-QUANTS attributes forces us to extend the SEMANTICS PRINCIPLE. But as we leave the overall treatment of quantifiers unchanged, just a new condition has to be added to the principle. It is given informally in (4.104).

(4.104) THE SEMANTICS PRINCIPLE *additional condition:*

(c) Conditions on the ALL-QUANTS value

i. In a *phrase* with an empty QSTORE value and a SYNSEM LOCAL CONTENT value of sort *psoa*, the SYNSEM LOC CONTENT QUANTS value is identical with the SYNSEM LOC CONTENT ALL-QUANTS value.

ii. For every daughter D of a *phrase* P it holds that if D is not a *phrase* with an empty QSTORE value and a SYNSEM LOCAL CONTENT value of sort *psoa*, then D's SYNSEM LOC CONTENT ALL-QUANTS value is identical with P's SYNSEM LOC CONTENT ALL-QUANTS value.

The two cases of condition (c) of the SEMANTICS PRINCIPLE can best be illustrated by looking at the example in (4.102) again. In (4.105) this AVM is augmented by the attribute ALL-QUANTS.

(4.105) *The* ALL-QUANTS *values in a phrase*

$$\left[\begin{array}{lll}
\multicolumn{3}{l}{\textit{phrase}} \\
\text{PHON} & \multicolumn{2}{l}{\langle \textit{jeder, Student, ein, Gedicht, kennt}\rangle} \\
\text{SYNSEM} & \text{LOCAL CONTENT} & \left[\begin{array}{ll} \textit{psoa} & \\ \text{QUANTS} & \boxed{7} < \boxed{2}, \boxed{3} > \\ \text{ALL-QUANTS} & \boxed{7} \\ \text{NUCLEUS} & \boxed{1} \end{array}\right] \\
\text{DTRS} & \multicolumn{2}{l}{\left[\begin{array}{lll} \multicolumn{3}{l}{\textit{head-comp-struc}} \\ \text{H-DTRS} & \boxed{4} & \left[\begin{array}{ll} \textit{word} & \\ \text{PHON} & \langle \textit{kennt}\rangle \\ \text{SYNSEM LOCAL CONTENT} & \left[\begin{array}{ll} \textit{psoa} & \\ \text{QUANTS} & \textit{elist-of-quantifiers} \\ \text{ALL-QUANTS} & \boxed{7} \\ \text{NUCLEUS} & \boxed{1} \end{array}\right] \\ \text{QSTORE} & \textit{elist-of-quantifiers} \end{array}\right] \\ \text{C-DTRS} & \multicolumn{2}{l}{\left\langle \boxed{5} \left[\begin{array}{ll} \textit{phrase} & \\ \text{PHON} & \langle \textit{jeder, Student}\rangle \\ \text{SS} \ldots & [\text{ALL-QUANTS} \ \boxed{7}] \\ \text{QSTORE} & < \boxed{2} > \end{array}\right], \boxed{6} \left[\begin{array}{ll} \textit{phrase} & \\ \text{PHON} & \langle \textit{ein, Gedicht}\rangle \\ \text{SS} \ldots & [\text{ALL-QUANTS} \ \boxed{7}] \\ \text{QSTORE} & < \boxed{3} > \end{array}\right] \right\rangle} \\ \multicolumn{3}{l}{\text{ALL-DTRS} < \boxed{4}, \boxed{5}, \boxed{6} >} \end{array}\right]} \\
\text{QSTORE} & \multicolumn{2}{l}{\textit{elist-of-quantifiers}} \\
\multicolumn{3}{l}{\text{RETRIEVED} < \boxed{2}, \boxed{3} >}
\end{array}\right]$$

In the overall *phrase* in (4.105), the QSTORE value is empty and the CONTENT value is a *psoa*. Therefore part (ci) of the extended SEMANTICS PRINCIPLE (4.104) applies, ensuring identity between the QUANTS value and the ALL-QUANTS value. The head daughter in (4.105) has an empty QSTORE value and a *psoa* as CONTENT value, as well. But, as it is not a *phrase* but a *word*, its ALL-QUANTS specification is subject to condition (cii) of the SEMANTICS PRINCIPLE. By this condition, the ALL-QUANTS list of the head daughter is guaranteed to be identical with the ALL-QUANTS value of the mother. The COMP-DTRS (C-DTRS) in (4.105) both have an nonempty QSTORE. Furthermore, their CONTENT value is of sort *npro*. Thus, their ALL-QUANTS value, too, is identified with the ALL-QUANTS list of the entire *phrase*. By this, all parts of the addition made to the SEMANTICS PRINCIPLE of Pollard and Sag (1994) in (4.104) have been illustrated in the AVM in (4.105).

In the discussion of example (4.102) we noted that scope information is needed on the elements of the ALL-DTRS list in order to construct the SCOPOI structure. Incorporating the ALL-QUANTS list into the grammar, the necessary information can now be gained from the head daughter. We shall show next that the same mechanism is able to handle the wide scope reading intended in (4.103). An AVM for the highest *phrase* of the sentence in (4.103) is given in (4.106).

(4.106) *An example with wide scope readings*

$$\left[\begin{array}{ll} \textit{phrase} & \\ \text{PHON} & \langle \textit{ich, glaube, daß, jeder, Student, ein, Gedicht, kennt} \rangle \\ \text{SYNSEM} & \text{LOCAL CONTENT} \left[\begin{array}{ll} \textit{psoa} & \\ \text{QUANTS} & \boxed{7} \langle \boxed{2}, \boxed{3} \rangle \\ \text{ALL-QUANTS} & \boxed{7} \\ \text{NUCLEUS} & \boxed{9} \end{array}\right] \\ \text{DTRS} & \left[\begin{array}{ll} \textit{head-comp-struc} & \\ \text{H-DTRS} & \boxed{10} \left[\begin{array}{ll} \textit{word} & \\ \text{PHON} & \langle \textit{glaube} \rangle \\ \text{SYNSEM LOCAL CONTENT} & \left[\begin{array}{ll} \textit{psoa} & \\ \text{QUANTS} & \textit{elist-of-quantifiers} \\ \text{ALL-QUANTS} & \boxed{7} \\ \text{NUCLEUS} & \boxed{9} \end{array}\right] \\ \text{QSTORE} & \textit{elist-of-quantifiers} \end{array}\right] \\ \text{C-DTRS} & \left\langle \boxed{11} \left[\begin{array}{ll} \textit{phrase} & \\ \text{PHON} & \langle \textit{ich} \rangle \\ \text{SS} \ldots & [\text{ALL-QUANTS } \boxed{7}] \\ \text{QSTORE} & \textit{elist-of-quantifiers} \end{array}\right], \boxed{12} \left[\begin{array}{ll} \textit{phrase} & \\ \text{PHON} & \langle \textit{daß}, \ldots \rangle \\ \text{SS} \ldots & [\text{ALL-QUANTS } \boxed{7}] \\ \text{DTRS} & \boxed{8} \\ \text{QSTORE} & \langle \boxed{2}, \boxed{3} \rangle \end{array}\right] \right\rangle \\ \text{ALL-DTRS} & \langle \boxed{10}, \boxed{11}, \boxed{12} \rangle \end{array}\right] \\ \text{QSTORE} & \textit{elist-of-quantifiers} \\ \text{RETRIEVED} & \langle \boxed{2}, \boxed{3} \rangle \end{array}\right]$$

where tag $\boxed{8}$ *stands for the* DTRS *value in example (4.105).*

The AVM in (4.106) is very similar to the one given in (4.105). Again, the ALL-QUANTS value of the mother has to obey condition (ci) of (4.104), whereas the daughters are subject to condition (cii) of the SEMANTICS PRINCIPLE. This condition applies to the head daughter because it is a *word* not a *phrase*. It applies to the first complement daughter, tagged as $\boxed{11}$, because the CONTENT value of this sign is not a *psoa* but a *ppro* (personal pronoun). Furthermore, the second complement daughter is subject to the same condition, (cii) of (4.104), as its QSTORE value is not empty.

The tag $\boxed{8}$ used in (4.106) is meant to indicate that parts of the structure in (4.105) occur in the longer sentence as well. As a consequence, the ALL-QUANTS list of the deepest head verb in (4.106), the verb with the phonology *kennt*, contains the information about the two quantifiers that occur in the sentence. Thus, even though the quantifiers are only retrieved in the matrix clause, the information about their scope relation is made available in the embedded clause.[66]

[66]It should be noticed that if the matrix clause introduces a quantifier on its own, this quantifier, too, can be found on the ALL-QUANTS lists in the embedded clause. This extra information, however, does no harm as it need not be included into the SCOPOI list.

Having illustrated the treatment of the ALL-QUANTS list on an informal level, we next give a formalization of the new condition of the SEMANTICS PRINCIPLE introduced in (4.104). The first condition, (ci), can be directly accounted for in the theory of *phrase*. To encode condition (cii) of (4.104), a new attribute has to be introduced whose value expresses the relation between the ALL-QUANTS value of the *phrase* and the elements on the ALL-DTRS list. In (4.107) the appropriateness conditions for the sort *phrase* are given including the new attribute SEM-PRINC which is used for the encoding of condition (cii) of the SEMANTICS PRINCIPLE.

(4.107) *Appropriateness Conditions for the sort phrase , including the* SEM-PRINC *attribute:*

phrase	PHON	*list-of-phons*
	BUILD-PHON	*build-phon*
	LP	*lp*
	SYNSEM	*synsem*
	DTRS	*const-struc*
	QSTORE	*list-of-quantifiers*
	RETRIEVED	*list-of-quantifiers*
	SEM-PRINC	*sem-princ*

The appropriateness conditions given in (4.107) include the attributes given in Pollard and Sag (1994) and those introduced in the earlier chapters of this thesis. It should be recalled that we deviate from Pollard and Sag (1994) in assuming a list as value of the attribute QSTORE instead of a set. In addition to the attributes already discussed, the new attribute SEM-PRINC has been given. Within its value, of sort *sem-princ*, the condition imposed by (cii) of the SEMANTICS PRINCIPLE is checked.

As condition (cii) establishes a relation between the ALL-QUANTS value of a *phrase* and the elements on the ALL-DTRS list of that *phrase*, the sort *sem-princ* needs at least two attributes, one whose value is identical with the *phrase*'s ALL-QUANTS list, and one whose value is identical with the DTRS ALL-DTRS list of the *phrase*. In (4.108) it is indicated how the two new clauses of the SEMANTICS PRINCIPLE are encoded in the theory of *phrase*.

(4.108) *Theory of phrase (partially), as AVM:*

1. *Condition (ci) of the* SEMANTICS PRINCIPLE

$$\left[\begin{array}{l} \textit{phrase} \\ \text{SYNSEM LOC CONTENT } \textit{psoa} \\ \text{QSTORE } \textit{elist-of-quantifiers} \end{array}\right] \Longrightarrow$$

$$\left[\begin{array}{ll} \textit{phrase} & \\ \text{SYNSEM LOC CONTENT} & \left[\begin{array}{ll} \textit{psoa} & \\ \text{QUANTS} & \boxed{1}\ \textit{list-of-quantifiers} \\ \text{ALL-QUANTS} & \boxed{1}\ \textit{list-of-quantifiers} \end{array}\right] \end{array}\right]$$

2. *Condition (cii) of the* SEMANTICS PRINCIPLE

$$\left[\textit{phrase}\right] \Longrightarrow$$

$$\left[\begin{array}{ll} \textit{phrase} & \\ \text{SYNSEM LOC} & \text{CONTENT ALL-QUANTS } \boxed{1}\ \textit{list-of-quantifiers} \\ \text{DTRS ALL-DTRS } \boxed{2}\ \textit{list-of-signs} & \\ \text{SEM-PRINC} & \left[\begin{array}{ll} \textit{sem-princ} & \\ \text{ALL-DTRS} & \boxed{2}\ \textit{list-of-signs} \\ \text{ALL-QUANTS} & \boxed{1}\ \textit{list-of-quantifiers} \end{array}\right] \end{array}\right]$$

Condition (ci) of the SEMANTICS PRINCIPLE is stated in an implicational constraint with a complex antecedent. It specifies that the left-hand side of the condition has to be met only by *phrase*s with a CONTENT value of sort *psoa* and an empty QSTORE list. The second additional condition of the SEMANTICS PRINCIPLE has a simpler antecedent. The constraint given in (4.108-2), however, just contains the 'input specification' to the sort *sem-princ* where the condition is actually checked.

In order to fully formalize condition (cii), the sort *sem-princ* has to be looked at in some detail. In (4.109) the appropriateness conditions of the sort *sem-princ* are given.

(4.109) *Appropriateness Conditions for the sort sem-princ*

sem-princ	ALL-DTRS	*list-of-signs*
	ALL-QUANTS	*list-of-quantifiers*
sem-princ0	ALL-DTRS	*elist-of-signs*
	ALL-QUANTS	*list-of-quantifiers*
sem-princ1	ALL-DTRS	*nelist-of-signs*
	ALL-QUANTS	*list-of-quantifiers*
	REC	*sem-princ*

The sort *sem-princ* is a recursive sort as have been most of the sorts discussed in this thesis. In the case of *sem-princ* the recursion proceeds along the ALL-DTRS list. At every step in the recursion it is checked whether the ALL-DTRS FIRST element meets the specification requirement expressed in the SEMANTICS PRINCIPLE. The principle enforces the ALL-QUANTS value of the present ALL-DTRS FIRST element to be identical with the ALL-QUANTS value of the *sem-princ* object, if the ALL-DTRS FIRST element is not a *phrase* with an empty QSTORE value and a CONTENT value of sort *psoa*.

The sort *sem-princ* has to account for several cases. If the ALL-DTRS list is empty, the recursive checking has come to an end (*sem-princ0*). If it is nonempty (*sem-princ1*), it is known that there will be a next recursion step. Thus the values ALL-DTRS REST and REC ALL-DTRS are set identical, and the ALL-QUANTS value is structure shared with the REC ALL-QUANTS value. This is expressed in the first part of the theory of *sem-princ1* given in (4.110). For the cases where a special checking of the ALL-QUANTS value of the first element of the ALL-DTRS list is necessary, the second part of the theory of *sem-princ1* is needed.

In (4.110) the theory of *sem-princ* is given formally. Besides the terms stated there, two AVMs are given that show different possible *sem-princ1* objects.

(4.110) *Theory of sem-princ*

$$:\sim\textit{sem-princ1} \quad \rightarrow \quad (\text{:ALL-DTRS REST} \approx \text{:REC ALL-DTRS} \wedge \text{:ALL-QUANTS} \approx \text{:REC ALL-QUANTS})$$

$$\begin{array}{l}(:\sim\textit{sem-princ1}\\ \wedge\ \neg\ (\text{:ALL-DTRS FIRST}\sim\textit{phrase}\\ \quad \wedge\ \text{:ALL-DTRS FIRST QSTORE}\sim\textit{elist-of-quantifiers}\\ \quad \wedge\ \text{:ALL-DTRS FIRST SYNSEM LOC CONTENT}\sim\textit{psoa}))\\ \rightarrow\ (\text{:ALL-DTRS FIRST SYNSEM LOC CONTENT ALL-QUANTS} \approx \text{:ALL-QUANTS})\end{array}$$

Some examples for *sem-princ1* objects as AVMs:

$$\begin{bmatrix} \textit{sem-princ1} & \\ \text{ALL-DTRS} & \begin{bmatrix} \textit{list-of-signs} & \\ \text{FIRST} & \begin{bmatrix}\textit{phrase}\\ \text{SYNSEM LOC CONTENT } \textit{psoa}\\ \text{QSTORE } \textit{elist-of-quantifiers}\end{bmatrix}\\ \text{REST} & \boxed{1}\ \textit{list-of-signs}\end{bmatrix}\\ \text{ALL-QUANTS} & \boxed{2}\ \textit{list-of-quantifiers}\\ \text{REC} & \begin{bmatrix}\textit{sem-princ} & \\ \text{ALL-DTRS} & \boxed{1}\ \textit{list-of-signs}\\ \text{ALL-QUANTS} & \boxed{2}\ \textit{list-of-quantifiers}\end{bmatrix}\end{bmatrix}$$

$$
\left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{sem-princ1}} \\
\text{ALL-DTRS} & \left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{list-of-signs}} \\
\text{FIRST} & \left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{sign}} \\
\text{SYNSEM LOC CONTENT} & \left[\begin{array}{l}
\textit{ppro} \\
\text{ALL-QUANTS } \boxed{2} \ \textit{list-of-quantifiers}
\end{array}\right]
\end{array}\right] \\
\text{REST} & \boxed{1} \ \textit{list-of-signs}
\end{array}\right] \\
\text{ALL-QUANTS} & \boxed{2} \ \textit{list-of-quantifiers} \\
\text{REC} & \left[\begin{array}{ll}
\multicolumn{2}{l}{\textit{sem-princ}} \\
\text{ALL-DTRS} & \boxed{1} \ \textit{list-of-signs} \\
\text{ALL-QUANTS} & \boxed{2} \ \textit{list-of-quantifiers}
\end{array}\right]
\end{array}\right]
$$

The first AVM in (4.110) shows an *sem-princ1* whose first element on the ALL-DTRS list is unaffected by the condition (cii) of the SEMANTICS PRINCIPLE. The consequent of (cii) does not apply because the first element on the ALL-DTRS list is a *phrase* with an empty QSTORE and a CONTENT value of sort *psoa*. Thus, the *sem-princ1* object only meets the left-hand side of the first constraint of the theory of *sem-princ1*. The second AVM shows an *sem-princ1* that meets the left-hand side of both constraints. As an example, we have taken its CONTENT value to be of sort *ppro*. Thus, the ALL-QUANTS values of the *sem-princ1* object and its ALL-DTRS FIRST element are identical.

To ensure the effects of condition (c) of the SEMANTICS PRINCIPLE as given under (4.104), a *sem-princ* object has to be properly connected with the rest of a *phrase*. This has been done via the attribute SEM-PRINC, as part of the theory of *phrase*. In (4.108) the relevant bits of the theory of *phrase* were given as AVM.

With the addition to the theory of *phrase* given in (4.108) and the definition of the sort *sem-princ* in (4.109) and (4.110) the new condition of the SEMANTICS PRINCIPLE stated in (4.104) is fully defined. The extensions to the SEMANTICS PRINCIPLE made in the present section have been necessary to provide the elements of the ALL-DTRS list of a *phrase* with scope information. The availability of information on the relative scope of quantifiers is an important precondition to the construction of the SCOPOI list. We shall not go into details of the construction of the SCOPOI list as it is quite similar to the way the BASE list has been arrived at by reordering the ALL-DTRS list. Basically, the SCOPOI list will be constructed as a list that orders the ALL-DTRS elements with respect to the scope relations on the ALL-QUANTS list of the head-daughter. Constructing the SCOPOI list, two problems arise:

First, the elements of an ALL-QUANTS value are not *signs* but of sort *quantifier*. Yet, as it was the case for the relation between a *sign* and its SYNSEM value,[67] we assume that there is a unique mapping in any sentence between the *quantifier*s which occur in the sentence and

[67]cf. the sort *signsynsem* in (4.91) and (4.92).

the *signs* appearing as complements that take these quantifiers as only element on their QSTORE value.[68]

Second, the ordering on the SCOPOI list is not fully determined by the head daughter's ALL-QUANTS list, because not all daughters of a phrase are necessarily quantifiers. Thus some conventions have to be formulated for the ordering of nonquantifiers on the SCOPOI list.[69] It might be reasonable to collect all nonquantifiers that occur on the ALL-DTRS list and to put them at the end of the SCOPOI list in arbitrary order.

The SCOPOI list can be constructed along the lines just sketched. One can write an auxiliary sort *ws3* that establishes a relation between the ALL-DTRS list and some permutation of this list, which reflects the scope inclusion relations of a *phrase*'s daughters. Once the SCOPOI list is built up, an LP rule like (LP-8) which orders the phonology of scope including elements before the one of included elements, can easily be written as an LP principle. A sort encoding this LP principle would roughly look like *lpp-*$\ll$ in (4.95). Differences would be that *lpp-scopoi* uses the SCOPOI list instead of the BASE list and that the first element, the one with widest scope in this case, is included in the precedence checking procedure whereas nonquantifiers that occur on the SCOPOI list are not.[70]

The discussion of the SCOPOI structure has brought to light a very important point concerning the modularity of the present approach. At first sight, it may seem unintuitive to extend the SEMANTICS PRINCIPLE in order to account for word order phenomena. The extension made in (4.104) of the present section, however, did not directly concern the LP component. The new attribute ALL-QUANTS had to be introduced to provide the possibility to link the semantic component of the theory to the LP component. In section 4.1.2 we claimed that the PHON and the ALL-DTRS values of a *phrase* are the only values from the overall phrase that will be used in the LP component. Within the LP component, we need information about the scope relations that hold among the elements whose phonologies are to be ordered. Therefore, this information has to be provided via the ALL-DTRS list. Unfortunately, as we have shown, scope information is not present in the elements of the ALL-DTRS list, if one sticks to the scope theory of Pollard and Sag (1994). It could, however, be demonstrated that the introduction of a new attribute ALL-QUANTS in connection with an extension of the SEMANTICS PRINCIPLE

[68] As in the rest of this thesis, adjuncts are tacitly left aside.

[69] In the construction of the BASE list similar conventions were necessary for the position of the head daughter.

[70] In (4.95) the first element of the BASE list has been excluded, as it was the head daughter and the LP rule only applied to complement daughters.

are sufficient to make the required information available at the attributes whose values serve as input to the LP component. This illustrates that the extensions proposed in the present section do not lead to a change in the semantic theory assumed in Pollard and Sag (1994), they only serve to provide a connection between two components of the grammar.

With these speculative remarks on possibilities to make a SCOPOI auxiliary structure available for the LP component, section 4.7.3 can be concluded. It has been shown why the SCOPOI list cannot be constructed directly upon the scope theory of Pollard and Sag (1994) and the ideas about the LP component laid out in section 4.1.2 of this thesis. In a second step we demonstrated what extensions of the sorts *content* and *phrase* are needed to provide scope information at the interface between the semantics and the LP component of the grammar. Thirdly, a rough outline of how the SCOPOI list can be constructed once the necessary information is made available, has been given. In a last step, it has been shown how the SCOPOI list can be used as auxiliary list for LP principles such as (LP-8). A certain parallelism between the treatment of the SCOPOI list and the BASE list has been stressed for the last two steps.

The comments given above on the construction and the use of the SCOPOI auxiliary structure complete the discussion of those parts of an *lp-hc* object that are not treated in full formal detail in this thesis.

4.7.4 Concluding Remarks on an extended sort *lp-hc*

Section 4.7 took as its starting point a collection of linearization regularities as they have been formulated by authors such as Uszkoreit (1986a), Reape (1994), Pollard et al. (1994), Jacobs (1988) and Pollard and Sag (1987). The LP rules looked at have been selected to illustrate different kinds of linearization regularities that can be encountered in the literature.

The LP rules collected in the beginning of section 4.7 were divided into three groups, depending on the auxiliary structure needed to restate them as LP principles in the present framework. The first group, considered in 4.7.1, consisted of LP principles that could be expressed with respect to the EMBED auxiliary structure defined in section 4.3. In section 4.7.1 the emphasis of the presentation lay in the fact that new LP principles can easily be given in a formally precise way by using sorts defined in previous sections of this thesis. Thus, it was shown that once a collection of recursive sorts has been given, these sorts can be taken to encode new regularities. Beside this technical point, a possible treatment of thematic roles has been presented and a way to express linearization regularities sensitive to thematic roles without having an extra attribute THEMATIC-ROLE on complements.

Sections 4.7.2 and 4.7.3 contained a discussion of LP principles that refer to the position of elements on the obliqueness hierarchy and to scope inclusion properties of elements. In section 4.7.2 the construction of the BASE list has been described. This list is assumed to contain all the elements of the ALL-DTRS list but in an ordering that respects the obliqueness hierarchy. The construction of the BASE list turned out to be relatively easy, as it only involved sorts that were analogous to sorts already defined previously. Furthermore, the encoding of some LP principles using the BASE list has been shown. The section was concluded with a comparison between the present LP component and some of the ideas developed in Uszkoreit (1986a). This lead to a sketch of how to integrate disjunctive LP principles into the present theory.

Section 4.7.3 was concerned with the last auxiliary list we assumed to be present at an *lp-hc* object. This list was defined as ordering the ALL-DTRS element according to their scope relations. In the case of the SCOPOI list we first had to extend the theory of quantifiers of Pollard and Sag (1994). After this extension it could be argued that the construction of the SCOPOI list is not only possible but technically very similar to the way the BASE list has been built. At the end of section 4.7.3 a brief discussion of an LP principle was given.

The extensions of the sort *lp-hc* discussed in section 4.7 have been stated to argue for two aspects of the present theory:

First, once a collection of recursive sorts has been defined, these can be used as building blocks for all parts of the grammar. Second, the LP component argued for in this thesis is capable to express all linearization regularities expressed in the literature. Even disjunctive LP principles can be accounted for with just a slight modification of the present conception of an LP component.

As a consequence, the expressive power of the present theory makes it possibly for every HPSG grammar writer to stick to her favored set of LP principles, and still adopt the present framework. The efforts of formalizing empirical generalizations of word order phenomena becomes smaller and smaller with every recursive sort newly defined, as these sorts can be used extensively throughout the grammar.

5 The Missing Links (*sign*)

In section 4.1.2 we saw that objects of sort *lp* come in various varieties. Which one of them occurs in a particular phrase merely depends on the constituent structure of its daughters. This relation between constituent structures and *lp* objects was already present in the informal suggestions of Pollard and Sag (1987). They assumed that the relational extension `order-constituents` takes the value of the DTRS feature as input for determining the phonology of the entire phrase (cf. example (2.1), chapter 2). In the present setup the dependency between constituent structure and linear precedence is reflected by an implicational constraint in the theory of *phrase*. For head complement structures as they are discussed in this thesis, the constraint says that any phrase whose DTRS value is of sort *head-comp-struc* has an LP attribute with sort *lp-hc* as value.

The other parts of the following theory of phrase deal with the correct treatment of the interfaces between the LP and BUILD-PHON module and the other parts of the grammar. In (5.1) the parts of the theory of *phrase* that are relevant for this thesis are given.

(5.1) *Theory of phrase (relevant parts)*

$$:\!\sim\! \mathit{phrase} \quad \rightarrow \quad (:\text{BUILD-PHON PERM ORIGIN} \approx :\text{DTRS ALL-DTRS} \\ \wedge :\text{BUILD-PHON SIGNPHON PHONS} \approx :\text{PHON} \\ \wedge :\text{LP PHON} \approx :\text{PHON})$$

$$(:\!\sim\! \mathit{phrase} \wedge :\text{DTRS} \sim \mathit{head\text{-}comp\text{-}struc}) \quad \rightarrow \quad :\text{LP} \sim \mathit{lp\text{-}hc}$$

$$(:\!\sim\! \mathit{phrase} \wedge :\text{LP} \sim \mathit{lp\text{-}hc}) \\ \rightarrow \quad :\text{LP WS2 BASE} \approx :\text{DTRS ALL-DTRS}$$

as AVM:

$$\begin{bmatrix} \mathit{phrase} \end{bmatrix} \Longrightarrow \begin{bmatrix} \mathit{phrase} & \\ \text{PHON} & \boxed{1} \\ \text{SYNSEM} & \mathit{synsem} \\ \text{BUILD-PHON} & \begin{bmatrix} \mathit{build\text{-}phon} & \\ \text{PERM} & \begin{bmatrix} \text{ORIGIN} \ \boxed{2} \end{bmatrix} \\ \text{SIGNPHON} & \begin{bmatrix} \text{PHONS} \ \boxed{1} \end{bmatrix} \end{bmatrix} \\ \text{LP} & \begin{bmatrix} \mathit{lp\text{-}hc} & \\ \text{PHON} & \boxed{1} \\ \text{WS2 BASE} & \boxed{2} \\ \text{EMBED} & \mathit{embed} \\ \text{LPP2} & \mathit{lpp2} \\ \text{LPP3} & \mathit{lpp3} \end{bmatrix} \\ \text{DTRS} & \begin{bmatrix} \mathit{head\text{-}comp\text{-}struc} \\ \text{ALL-DTRS} \ \boxed{2} \end{bmatrix} \end{bmatrix}$$

The partial theory of phrase in (5.1) leads to the following proposition about constituent order:

PROPOSITION
For each u with $\mathrm{S}(u) = phrase$,

1. if [ALL-DTRS]$u = < d_1, \ldots, d_n >$ and for each $i, 1 \leq i \leq n$, [PHON]$d_i = P_i$, then it is the case that [PHON]u is a permutation of $P_1 \hat{} P_2 \hat{} \ldots \hat{} P_n$.
2. if S([DTRS]u) = *head-comp-struc* then for each v which is an element of [DTRS ALL-DTRS]u and a verb, it is the case that
   (a) if there is a w, such that w is a pronoun and [SYNSEM]w is the subject of v, then for every w', such that w' is a pronoun and a native argument of v and $w' \neq w$, it holds that [PHON]w precedes [PHON]w' on [PHON]u.
   (b) for any pair p, q such that p and q are pronouns and arguments of v it holds that

   if p is a native argument of v, and q is a not a native argument of v, then [PHON]p precedes [PHON]q on [PHON]u.

PROOF

(1)
For each u, if $\mathrm{S}(u) = phrase$,

then it holds that S([BUILD-PHON]u) = *build-phon* and [BUILD-PHON ORIGIN]u = [DTRS ALL-DTRS]u and [BUILD-PHON SIGNPHON PHONS]u = [PHON]u. It follows from the properties of *build-phon*, that [PHON]u is a permutation of the concatenation of the PHON values of the elements of [DTRS ALL-DTRS]u.

Thus (1) holds.

(2)
For each u with $\mathrm{S}(u) = phrase$, if S([DTRS]u) = *head-comp-struc*,

then S([LP]u) = *lp-hc*, [PHON]u = [LP PHON]u and [DTRS ALL-DTRS]u = [LP BASE]u. We know by part (1) of the proof, that [LP PHON]u is a permutation of the PHON values of the elements of [LP BASE]u. Thus the precondition that [LP PHON]u be some permutation the PHON values of the elements on [LP BASE]u for the proposition (4.76) is met for any v which is an element of [DTRS ALL-DTRS]u and a verb. Hence part (2) of (5.1) holds as well. With (1) and (2) the entire proposition holds.

$\blacksquare$

The theory of *phrase* served to establish the links between the presumably universal part of the LP component which is put into the value of the BUILD-PHON attribute on the one hand, and the the part of the LP component that depends on the particular language and constituent structure investigated. The theory given in (5.1) guarantees that the preconditions are met that were needed in the proofs of parts of the LP component. The fact that the theory of *phrase* and the proof of the corresponding proposition are fairly simple shows that the LP component has been built into the overall architecture in a modular way. In (5.2) an AVM for a *phrase* is given with the tags from the theories of *phrase*, *build-phon* and *lp-hc*.

(5.2) *A phrase showing the theories of phrase, build-phon and lp-hc:*

$$
\left[\begin{array}{ll}
\textit{phrase} & \\
\textsc{phon} & \boxed{1}\ \textit{list-of-phons}\\
\textsc{synsem} & \textit{synsem}\\
\textsc{build-phon} & \left[\begin{array}{ll}
\textit{build-phon} & \\
\textsc{perm} & \left[\begin{array}{ll}\textit{perm} & \\ \textsc{origin} & \boxed{2}\ \textit{list-of-signs}\\ \textsc{permuted} & \boxed{3}\ \textit{list-of-signs}\end{array}\right]\\
\textsc{signphon} & \left[\begin{array}{ll}\textit{signphon} & \\ \textsc{signs} & \boxed{3}\ \textit{list-of-signs}\\ \textsc{phons} & \boxed{1}\ \textit{list-of-phons}\end{array}\right]
\end{array}\right]\\
\textsc{lp} & \left[\begin{array}{ll}
\textit{lp-hc} & \\
\textsc{phon} & \boxed{1}\ \textit{list-of-phons}\\
\textsc{ws2} & \left[\begin{array}{ll}\textit{ws2} & \\ \textsc{base} & \boxed{2}\ \textit{list-of-signs}\\ \textsc{embedding embed} & \boxed{4}\ \textit{embed}\end{array}\right]\\
\textsc{embed} & \boxed{4}\ \textit{embed}\\
\textsc{lpp2} & \left[\begin{array}{ll}\textit{lpp2} & \\ \textsc{phon} & \boxed{1}\ \textit{list-of-phons}\\ \textsc{embed} & \boxed{4}\ \textit{embed}\end{array}\right]\\
\textsc{lpp3} & \left[\begin{array}{ll}\textit{lpp3} & \\ \textsc{phon} & \boxed{1}\ \textit{list-of-phons}\\ \textsc{embed} & \boxed{4}\ \textit{embed}\end{array}\right]
\end{array}\right]\\
\textsc{dtrs} & \left[\begin{array}{ll}\textit{head-comp-struc} & \\ \textsc{all-dtrs} & \boxed{2}\ \textit{list-of-signs}\end{array}\right]
\end{array}\right]
$$

The AVM in (5.2) shows the interactions between the theories of some of the sorts given in this thesis. If one compares (5.2) with the AVMs given for the single theories in (2.13) for *build-phon*, (4.76) for *lp-hc* and (5.1) for *phrase*, it becomes evident that the different parts of the present LP component interact in a non-trivial but precisely specified way.

6 Summary

In this thesis an architecture for encoding a theory of constituent order within the HPSG framework developed in Pollard and Sag (1994) and formalized in the logic of King (1994) was proposed. It was shown that it is not necessary to extend the HPSG formalism to formalize the type of rules linguists usually establish about constituent order. It is possible to write LP principles in the same manner as other principles of grammar like the Head Feature Principle and the Subcategorization Principle.

In chapter 1, the logic used throughout this thesis has been presented. We argued that there is no need for relational extensions in the description language, as the logical setup presented in King (1994) provides the syntax and semantics to encode complex relations via recursive sort definitions.

Our theory of linear precedence can be viewed as a modular extension of the classical HPSG sort hierarchy. In chapter 2 a new sort *build-phon* was introduced. This sort appears as the value of some attribute on a *phrase* and guarantees that the phonology of every phrase is indeed a permutation of the phonology of its daughters. It was shown that the theory of *build-phon* together with a theory of the word order domain for LP principles provides a principled formalization of the functions `order-constituents` and `interleave-constituents` which were informally proposed in Pollard and Sag (1987).

The theory of word order domains resides in the assumptions on how to construct the ALL-DTRS list in different constituent structures. We argued that word order domains can be held constant even if the structural analysis is changed. As an example, a flat and a binary branching structural analysis for German have been given in chapter 3.

In chapter 4 the actual LP component was introduced with a new attribute LP on phrases. The LP component itself is constructed in a modular way. There are well defined interfaces for the interaction between the overall *phrase*, the external interfaces, the auxiliary structures and the LP principle attributes.

The linearization regularities formalized at the end of chapter 4 suggest that the necessary formal devices for further LP principles have already been established in the present setup and that only minor modifications of some sorts are required. It is therefore not necessary for a linguist using our idea of linearization to go to great lengths formalizing rather unintuitive new sorts. Instead, empirical work can largely dispense with a discussion of the formal devices that were subject of this thesis.

We did not discuss any data as we were concerned with the question how LP rules could be expressed in the HPSG framework in principle. We consider it a strength of our proposal that it does not depend on any particular linguistic analysis. The LP component developed in chapter 4, however, makes predictions about the kind of linearization regularities expected. Thus, LP principles may refer to the auxiliary structures BASE, EMBED and SCOPOI and be of the simple or recursive type. However, we do not expect that one LP principle needs more than one auxiliary structure or that all principles can be formalized without any such auxiliary structure. Empirical investigation will show whether the proposed LP component is an adequate tool to express linearization regularities.

Even though the emphasis lay on the formal aspects, we did not extend the number of sorts given to a full 'library of sorts'. As mentioned in chapter 1, such a library would contain sorts that can be used as building blocks for the formalization of principles of the grammar. This includes sorts for concatenation, extraction of elements from a list, and others. Nevertheless many of those technical sorts have been presented.

References

Ait-Kaci, H.: 1984, *A Lattice Theoretic Approach to Computation Based on a Calculus of Partially Ordered Type Structures*, PhD thesis, University of Pennsylvania.

Carpenter, B.: 1992, *The Logic of Typed Feature Structures*, Cambridge University Press. Cambridge, Massachusetts, USA.

Carpenter, B.: 1993, Ale - The Attribute Logic Engine - User's Guide, Manuscript. Laboratory for Computational Linguistics, Philosophy Department, Carnegie Mellon University, Pittsburgh, PA 15213.

Dörre, J. and Dorna, M.: 1993, CUF - a formalism for linguistic knowledge representation, *in* J. Dörre (ed.), *Computational aspects of constraint based linguistic descriptions I*, DYANA-2 Deliverable R1.2.A, Universität Stuttgart, pp. 1–22.

Dowty, D.: 1989, On the semantic content of the notion 'thematic role', *in* B. Partee, G. Chierchia and R. Turner (eds), *Properties, Types and Meaning*, Vol. 2, Kluwer. Dordrecht, pp. 69–130.

Engelkamp, J., Erbach, G. and Uszkoreit, H.: 1992, Handling linear precedence constraints by unification, *ACL Proceedings, 23th Annual Meeting*, pp. 201–208.

Frank, A.: n.d., A new HPSG analysis of scrambling and binding, Manuscript. Institut für Maschinelle Sprachverarbeitung, Universität Stuttgart.

Gazdar, G. and Pullum, G. K.: 1981, Subcategorization, Constituent Order, and the Notion "Head", *in* M. Moortgaat, H. v. Hulst and T. Hoekstra (eds), *The Scope of Lexical Rules*, Foris. Dordrecht, pp. 107–122.

Gazdar, G. and Pullum, G. K.: 1982, *Generalized Phrase Structure Grammar: A Theoretical Synopsis*, Indiana University Linguistics Club. Bloomington, Indiana.

Gazdar, G., Klein, E., Pullum, G. K. and Sag, I.: 1985, *Generalized Phrase Structure Grammar*, Harvard University Press. Cambridge Massachusetts.

Gerdemann, D. D.: 1993, Troll - Type Resolution System - Fundamental Principles & User's Guide, Manuscript. Universität Tübingen. Version β.

Hinrichs, E. and Nakazawa, T.: 1989, Review of Word Order and Constituent Structure by Hans Uszkoreit, *Language* **65**, 141–149.

Hinrichs, E. and Nakazawa, T.: 1994, Partial-VP and Split-NP Topicalization in German - An HPSG Analysis, *in* E. Hinrichs, D. Meurers and T. Nakazawa (eds), *Partial-VP and Split-NP Topicalization in German — A HPSG Analysis and its Implementation*, Arbeitspapiere des SFB 340, Nr. 58, Universität Tübingen, pp. 1–46.

Höhfeld, M. and Smolka, G.: 1988, Definite relations over constraint languages, *LILOG Report 53*, IBM Deutschland.

Höhle, T. N.: 1982, Explikationen für "normale Betonung" und "normale Wortstellung", *in* W. Abraham (ed.), *Satzglieder im Deutschen. Vorschläge zur syntaktischen, semantischen und pragmatischen Fundierung*, Tübingen.

Höhle, T. N.: 1986, Der Begriff "Mittelfeld". Anmerkungen über die Theorie der topologischen Felder, *in* A. Schöene (ed.), *Kontroversen alte und neue. Akten des 7. Internationalen Germanistenkongresses Göttingen 1985*, Vol. 3, pp. 329–340.

Höhle, T. N.: 1994, Spurenlose Extraktion, Manuscript. Universität Tübingen. July 7th, 1994.

Jacobs, J.: 1988, Probleme der freien Wortstellung im Deutschen, *Sprache und Pragmatik* **5**, 8–37.

Kepser, S.: 1994, A satisfiability algorithm for a typed feature logic, *Arbeitspapiere des SFB 340 60*, Universität Tübingen.

King, P. J.: 1989, *A logical Formalism for Head-Driven Phrase Structure Grammar*, PhD thesis, University of Manchester.

King, P. J.: 1992, Unification grammars and descriptive formalisms, unpublished lecture notes, Universität Tübingen.

King, P. J.: 1994, An expanded logical formalism for Head-Driven Phrase Structure Grammar, *Arbeitspapiere des SFB 340 59*, Universität Tübingen.

King, P. J. and Götz, T.: 1993, Eliminating the Feature Introduction Condition by Modifying Type Inference, *Arbeitspapiere des SFB 340 31*, Universität Tübingen.

Kiss, T.: 1993, *Infinite Komplementation. Neue Studien zum deutschen Verbum infinitum*, Theorie des Lexikons. Arbeiten des Sonderforschungsbereichs 282 Nr. 42, Bergische Universität Gesamthochschule Wuppertal.

Lenerz, J.: 1977, *Zur Abfolge nominaler Satzglieder im Deutschen*, Verlag Gunter Narr. Tübingen.

Meurers, D.: 1994, On implementing an HPSG theory, *in* E. Hinrichs, D. Meurers and T. Nakazawa (eds), *Partial-VP and Split-NP Topicalization in German — A HPSG Analysis and its Implementation*, Arbeitspapiere des SFB 340, Nr. 58, Universität Tübingen, pp. 47–155.

Meurers, D. and Morawietz, F.: 1994, An Alternative Approach to ID/LP: SUP Grammars, Manuscript. Seminar für Sprachwissenschaft, Universität Tübingen.

Meurers, D. and Richter, F.: 1994, implementation of E. Hinrichs, T. Nakazawa: *"Partial-VP and Split-NP Topicalization in German - An HPSG Analysis"*, SFB 340/B4, July 27th, Universität Tübingen.

Morawietz, F.: 1994, *Formalization and Parsing of typed Unification-Based ID/LP Grammars*, Master's thesis, Universität Tübingen.

Nerbonne, J.: 1994, Partial verb phrases and spurious ambiguities, *in* J. Nerbonne, K. Netter and C. Pollard (eds), *German Grammar in Head-Driven Phrase Structure Grammar*, Vol. 46, CSLI, pp. 109–150.

Parsons, T.: 1990, *Events in the Semantics of English: A Study of Subatomic Semantics*, MIT Press. Cambridge, Massachusetts.

Pollard, C.: 1990, On Head Non-Movement, Manuscript. Carnegie-Mellon University.

Pollard, C.: 1993, Lexical rules as metadescriptions, Manuscript. Paper presented at the Conference "Between Syntax, Semantics, and Logic" at the Universität Stuttgart.

Pollard, C. and Sag, I.: 1987, *Information Based Syntax and Semantics. Vol.1: Fundamentals*, CSLI Lecture Notes 13.

Pollard, C. and Sag, I.: 1994, *Head-Driven Phrase Structure Grammar*, University of Chicago Press.

Pollard, C. J. and Moshier, M. D.: 1990, Unifying partial descriptions of sets, *in* P. P. Hanson (ed.), *Information, Language, and Cognition*, University of British Columbia Press. Vancouver, pp. 285–322.

Pollard, C., Levine, R. and Kasper, R.: 1994, Studies in Constituent Ordering. Toward a Theory of Linearization in Head-Driven Phrase Structure Grammar, Manuscript. Research Proposal to the National Science Foundation, Ms. Ohio State University.

Primus, B.: 1989, Parameter der Herrschaft: Reflexivpronomina im Deutschen, *Zeitschrift für Sprachwissenschaft* **8**, 53–88.

Reape, M.: 1989, A logical treatment of semi-free word order and bounded discontinuous constituency, *Proceedings of the 4th Conference of the European Chapter of the ACL*, pp. 103–110.

Reape, M.: 1990, Getting things in order, *Proceedings of the Symposium on Discontinuous Constituency*, Institute for Language Technology and Artificial Intelligence.

Reape, M.: 1992, *A Formal Theory of Word Order: A Case Study of West Germanic*, PhD thesis, University of Edinburgh.

Reape, M.: 1994, Domain union and word order variation in german, *in* J. Nerbonne, K. Netter and C. Pollard (eds), *German Grammar in Head-Driven Phrase Structure Grammar*, Vol. 46, CSLI, pp. 151–197.

Reis, M.: 1987, Die Stellung der Verbargumente im Deutschen. Stilübungen zum Grammatik:Pragmatik-Verhältnis, *in* I. Rosengren (ed.), *Sprache und Pragmatik*, Lunder Symposium 1986, pp. 139–177.

Rentier, G.: n.d., Dutch cross serial dependencies in HPSG, Manuscript. Institute for Language Technology and Artificial Intelligence, Tilburg.

Sag, I.: 1987, Grammatical hierarchy and linear precedence, *in* G. Huck and A. Ojeda (eds), *Discontinuous Constituency*, Vol. 20 of *Syntax and Semantics*, New York: Academic Press.

Shieber, S. M.: 1984, Direct Parsing of ID/LP Grammars, *Linguistics and Philosophy* **7**, 135–154.

Uszkoreit, H.: 1986a, Constraints on Order, *CSLI-Report 86-46*, CSLI.

Uszkoreit, H.: 1986b, Linear Precedence in Discontinuous Constituents: Complex Fronting in German, *CSLI-Report 86-47*, CSLI.

A Appendix

In order to provide an easy orientation for this thesis, the present appendix contains all sorts and definitions used in the formal parts (chapter 2, sections 4.3 – 4.6). In A.1 the appropriateness conditions of the newly introduced sorts are given. A.2 contains the corresponding theories in term notation. In these two parts of the appendix, the sorts are ordered alphabetically. The number of the exampleand the page of occurrence in the main text are given. In A.3 the definitions needed in the proofs are given in the order of occurrence in the text.

A.1 Appropriateness Conditions

The sort alist (4.54), page 155

alist	BIG	*list-of-phons*		
	SMALL	*list-of-phons*		
	PREF	*list-of-phons*		
	alist0	BIG	*list-of-phons*	
		SMALL	*elist*	
		PREF	*elist*	
	alist1	BIG	*nelist-of-phons*	
		SMALL	*nelist-of-phons*	
		PREF	*nelist-of-phons*	
		TEST	*in-and-out*	
		REC	*alist*	

The sort append-of-phons (2.9), page 40

append-of-phons	RES	*list-of-phons*	
	LEFT	*list-of-phons*	
	RIGHT	*list-of-phons*	
finished-append-of-phons		RES	*list-of-phons*
		LEFT	*elist*
		RIGHT	*list-of-phons*
unfinished-append-of-phons		RES	*list-of-phons*
		LEFT	*nelist-of-phons*
		RIGHT	*list-of-phons*
		REC	*append-of-phons*

The sort blist (4.62), page 165

blist	BIG	*list-of-phons*	
	SMALL	*list-of-phons*	
	blist0	BIG	*list-of-phons*
		SMALL	*elist*
	blist1	BIG	*list-of-phons*
		SMALL	*nelist-of-phons*
		TEST	*in-and-out*
		REC	*blist*

The sort build-phon (2.3), page 34

build-phon	PERM	*perm*
	SIGNPHON	*signphon*

The sort check (4.24), page 112

check	TO-CHECK	*list-of-lists-of-synsems*	
	DIFFL	*list-of-lists-of-synsems*	
	check0	TO-CHECK	*elist*
		DIFFL	*elist*
	check1	TO-CHECK	*nelist-of-lists-of-synsems*
		DIFFL	*list-of-lists-of-synsems*
		REC	*check0*
	check2	TO-CHECK	*nelist-of-lists-of-synsems*
		DIFFL	*list-of-lists-of-synsems*
		TEST	*subset*
		REC	*check*

The sort compsverb (4.22), page 110

compsverb	VERBL	*list-of-signs*	
	COMPSL	*list-of-lists-of-synsems*	
	compsverb0	VERBL	*elist*
		COMPSL	*elist*
	compsverb1	VERBL	*nelist-of-signs*
		COMPSL	*nelist-of-lists-of-synsems*
		REC	*compsverb*

The sort *embed* (4.35), page 129

embed

embed0

embed1	VERB	*sign*
	ARGS	*list-of-signs*
	DEEPER	*embed*

The sort *embedding* (4.30), page 123

embedding	BASE	*list-of-signs*
	VERBL	*list-of-signs*
	ARGS	*list-of-lists-of-synsems*
	EMBED	*embed*

embedding0	BASE	*list-of-signs*
	VERBL	*elist*
	ARGS	*elist*
	EMBED	*embed0*

embedding1	BASE	*list-of-signs*
	VERBL	*nelist-of-signs*
	ARGS	*list-of-lists-of-synsems*
	EMBED	*embed1*
	MAKE-LOS	*lsignsynsems*
	REC	*embedding*

The sort *find-pron* (4.43), page 142

find-pron	ARGS	*list-of-signs*
	PRONS	*list-of-signs*

find-pron0	ARGS	*elist*
	PRONS	*elist*

find-pron1	ARGS	*nelist-of-signs*
	PRONS	*list-of-signs*
	REC	*find-pron*

The sort *find-s/ns* (4.42), page 141

find-s/ns	VERB	*sign*
	ARGS	*list-of-signs*
	SU	*signphon*
	NSU	*signphon*
	FIND-PRON	*findpron*
	SEP-PRON	*separate-pron*

The sort *find-verbs* (4.18), page 105

find-verbs	BASE	*list-of-signs*
	VERBL	*list-of-signs*

find-verbs0	BASE	*elist*
	VERBL	*elist*
find-verbs1	BASE	*nelist-of-signs*
	VERBL	*list-of-signs*
	REC	*find-verbs*
find-verbs2	BASE	*nelist-of-signs*
	VERBL	*list-of-signs*
	REC	*find-verbs*

The sort *in-and-out* (4.55), page 156

in-and-out	SET	*list-of-phons*
	ELE	*phon*
	NSET	*list-of-phons*

in-and-out0	SET	*elist*
	ELE	*phon*
	NSET	*elist*
in-and-out1	SET	*nelist-of-phons*
	ELE	*phon*
	NSET	*list-of-phons*
in-and-out2	SET	*nelist-of-phons*
	ELE	*phon*
	NSET	*list-of-phons*
	REC	*in-and-out*

The sort *is-in-ss* (4.32), page 124

is-in-ss	SET	*nelist-of-signs*
	ELESYNSEM	*synsem*
	ELESIGN	*sign*

is-in-ss0	SET	*nelist-of-signs*
	ELESYNSEM	*synsem*
	ELESIGN	*sign*
is-in-ss1	SET	*nelist-of-signs*
	ELESYNSEM	*synsem*
	ELESIGN	*sign*
	REC	*is-in-ss*

The sort lp-hc prelimminary version (4.10), page 95

lp-hc	PHON	*list-of-phons*
	WS1	*ws1*
	WS2	*ws2*
	WS3	*ws3*
	WS4	*ws4*
	⋮	
	WSn	*ws-n*
	BASE	*list-of-signs*
	EMBED	*embed*
	SCOPOI	*list-of-signs*
	AUX4	*aux4*
	⋮	
	AUXn	*aux-n*
	LPP1	*lpp1*
	LPP2	*lpp2*
	LPP3	*lpp3*
	⋮	
	LPPm	*lpp-m*

The sort lp-hc (revised) (4.11), page 96

lp-hc	PHON	*list-of-phons*
	WS2	*ws2*
	EMBED	*embed*
	LPP2	*lpp2*
	LPP3	*lpp3*

The sort lpp2 (4.40), page 139

lpp2	PHON	*list-of-phons*
	EMBED	*embed*

lpp2_0	PHON	*list-of-phons*
	EMBED	*embed0*

lpp2_1	PHON	*list-of-phons*
	EMBED	*embed1*
	FIND-S/NS	*find-s/ns*
	TEST	*prec*
	REC	*lpp2*

The sort *lpp3* (4.74), page 182

lpp3	PHON	*list-of-phons*
	EMBED	*embed*
	CUR-PRONS	*signphon*
	DEEP-PRONS	*signphon*

lpp3_0	PHON	*list-of-phons*
	EMBED	*embed0*
	CUR-PRONS	*signphon0*
	DEEP-PRONS	*signphon0*
lpp3_1	PHON	*list-of-phons*
	EMBED	*embed1*
	CUR-PRONS	*signphon*
	DEEP-PRONS	*signphon*
	FIND-CUR	*find-pron*
	FIND-DEEP	*append-of-signs*
	TEST	*prec*
	REC	*lpp3*

The sort *lsignsynsems* (4.31), page 124

lsignsynsems	BASE	*list-of-signs*
	LSIGN	*list-of-signs*
	LSYNSEM	*list-of-synsems*

lsignsynsems0	BASE	*list-of-signs*
	LSIGN	*elist*
	LSYNSEM	*elist*
lsignsynsems1	BASE	*list-of-signs*
	LSIGN	*nelist-of-signs*
	LSYNSEM	*nelist-of-synsems*
	TEST	*is-in-ss*
	REC	*lsignsynsems*

The sort *order-embed* (4.21), page 109

order-embed	PERMVERBS	*perm*
	CHECK	*check*
	COMPSVERB	*compsverb*

The sort perm (2.4), page 35

perm	ORIGIN	*list-of-signs*
	PERMUTED	*list-of-signs*

perm0	ORIGIN	*elist*
	PERMUTED	*elist*
perm1	ORIGIN	*nelist-of-signs*
	PERMUTED	*nelist-of-signs*
	TEST	*take-out*
	REC	*perm*

The sort prec (4.53), page 155

prec	C	*list-of-phons*
	A	*list-of-phons*
	B	*list-of-phons*
	ARG1	*alist*
	ARG2	*blist*

The sort separate-pron (4.47), page 146

separate-pron	PRONS	*list-of-signs*
	VERB	*sign*
	SU	*list-of-signs*
	NSU	*list-of-signs*

separate-pron0	PRONS	*elist*
	VERB	*sign*
	SU	*elist*
	NSU	*elist*
separate-pron1	PRONS	*nelist-of-signs*
	VERB	*sign*
	SU	*nelist-of-signs*
	NSU	*list-of-signs*
	REC	*separate-pron*
separate-pron2	PRONS	*nelist-of-signs*
	VERB	*sign*
	SU	*list-of-signs*
	NSU	*nelist-of-signs*
	REC	*separate-pron*

The sort signphon (2.8), page 40

signphon	SIGNS	*list-of-signs*
	PHONS	*list-of-phons*

signphon0	SIGNS	*elist*
	PHONS	*elist*
signphon1	SIGNS	*nelist-of-signs*
	PHONS	*list-of-phons*
	APPEND	*append-of-phons*
	REC	*signphon*

The sort subset (4.25), page 113

subset	BIG	*list-of-synsems*
	SMALL	*list-of-synsems*
	DIFF	*list-of-synsems*

subset0	BIG	*list-of-synsems*
	SMALL	*elist*
	DIFF	*list-of-synsems*
subset1	BIG	*nelist-of-synsems*
	SMALL	*nelist-of-synsems*
	DIFF	*list-of-synsems*
	TAKE-OUT	*take-out''*
	REC	*subset*

The sort take-out (2.5), page 36

take-out	SET	*nelist-of-signs*
	ELE	*sign*
	NSET	*list-of-signs*

take-out0	SET	*nelist-of-signs*
	ELE	*sign*
	NSET	*list-of-signs*
take-out1	SET	*nelist-of-signs*
	ELE	*sign*
	NSET	*list-of-signs*
	REC	*take-out*

The sort ws2 (4.16), page 102

ws2	FIND-VERBS	*find-verbs*
	ORDER	*order-embed*
	EMBEDDING	*embedding*
	BASE	*list-of-signs*

A.2 Theories

The sort alist (4.59), page 161

$$:\sim alist1 \quad \to \quad (\text{:TEST SET} \approx \text{:SMALL} \wedge \text{:TEST ELE} \approx \text{:BIG FIRST} \wedge \text{:PREF FIRST} \approx \text{:BIG FIRST} \wedge \text{:PREF REST} \approx \text{:REC PREF} \wedge \text{:REC BIG} \approx \text{:BIG REST} \wedge \text{:REC SMALL} \approx \text{:TEST NSET})$$

The sort append-of-phons (2.10), page 41

$$:\sim \textit{finished-append-of-phons} \quad \to \quad (\text{:RES} \approx \text{:RIGHT})$$

$$:\sim \textit{unfinished-append-of-phons} \quad \to \quad (\text{:RES FIRST} \approx \text{:LEFT FIRST} \wedge \text{:REC RES} \approx \text{:RES REST} \wedge \text{:REC LEFT} \approx \text{:LEFT REST} \wedge \text{:REC RIGHT} \approx \text{:RIGHT})$$

The sort blist (4.63), page 165

$$:\sim blist1 \quad \to \quad (\text{:TEST SET} \approx \text{:BIG} \wedge \text{:TEST ELE} \approx \text{:SMALL FIRST} \wedge \text{:TEST NSET} \approx \text{:TEST SET} \wedge \text{:REC BIG} \approx \text{:BIG} \wedge \text{:REC SMALL} \approx \text{:SMALL REST})$$

The sort build-phon (2.13), page 46

$$:\sim \textit{build-phon} \quad \to \quad (\text{:PERM PERMUTED} \approx \text{:SIGNPHON SIGNS})$$

The sort check (4.27), page 116

$$:\sim check1 \quad \to \quad (\text{:TO-CHECK REST} \sim \textit{elist} \wedge \text{:DIFFL FIRST} \approx \text{:TO-CHECK FIRST} \wedge \text{:DIFFL REST} \approx \text{:REC DIFFL} \wedge \text{:REC TO-CHECK} \approx \text{:TO-CHECK REST})$$

$:\!\sim\!check2 \quad \rightarrow \quad (\text{:TO-CHECK FIRST} \approx \text{:TEST BIG}$
$\wedge\ \text{:TO-CHECK REST FIRST} \approx \text{:TEST SMALL}$
$\wedge\ \text{:DIFFL FIRST} \approx \text{:TEST DIFF}$
$\wedge\ \text{:DIFFL REST} \approx \text{:REC DIFFL}$
$\wedge\ \text{:REC TO-CHECK} \approx \text{:TO-CHECK REST})$

The sort compsverb (4.23), page 110

$:\!\sim\!compsverb1 \quad \rightarrow$
$(\text{:VERBL FIRST SYNSEM LOC CAT VAL COMPS} \approx \text{:COMPSL FIRST}$
$\wedge\ \text{:REC VERBL} \approx \text{:VERBL REST}$
$\wedge\ \text{:REC COMPSL} \approx \text{:COMPSL REST})$

The sort embedding (4.36), page 130

$:\!\sim\!embedding1 \quad \rightarrow \quad (\text{:MAKE-LOS BASE} \approx \text{:BASE}$
$\wedge\ \text{:MAKE-LOS LSYNSEM} \approx \text{:ARGS FIRST}$
$\wedge\ \text{:EMBED VERB} \approx \text{:VERBL FIRST}$
$\wedge\ \text{:EMBED ARGS} \approx \text{:MAKE-LOS LSIGN}$
$\wedge\ \text{:EMBED DEEPER} \approx \text{:REC EMBED}$
$\wedge\ \text{:REC BASE} \approx \text{:BASE}$
$\wedge\ \text{:REC VERBL} \approx \text{:VERBL REST}$
$\wedge\ \text{:REC ARGS} \approx \text{:ARGS REST})$

The sort find-pron (4.44), page 142

$(:\!\sim\!find\text{-}pron1$
$\wedge\ \text{:ARGS FIRST SYNSEM LOC CONTENT} \sim pron)$
$\rightarrow \quad (\text{:PRONS FIRST} \approx \text{:ARGS FIRST}$
$\wedge\ \text{:PRONS REST} \approx \text{:REC PRONS}$
$\wedge\ \text{:REC ARGS} \approx \text{:ARGS REST})$

$(:\!\sim\!find\text{-}pron1$
$\wedge\ \neg(\text{:ARGS FIRST SYNSEM LOC CONTENT} \sim pron))$
$\rightarrow \quad (\text{:PRONS} \approx \text{:REC PRONS}$
$\wedge\ \text{:REC ARGS} \approx \text{:ARGS REST})$

The sort find-s/ns (4.50), page 151

$:\sim$*find-s/ns* $\rightarrow$ $($:FIND-PRON ARGS $\approx$:ARGS
$\wedge$:SEP-PRON PRONS $\approx$:FIND-PRON PRONS
$\wedge$:SEP-PRON VERB $\approx$:VERB
$\wedge$:SU SIGNS $\approx$:SEP-PRON SU
$\wedge$:NSU SIGNS $\approx$:SEP-PRON NSU$)$

The sort find-verbs (4.19), page 105

$:\sim$*find-verbs1* $\rightarrow$
$($:BASE FIRST SYNSEM LOC CAT HEAD $\sim$ *verb*
$\wedge$:BASE FIRST $\sim$ *word*
$\wedge$:VERBL FIRST $\approx$:BASE FIRST
$\wedge$:REC BASE $\approx$:BASE REST
$\wedge$:REC VERBL $\approx$:VERBL REST$)$

$:\sim$*find-verbs2* $\rightarrow$
$(\neg($:BASE FIRST SYNSEM LOC CAT HEAD $\sim$ *verb*
$\wedge$:BASE FIRST $\sim$ *word*$)$
$\wedge$:REC BASE $\approx$:BASE REST
$\wedge$:REC VERBL $\approx$:VERBL$)$

The sort in-and-out (4.56), page 157

$:\sim$*in-and-out1* $\rightarrow$ $($:SET FIRST $\approx$:ELE
$\wedge$:SET REST $\approx$:NSET$)$

$:\sim$*in-and-out2* $\rightarrow$ $(\neg($:SET FIRST $\approx$:ELE$)$
$\wedge$:NSET FIRST $\approx$:SET FIRST
$\wedge$:REC SET $\approx$:SET REST
$\wedge$:REC ELE $\approx$:ELE
$\wedge$:REC NSET $\approx$:NSET REST$)$

The sort is-in-ss (4.33), page 125

$:\sim$*is-in-ss0* $\rightarrow$ $($:SET FIRST SYNSEM $\approx$:ELESYNSEM
$\wedge$:SET FIRST $\approx$:ELESIGN$)$

$:\sim$*is-in-ss1* $\rightarrow$ $(\neg($:SET FIRST SYNSEM $\approx$:ELESYNSEM$)$
$\wedge$:REC SET $\approx$:SET REST
$\wedge$:REC ELESYNSEM $\approx$:ELESYNSEM
$\wedge$:REC ELESIGN $\approx$:ELESIGN$)$

The sort lp-hc (4.76), page 188

$:\sim lp\text{-}hc \quad \rightarrow$ (:EMBED ≈ :WS2 EMBEDDING EMBED
∧ :LPP2 PHON ≈ :PHON
∧ :LPP2 EMBED ≈ :EMBED
∧ :LPP3 PHON ≈ :PHON
∧ :LPP3 EMBED ≈ :EMBED)

The sort lpp2 (4.72), page 177

$:\sim lpp2\_1 \quad \rightarrow$ (:EMBED VERB ≈ :FIND-S/NS VERB
∧ :EMBED ARGS ≈ :FIND-S/NS ARGS
∧ :TEST C ≈ :PHON
∧ :TEST A ≈ :FIND-S/NS SU PHONS
∧ :TEST B ≈ :FIND-S/NS NSU PHONS
∧ :REC PHON ≈ :PHON
∧ :REC EMBED ≈ :EMBED DEEPER)

The sort lpp3 (4.75), page 183

$:\sim lpp3\_1 \quad \rightarrow$ (:FIND-CUR ARGS ≈ :EMBED ARGS
∧ :CUR-PRONS SIGNS ≈ :FIND-CUR PRONS
∧ :DEEP-PRONS SIGNS ≈ :FIND-DEEP RES
∧ :REC CUR-PRONS SIGNS ≈ :FIND-DEEP LEFT
∧ :REC DEEP-PRONS SIGNS ≈ :FIND-DEEP RIGHT
∧ :TEST C ≈ :PHON
∧ :TEST A ≈ :CUR-PRONS PHONS
∧ :TEST B ≈ :DEEP-PRONS PHONS
∧ :REC PHON ≈ :PHON
∧ :REC EMBED ≈ :EMBED DEEPER)

The sort lsignsynsems (4.34), page 127

$:\sim lsignsynsems1 \quad \rightarrow$ (:BASE ≈ :TEST SET
∧ :LSYNSEM FIRST ≈ :TEST ELESYNSEM
∧ :LSIGN FIRST ≈ :TEST ELESIGN
∧ :REC BASE ≈ :BASE
∧ :REC LSIGN ≈ :LSIGN REST
∧ :REC LSYNSEM ≈ :LSYNSEM REST)

The sort order-embed (4.28), page 119

$:\!\sim\! \mathit{order\text{-}embed} \rightarrow$
$(\text{:PERMVERBS PERMUTED} \approx \text{:COMPSVERB VERBL}$
$\wedge\ \text{:COMPSVERB COMPSL} \approx \text{:CHECK TO-CHECK})$

The sort phrase (relevant parts) (5.1), page 237

$:\!\sim\! \mathit{phrase} \rightarrow (\text{:BUILD-PHON PERM ORIGIN} \approx \text{:DTRS ALL-DTRS}$
$\wedge\ \text{:BUILD-PHON SIGNPHON PHONS} \approx \text{:PHON}$
$\wedge\ \text{:LP PHON} \approx \text{:PHON})$

$(:\!\sim\! \mathit{phrase} \wedge \text{:DTRS} \sim \mathit{head\text{-}comp\text{-}struc}) \rightarrow \text{:LP} \sim \mathit{lp\text{-}hc}$

$(:\!\sim\! \mathit{phrase} \wedge \text{:LP} \sim \mathit{lp\text{-}hc})$
$\rightarrow \text{:LP WS2 BASE} \approx \text{:DTRS ALL-DTRS}$

The sort perm (2.7), page 38

$:\!\sim\! \mathit{perm1} \rightarrow (\text{:ORIGIN} \approx \text{:TEST SET}$
$\wedge\ \text{:PERMUTED FIRST} \approx \text{:TEST ELE}$
$\wedge\ \text{:REC PERMUTED} \approx \text{:PERMUTED REST}$
$\wedge\ \text{:REC ORIGIN} \approx \text{:TEST NSET})$

The sort prec (4.65), page 168

$:\!\sim\! \mathit{prec} \rightarrow (\text{:C} \approx \text{:ARG1 BIG}$
$\wedge\ \text{:A} \approx \text{:ARG1 SMALL}$
$\wedge\ \text{:B} \approx \text{:ARG2 SMALL}$
$\wedge\ \text{:ARG1 PREF} \approx \text{:ARG2 BIG})$

The sort separate-pron (4.48), page 147

$:\!\sim\! \mathit{separate\text{-}pron1}$
$\rightarrow (\text{:VERB SYNSEM LOC CAT VAL SUBJ FIRST} \approx \text{:PRONS FIRST SYNSEM}$
$\wedge\ \text{:SU FIRST} \approx \text{:PRONS FIRST}$
$\wedge\ \text{:SU REST} \sim \mathit{elist}$
$\wedge\ \text{:NSU} \approx \text{:REC NSU}$
$\wedge\ \text{:PRONS REST} \approx \text{:REC PRONS})$

:$\sim$*separate-pron2*
$\rightarrow$ ((¬(:VERB SYNSEM LOC CAT VAL SUBJ FIRST ≈ :PRONS FIRST SYNSEM)
∨ :VERB SYNSEM LOC CAT VAL SUBJ ∼ *elist*)
∧ :SU ≈ :REC SU
∧ :NSU FIRST ≈ :PRONS FIRST
∧ :NSU REST ≈ REC NSU
∧ :PRONS REST ≈ :REC PRONS)

The sort *sign* (cf. *phrase*)

The sort signphon (2.11), page 42

:$\sim$*signphon1* → (:APPEND RES ≈ :PHONS
∧ APPEND LEFT ≈ :SIGNS FIRST PHON
∧ APPEND RIGHT ≈ :REC PHONS
∧ REC SIGNS ≈ :SIGNS REST)

The sort subset (4.26), page 113

:$\sim$*subset0* → (:BIG ≈ :DIFF)

:$\sim$*subset1* → (:TAKE-OUT SET ≈ :BIG
∧ :TAKE-OUT ELE ≈ :SMALL FIRST
∧ :REC BIG ≈ :TAKE-OUT NSET
∧ :REC SMALL ≈ :SMALL REST
∧ :REC DIFF ≈ :DIFF)

The sort take-out (2.6), page 36

:$\sim$*take-out0* → (:SET FIRST ≈ :ELE
∧ :SET REST ≈ :NSET)

:$\sim$*take-out1* → (¬ (:SET FIRST ≈ :ELE)
∧ :SET FIRST ≈ :NSET FIRST
∧ :REC SET ≈ :SET REST
∧ :REC ELE ≈ :ELE
∧ :REC NSET ≈ :NSET REST)

The sort ws2 (4.37), page 133

$$
\begin{aligned}
:\!\sim ws2 \quad \rightarrow \quad & \\
& (\textsc{:find-verbs verbl} \approx \textsc{:order permverbs origin} \\
& \wedge\ \textsc{:order compsverb verbl} \approx \textsc{:embedding verbl} \\
& \wedge\ \textsc{:order check diffl} \approx \textsc{:embedding args} \\
& \wedge\ \textsc{:base} \approx \textsc{:find-verbs base} \\
& \wedge\ \textsc{:base} \approx \textsc{:embedding base})
\end{aligned}
$$

A.3 Definitions

Def 1 *Verbs* (page 106)

v **is a verb** iff
S([SYNSEM LOC CAT HEAD]u) $\sqsubseteq$ *verb*.

Def 2 *membership in a list* (page 106)

for any object o of a given sort *sort*, and any finite list $l = < o_1, \ldots, o_n >$ of objects of the sort *sort*,
o **is an element of** l iff
there is an $i, 1 \leq i \leq n$ such that $o = o_i$.

Def 3 *List of complements* (page 120)

c **is the list of complements of** v iff
$c =$ [SYNSEM LOC CAT VAL COMPS]v.

Def 4 *Pronouns* (page 143)

p **is a pronoun** iff
S([SYNSEM LOC CONTENT]p) $\sqsubseteq$ *pron*

Def 5 *Subject* (page 148)

For any objects s and v:
s **is the subject of** v iff

1. S([SYNSEM LOC CAT HEAD]v) $=$ *verb* and
2. the path SYNSEM LOC CAT VAL SUBJ FIRST is defined on v
3. [SYNSEM LOC CAT VAL SUBJ FIRST]$v = s$

Def 6 *first occurrence on a list* (page 158)

For any object o and any list $l = < o_1, \ldots, o_n >$,
$o_i, 1 \leq i \leq n$, **is the first occurrence of** o **on** l, iff

1. o is an element of l and
2. $o_i = o$ and
3. there is no $j, 1 \leq j \leq n$, with $j < i$ such that $o_j = o$.

Def 7 *first-o difference* (page 158)

For any object o and any two lists l, h, with $l = < o_1, \ldots, o_n >$
l **is first-o different from** h iff

if o is an element of l and o_i is the first occurrence of o in l

then $h = < o_1, \ldots, o_{i-1}, o_{i+1}, \ldots, o_n >$

else $h = < o_1, \ldots, o_n >$

Def 8 *Precedence (lists)* (page 168)

Given a list $l = < o_1, \ldots, o_n >$ and two lists $h = < a_1, \ldots, a_m >$ and h' $= < a'_1, \ldots, a'_{m'} >$ then
(all of) h **precedes (all of)** h' **on** l iff
there is an $i, 1 \leq i \leq n$ such that all elements of h are on $< o_1, \ldots, o_i >$ and all elements of h' are on $< o_{i+1}, \ldots, o_n >$.

Def 9 *Current Pronoun of* (page 179)

Let u be any object with $\mathrm{S}(u) \sqsubseteq$ *embed*. Assume that p is a pronoun.
p **is a current pronoun of** u iff
p is an element of $[\textsc{args}]u$.

Def 10 *Deeper Pronoun of* (page 179)

p **is a deeper pronoun of** u iff

1. $[\textsc{deeper args}]u = u'$ and p is a current pronoun of u' or
2. $[\textsc{deeper}]u = u'$ and p is a deeper pronoun of u'.

Def 11 *Pronoun of* (page 179)

p **is a pronoun of** u iff

1. p is a current pronoun of u or
2. p is a deeper pronoun of u.

Def 12 *Same Level Pronouns of* (page 180)

Let p, q be pronouns.
p and q **are same level pronouns of** u iff

1. p and q are current pronouns or
2. $[\textsc{deeper}]u = u'$ and p and q are same level pronouns of u'.

Def 13 *Deeper Pronoun than* (page 184)

Let u be an object with $\mathrm{S}(u) \sqsubseteq$ *embed* and p and q be pronouns.
p **is a deeper pronoun of** u **than** q iff

1. q is a current pronoun of u and p is a deeper pronoun of u or
2. $[\textsc{deeper}]u = u'$ and p is a deeper pronoun of u' than q.

Def 14 *Argument* (page 189)

u **is an argument of** v iff
$[\textsc{synsem}]u$ is an element of $[\textsc{synsem loc cat val comps}]v$.

Def 15 *Native Argument* (page 189)

u **is a native argument of** v iff

1. u is an argument of v and
2. there is no v' which is a verb and an argument of v such that u is an argument of v'.