An Alternative Approach to ID/LP SUP Grammars*

Detmar Meurers and Frank Morawietz†
Seminar für Sprachwissenschaft
Eberhard–Karls–Universität Tübingen
Kleine Wilhelmstraße 113
72074 Tübingen
Germany

March 21, 1994

Abstract

Since Gazdar et al. 1985 the ID/LP format has been the most prominent approach aiming to separate the order of constituents from constituent structure. In this paper we want to view some linguistic, formal and processing properties of LP statements and point out problems inherent to the approach. We then propose an alternative based on allowing reorderings on a given linear order instead of ruling out orders on unconstrained domains. The approach is couched in a typed feature logic à la King 1992. Linguistic motivation is given in HPSG related terms (Pollard and Sag in press).

Keywords

linguistic types of LP, properties of LP, transitivity of LP, processing ID/LP format, SUP format

1 Introduction

The ID/LP format has been a very useful formalism for expressing grammars allowing to separate the information on immediate dominance from the one on linear precedence. This enables the grammar writer to express generalizations about linearization. On the other hand ID/LP grammars do not "produce" correct structures, as was the case for simple phrase structure grammars, but rather work via having LP statements ruling out some of the many structures premitted by the immediate dominance rules. This can be seen to be disadvantageous in two respects. From a theoretical point of view checking LP statments requires negation to be part of the underlying logic. Approaching LP from a processing perspective opens up serious efficiency problems. Several proposals have been made to deal with the task of parsing grammars written in ID/LP format, all of

*The research presented in this paper was partly sponsored by Teilprojekt B4 "Constraints on Grammar for Efficient Generation" of the Sonderforschungsbereich 340 "Sprachtheoretische Grundlagen für die Computerlinguistik" of the Deutsche Forschungsgemeinschaft. We want to thank Thilo Götz for helpful discussion and advice. Needless to say, all errors and infelicities that remain are our own.

†Email: {dm,frank}@sns.neuphilologie.uni-tuebingen.de

which are faced which a number of problems. We therefore propose an alternative format, called SUP, which avoids some of the problems of ID/LP.

The paper proceeds as follows: In section 2 we view the different types of linearization constraints used in Linguistics. Section 3 consists of an overview of work dealing with processing ID/LP grammars, focussing on what kind of properties of LP are dealt with. We then turn to the characteristics and problems of LP statements as a formal system: In section 4.1 the linguistic motivation, theoretic and processing properties of transitivity of LP is discussed. This leads us to the formulation of a denotational semantics for LP statments, showing that such a formalization does not yield the linguistically desired type of transitivity. In section 4.2 we show that checking LP statments involves negation, which is problematic for common feature logics. The last section on LP is concerned with clashing LP statements and undesired results in the computation of the transitive closure. The final sections of this paper are then used to devellop an alternative formalization of linear precedence based on permitting additional orders on the trees licenced by phrase structure rules.

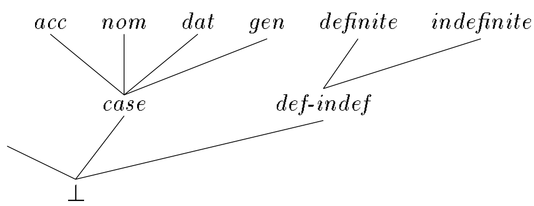
Since in this paper we are concerned with computational linguistic issues, the examples given serve to motivate and illustrate the properties of the formalisms in which generalizations about linear order can be expressed. In particular the examples are not intended to be part of a linguistic analysis of some natural language.

We take feature terms to denote sets of feature structures which model linguistic objects.[1] Since we are not concerned with feature logic as such, we do not define such a logic in this paper, but assume a system along the lines of King 1992.

Throughout the paper the term *LP statement* is used synonymous with *LP rule* to denote a relation between a pair of *LP elements* and will be noted as "$F \prec G$"[2]. The notation $[\![F]\!]$ is employed as being the set of feature-structures denoted by the feature term F.[3]

For several of the examples given below we assume the signature in (1).

(1) $\;{}_{np}\begin{bmatrix} \text{CASE} & \textit{case} \\ \text{DEF} & \textit{def-indef} \end{bmatrix}$



[1]We will use *description* as synonymous with feature term and *describe* as meaning the same as denote.

[2]A formal definition of this relation is given in section 4.1.2

[3]We use the uppercase letters F, G, H, I for feature terms, A, B, C, D for daughters (feature terms as "used in a tree"), X, Y as Attributes, lowercase letters $a - g, u$ for feature-structures, and t, r, s as types.

2 LP and linguistics

Several types of LP statements seem to play a role in linguistics. They differ in respect to what information they require to apply, and to whether a relative or absolute order is expressed.

2.1 Type A: Ordering according to sign inherent properties

Relative order: This type of LP statement is the most straightforward one. Pairs of feature structures are forced into an order depending on their inherent linguistic properties. An example encoding a regularity of word order in the German *Mittelfeld*[4] noted in Lenerz 1977 is the following LP statement forcing indefinite dative NPs to occur after definite accusative NPs:

(3) $$\begin{bmatrix} \text{CASE} & acc \\ \text{DEF} & def \end{bmatrix}_{np} \prec \begin{bmatrix} \text{CASE} & dat \\ \text{DEF} & indef \end{bmatrix}_{np}$$

Slightly more formally we can classify relative type A LP statements as being of the form:

(4) $F \prec G$, with F and G feature terms describing linguistic *signs*.

Absolute order: While in most cases the position of an element is specified relative to another element, sometimes a further generalization is possible via specifying absolute positions of elements. Such a generalization is captured in the LP statement (5) expressing that a personal pronoun always occurs first in the German Mittelfeld[5]:

(5) $pers\text{–}pron \prec \Gamma$

We will use Γ throughout the paper to describe all feature structures different from those denoted by the other lp element; in the above example, all feature structures that are not personal pronouns.[6]

Since the purpose of absolute LP statements is to capture a generalization over relative ones, it is clear that it is possible to avoid absolute statements (and loose the generalization) by specifying all possible sisters possibly occurring in the domain of LP application.

In general absolute type A LP statements are of the form:

(6) $\Gamma \prec F$ or $F \prec \Gamma$, where
F is a feature term describing a linguistic sign
Γ describes any feature structure not described by F

[4] Roughly speaking, the term *Mittelfeld* – for main clauses with compound verb forms – refers to the elements between a finite verb, occuring as first or second constituent, and the non–finite verbs. In (2) for example the Mittelfeld is made up of two elements "dem Jungen den Ball".

(2) Sarah hat dem Jungen den Ball gegeben.
Sarah has given the ball to the boy.

[5] In order for this LP statement to work correctly, the domain of application (i.e. the local tree, head domain, ...) needs to be equivalent to the domain associated with the term "Mittelfeld". See section 4.1.2 for a short discussion of the domain issue.

[6] It is not possible to equate Γ with $\perp$ since this would include the reflexive ordering of elements in the denotation of F ("$[\![F]\!] < [\![F]\!]$"). The issue of reflexivity is discussed in the section formalizing LP (4.1.2).

2.2 Type B: Ordering based on the function of a sign in a tree

While the information specified on the LP elements of type A statements always is information inherent to the sign as such, type B statements express order on a sign realizing a certain function (e.g. being the head or complement) in a local tree. Theoretically the subclassification into relative and absolute specification of order can be carried over from type A statements. However, since local trees usually combine signs of two functions only (i.e. a head and its complements, a head and adjuncts, a marker and a head).[7] the distinction is not as relevant as with type A statements. An example for an absolute type B statement is the rule which has to be assumed to order fillers in front of heads in constructions licensed by the head–filler–schema(Pollard and Sag in press).
Absolute type B LP statements fit the following pattern:

(7) $\Gamma \prec F$ or $F \prec \Gamma$, where
- F describes a set of feature structures that contain information about their *usage in a local tree*
- Γ describes any feature structure not describe by F

Type B LP statements can be expressed as type A if we encode the usage of the sign in the sign itself. Starting out in the lexicon with an unspecified function, the function would get specified upon usage in a constituent structure. To illustrate this in an HPSG framework, assume a new attribute FUNCTION for signs. Head complement structures could now be specified to encode the usage in each constituent by adding the constraint[8][9]

(9) $$\left[\begin{array}{ll} \text{HEAD–DTR} & {}_{sign}\left[\text{FUNCTION} \quad \textit{head–fn}\right] \\ \text{COMP–DTRS} & < {}_{sign}\left[\text{FUNCTION} \quad \textit{comp–fn}\right], \ldots > \end{array}\right]_{\textit{head–comps–struc}}$$

A further way to see type B LP is to interpret them as construction operations. This amounts to reintroducing a phrase structure backbone with disjunctions allowing free order for the constituents not ordered by LP statements dealing with their usage. In other words: type B statements can be hardwired into the immediate dominance schema. If several LP statements apply to one IDS, the partial ordering constraint conjoined to the IDS is a disjunction.
Summing up we note that a general way of treating LP statements is only necessary for type A statements.[10]

---

[7]The flat structure proposed in Pollard in press for German finite sentences is an example for a rule combining signs with more that two functions, namely the head, its subject and all its complements. No LP statements referring to these functions are used though.

[8]If we assume subtypes of sign, one for each possible function, one can encode the function specification directly in the appropriateness conditions of each constituent structure

[9]Note that the notation in (9) is not completely formal because of the "…" notation. The COMP–DTRS attribute really should have value a polymorphic list

(8) $\textit{list}({}_{sign}\left[\text{FUNCTION} \quad \textit{comp–fn}\right])$

as its value. Introducing polymorphism to the formalizm can be avoided by introducing subtypes of list for each usage.

[10]Note that processingwise this general way of treating LP statements is exactly the

3 Approaches to processing ID/LP grammars

3.1 Earley based Approaches

Shieber 1984: In this paper Shieber was the first to propose a treatment for ID/LP grammars that allowed direct parsing with an Earley based chart parser (Earley 1970) thereby overcoming the deficiency to precompile the ID/LP format to ordinary context free rules which for large grammars leads to huge numbers of grammar rules which in turn slows down parsing with standard methods like Earley's algorithm because the constant factor of the grammar size exceeds the cubic time depending on the length of the input string.
Shieber modifies Earley's algorithm in such a way that LP acceptable permutations of right hand sides of rules are accepted. Right hand sides of rules are treated as multisets and in both the predictor and the completer step elements of this set are extracted and LP acceptability is checked on them in relation to the order formed by the constituents already found. If the structure is valid, the hypothesis is accepted, if not, rejected.
Two problems remain. The first is of no concern here since it is dealing with the complexity of the algorithm (cf. Barton 1985). The second stems from the architecture of the grammar used: The given algorithm does not deal with feature terms and unification grammars but just with 'ordinary' ID/LP grammars. It is not entirely trivial to augment the algorithm so that it does, as can be seen in the following section.

Seiffert 1987: Seiffert modifies the Shieber/Earley algorithm so that it can handle unification based ID/LP grammars in a straightforward way and incorporates Shieber's idea of using a restrictor (Shieber 1985).
Two problems come up. One is connected with the use of the restrictor and the other one is the possible non local flow of information with unification ID/LP grammars.
If one picks a badly chosen restrictor it may well be that exactly that part of the information which is necessary to determine LP acceptability is discarded. This can easily be overcome by changing the restrictor to a more appropriate one or checking LP acceptability on the whole feature term and not on the restricted version of it.
The other problem is far more serious and leads to a complication of the algorithm. Consider a grammar like the following (taken from Seiffert 1987):

---

source of computational problems, since type A statments constrain constituents which might or might not appear in a specific local tree and one therefore has to try to match every constituent, whereas type B statements order elements which have to appear in a tree licenced by a particular ID schema.

(10) a. Start dag: $\begin{bmatrix}\textsc{cat} & a\end{bmatrix}$

b. ID–Rules:

$$\begin{bmatrix}\textsc{cat} & a\end{bmatrix} \rightarrow \begin{bmatrix}\textsc{cat} & b \\ \textsc{f} & \boxed{1}\end{bmatrix}, \begin{bmatrix}\textsc{cat} & c \\ \textsc{f} & \boxed{1}\end{bmatrix}$$

$$\begin{bmatrix}\textsc{cat} & b \\ \textsc{f} & \begin{bmatrix}\textsc{f}1 & \boxed{2} \\ \textsc{f}2 & \boxed{3}\end{bmatrix}\end{bmatrix} \rightarrow \begin{bmatrix}\textsc{cat} & d \\ \textsc{f}1 & \boxed{2}\end{bmatrix}, \begin{bmatrix}\textsc{cat} & e \\ \textsc{f}2 & \boxed{3}\end{bmatrix}$$

$$\begin{bmatrix}\textsc{cat} & c \\ \textsc{f} & \begin{bmatrix}\textsc{f}1 & \boxed{4} \\ \textsc{f}2 & \boxed{5}\end{bmatrix}\end{bmatrix} \rightarrow \begin{bmatrix}\textsc{cat} & f \\ \textsc{f}1 & \boxed{4}\end{bmatrix}, \begin{bmatrix}\textsc{cat} & g \\ \textsc{f}2 & \boxed{5}\end{bmatrix}$$

$$\begin{bmatrix}\textsc{cat} & d\end{bmatrix} \rightarrow \mathrm{d}$$

$$\begin{bmatrix}\textsc{cat} & e\end{bmatrix} \rightarrow \mathrm{e}$$

$$\begin{bmatrix}\textsc{cat} & f \\ \textsc{f}1 & one\end{bmatrix} \rightarrow \mathrm{f}$$

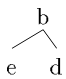
$$\begin{bmatrix}\textsc{cat} & g \\ \textsc{f}2 & two\end{bmatrix} \rightarrow \mathrm{g}$$

c. LP–Rules:

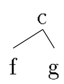
$$\begin{bmatrix}\textsc{f}1 & one\end{bmatrix} \prec \begin{bmatrix}\textsc{f}2 & two\end{bmatrix}$$

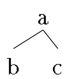
$$\begin{bmatrix}\textsc{cat} & b\end{bmatrix} \prec \begin{bmatrix}\textsc{cat} & c\end{bmatrix}$$

On input "**e d f g**", which is not well formed, the parser, proceding from left to right and bottom-up, can not determine whether the local tree (11)

(11) 

is well formed since the information that is relevant for the LP rule, namely that *one* has to precede *two*, is not yet available. Only after the local trees (12) and (13) have been constructed, the information becomes accessible through structure sharing.

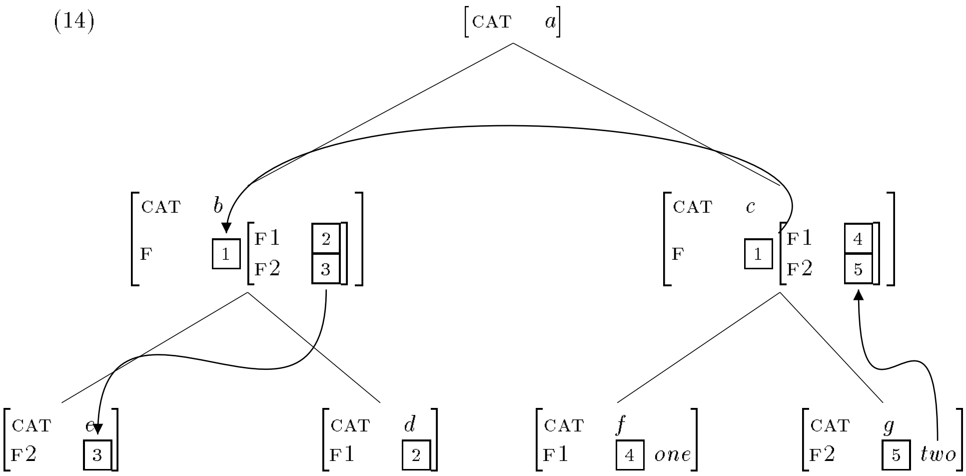
(12) 

(13) 

The non local flow can be seen in the parse tree in (14)[11]. The local tree (11) has to be ruled out since the LP rule "$\begin{bmatrix}\textsc{f}1 & one\end{bmatrix} \prec \begin{bmatrix}\textsc{f}2 & two\end{bmatrix}$" is violated.

[11]For readability reasons the arrows are drawn just for one value, namely *two*.

(14)



The information *one* and *two* comes from the lexical entries for *f* and *g*, is passed to their mother *c* via the structure sharing indicated by the tags $\boxed{4}$ and $\boxed{5}$. The information on *c* is (partly) structure shared with the information on *b* indicated by the tag $\boxed{1}$. This forces equality of the tags $\boxed{2}$ and $\boxed{4}$, and $\boxed{3}$ and $\boxed{5}$ respectively. Then the information is passed to the daughters of *b*, namely *e* and *d*, via the tags $\boxed{2}$ and $\boxed{3}$. Now the information that the value for *F2* at *e* is *two* and that the value for *F1* at *d* is *one* is available and the local tree (11) has to be ruled out since the LP rule "$\begin{bmatrix}\text{F1} & \textit{one}\end{bmatrix} \prec \begin{bmatrix}\text{F2} & \textit{two}\end{bmatrix}$" is violated.

This makes it necessary for Seiffert to take the completed parse tree and check it for any violations of LP rules after it has been completed. So he is checking LP acceptability twice, namely while constructing each local tree and after the parse has been completed. It may turn out that grammars describing fragments of natural language do not contain this non local flow of information. In this case the algorithm can be simplified and works as desired.

No attempt is made to deal with the transitivity property of the LP relation. In a case where a local tree contains three daughters, say *A*,*B* and *C*, and *A* has to be ordered in front of *B* which in turn has to be ordered in front of *C*, the order of the daughters is determined. But what happens in case we are faced with a local tree that contains just *A* and *C*? Do we want transitivity to force an order? In Seiffert's approach no order is forced if the transitive closure of LP rules is not precomputed. We turn to the problems involved with this precomputation in section 4.1.2.

The above mentioned problem of non locality of the algorithm can be avoided in another way. In Morawietz ms a relation of possible application is defined that allows to determine whether an LP rule might at some later stage apply to the structure and does a kind of freeze operation on it untill the relevant information is available. This fairly complicated mechanism allows to discover LP violations as early as possible thereby ruling out the undesired hypotheses.

Nevertheless both approaches suffer from the problem that the domain for LP rules are local trees. It may be necessary from a linguistic point of view to change the domain (Uszkoreit 1986, Reape 1990, Reape ms). Furthermore both require some kind of phrase structure backbone instead of HPSG's ID schemata. So a lot of work that the system should do, has to be done by the linguist.

Meknavin et al. 1992: The method proposed in this paper is a variation of Shieber's original idea too. ID rules are compiled into generalized discrimination networks and LP rules into Hasse diagrams (cf. Meknavin et al. 1992 for details on these terms) using bit vectors for greater efficiency. This representation is used to construct a chart parser which outperforms Shieber's parser by far, although the drawbacks are the same as noted for Shieber in the section on Seiffert's approach. It does not deal with unification and feature structure based grammars, but just with atomic terminals and nonterminals. Therefore non local flow of information is not discussed either.

Erbach 1991: Erbach's idea allows for the specification of an arbitrary domain by the linguist. Internally the constituents are collected in a list and LP acceptability is tested whenever a new item has to be added to the list. The problem of non locality as noted before is not solved, even though by having a larger domain it is less likely to occur.

3.2 Constraint based Approaches

Oliva 1992: This is the counterpart of Erbach's approach formulated in constraint based terms. By beefing up the type hierarchy larger domains than local trees are introduced. The new list types serve to collect constituents just as done by Erbach 1991. Nothing is said about how the LP rules are interpreted, nor when they do apply. The paper is more concerned with the creation of this new domain than with a treatment of LP rules.

Engelkamp et al. 1992: In this paper the authors try to exploit HPSG's properties to cope with LP rules in a way that a violation of an LP rule leads to unification failure. They assume binary branching ID rules and the domain for the application of LP rules is a so called head domain consisting of the head and its adjuncts and complements. LP rules are precompiled in the lexical entries in such a way that every entry contains information on which constituents must not appear to its left and right. This information is accumulated so that the violation can be discovered locally. Two new principles and some features are introduced to ensure this. They choose to have the value for the SUBCAT feature to be a set rather than a list following (cf. Pollard in press) since they want to determine linear precedence entirely by LP statements. This is not necessary since word order in HPSG is not done via the subcat list, though it happens to coincide very nicely for English so that one might be tempted to do so. Parameterized types are used in the lexicon to ensure the right description of the LP relevant features. Clearly this precompilation only works if one assumes that the features of the LP constraints appear in the lexicon.
The drawbacks of this approach lie in the corruption of features to do word order and the duplication of the information on each entry and all the intermediate stages of the parse. The restriction on having binary branching structures only can be avoided by having additional principles to cope with the resulting varia-

tions on percolating the information relevant to linear precedence. The decision of having a subcat set would lead to problems with HPSG's binding theory, but this is not an issue here.

Though the problem is mentioned, no treatment of conflicting LP statements is proposed. LP constraints are viewed as absolute constraints that lead to failure.

Blache 1992: The method used in this paper treats LP as active constraints and uses a variatant of bottom up filtering (Blache and Morin 1990). LP rules are used to determine possible boundaries for phrases, possible immediate precedence relations and the notion of an initial element. LP acceptability is checked on each phrase. So the purpose of LP rules is twofold. They help to determine phrases thereby influencing the control strategy and they check the validity of the phrases themselves. Though this is formulated only for basic ID/LP grammars, Blache claims that it is no problem to extend this to feature based systems.

It is not clear how LP acceptability is checked on the phrases. Some open questions remain. When does an LP constraint apply, are the domains really phrases, what happens with the transitivity property?

3.3 Summing up

No fully satisfactory treatment of the ID/LP format in relation to computational linguistics has been proposed. Although some approaches do work, more effort is necessary to overcome their deficencies. The major drawback of all the approaches seems to be that none tries to provide a semantics for LP statements; LP satements just function when used in an algorithm. Some of the supposed properties of LP are not considered. Several open questions remain: What does an LP statement mean? What does the property of LP being transitive give us? When does an LP rule apply?

4 Some problematic properties of LP

4.1 Transitivity of LP statements

4.1.1 Varieties of transitivity and their usefullness for linguistics

In Gazdar et al. 1985 LP is proposed to be an antisymmetric, transitive relation (p.46).[12] It is rather clear that one does not want LP to be reflexive, since this would rule out two elements of the denotation of a feature term to occur together in some domain.[13] Therefore LP cannot be an order in the classical sense.[14]

[12] For convenience we give the standard definitions of mathematical terms used (S a set; R a relation; $x, y \in \mathcal{S}$):
asymmetry: $x \leq y \Rightarrow \neg(y \leq x)$. This is anti–symmetry and irreflexivity.
anti–symmetry: $(x \leq y \;\&\; y \leq x \Rightarrow x = y)$
reflexivity: $\forall x \in \mathcal{S} : (x < x) \in \mathcal{R}$
irreflexivity: $\neg \exists x \in \mathcal{S} : (x < x) \in \mathcal{R}$
non–reflexivity: $\neg$ reflexive $\wedge$ $\neg$ irreflexive

[13] This might be useful in some cases though. For an example cf. Gazdar et al. 1985, p.180, fn.11.

[14] An *order* is a binary relation which is transitive and in addition either (i) reflexive and antisymmetric or else (ii) irreflexive and asymmetric. The former are *weak* orders; the latter are *strict* (or *strong*)(Partee et al. 1990)

The idea of transitivity has not received much attention in the years since. What kind of transitivity is linguistically motivated and how does it fit into the formalism?

In section 3.1 on Seiffert's approach a problem with the transitivity is briefly mentioned, namely the question of what should happen in case we have to determine a linear order between two elements where we could have a third one in between that would determine the order of the other two by forcing them to its right and left respectively. To illustrate this consider the topological model of German sentences proposed in Hoehle Höhle 1986. In an approach like Reape's one might be tempted to use the topological units as properties of the signs to be ordered and specify e.g. for verb second sentences the (informally noted) LP statements[15] given in (15).

(15) $topic \prec finite--verb \prec Mittelfeld \prec nonfinite-verbs \prec extraposed$

At least in this case it is clear that the order of finite and nonfinite verbs is determined also in the absence of a Mittelfeld as in (16).

(16) Karl hat geschnarcht.

In this case we have to precompute the transitive closure of the LP rules when doing parsing so that we can determine all possible orders. In which form is this linguistically desired? Under which circumstances do we want transitivity to apply?

The LP statements to be considered are of the form $A \prec B$ and $C \prec D$. For reasons of simplicity only the relation between B and C will be considered and not the one that possibly exists between A and D.

When viewing transitivity and its linguistic relevance the following three cases should be considered.

4.1.1.1 Transitivity applies if $[\![B]\!] \cap [\![C]\!] \neq \emptyset$

Transitivity applies if $[\![B]\!] \cap [\![C]\!] \neq \emptyset$

(17) a. $\begin{bmatrix}\text{CASE} & acc\end{bmatrix} \prec \begin{bmatrix}\text{CASE} & dat\end{bmatrix}$

b. $\begin{bmatrix}\text{DEF} & definite\end{bmatrix} \prec \begin{bmatrix}\text{DEF} & indefinite\end{bmatrix}$

Since the intersection of $[\![\begin{bmatrix}\text{CASE} & dat\end{bmatrix}]\!]$ and $[\![\begin{bmatrix}\text{DEF} & definite\end{bmatrix}]\!]$ presumably is not empty computing the transitive closure for this example would lead to the additional LP statement "$\begin{bmatrix}\text{CASE} & acc\end{bmatrix} \prec \begin{bmatrix}\text{DEF} & indefinite\end{bmatrix}$". This is clearly an overgeneration considering linguistic needs.

4.1.1.2 Transitivity applies if $[\![B]\!] \subseteq [\![C]\!]$

Transitivity applies if $[\![B]\!] \subseteq [\![C]\!]$

16

The problem encountered in example (17) no longer occurs under the new assumption for transitivity since in general $[\![\begin{bmatrix}\text{CASE} & dat\end{bmatrix}]\!] \not\subseteq [\![\begin{bmatrix}\text{DEF} & definite\end{bmatrix}]\!]$. However consider another case:

(18) a. $\begin{bmatrix}\text{CASE} & acc\end{bmatrix} \prec \begin{bmatrix}\text{CASE} & dat \\ \text{DEF} & definite\end{bmatrix}$

[15]We use the notation "$A \prec$ "$B \prec C$"" as abbreviation for the statements "$A \prec B$" and "$B \prec C$"

[16]Naturally an argumentation parallel to the one we are presenting her also holds for the inverse case, where $[\![B]\!] \supseteq [\![C]\!]$.

b. $[\text{DEF} \quad \textit{definite}] \prec [\text{DEF} \quad \textit{indefinite}]$

Here computing the closure results in the new LP statement:

18c. "$[\text{CASE} \quad \textit{acc}] \prec [\text{DEF} \quad \textit{indefinite}]$".

It is clear that linguistically one does not want to deduce the fact that accusative NPs precede indefinite NPs from knowing that accusative NPs precede definite dative NPs and definites precede indefinites.

4.1.1.3 Transitivity applies if $[\![B]\!] = [\![C]\!]$

Transitivity applies if $[\![B]\!] = [\![C]\!]$

This most restrictive view of transitivity avoids the problems noted earlier.

The problem with this approach lies in the possible use of negation in LP elements[17]. Consider the following two LP statements.

(19) a. $[\text{CASE} \quad \textit{nom}] \prec [\text{CASE} \quad (\neg\, \textit{nom} \wedge \neg\, \textit{gen})]$
b. $[\text{CASE} \quad \textit{acc}] \prec [\text{CASE} \quad \textit{gen}]$

We know from the signature that the type *case* is partitioned b y *nom*, *acc*, *dat*, and *gen*. Assuming that the maximally specified types partition the domain (19a) is equivalent to "$[\text{CASE} \quad \textit{nom}] \prec [\text{CASE} \quad \textit{acc} \vee \textit{dat}]$" one would then want to be able to compute "$[\text{CASE} \quad \textit{nom}] \prec [\text{CASE} \quad \textit{gen}]$". The identity criterion however does not permit the computation of the closure in this case without additional assumptions. If we allow equivalence transformations on LP statements and with the assumption of partitioning (as in our example) we can transform (19a) into the following:

(19) a'. $[\text{CASE} \quad \textit{nom}] \prec [\text{CASE} \quad \textit{acc}]$
a". $[\text{CASE} \quad \textit{nom}] \prec [\text{CASE} \quad \textit{dat}]$

Statement (19a') together with (19b) now allows us to compute the desired LP statement "$[\text{CASE} \quad \textit{nom}] \prec [\text{DEF} \quad \textit{indef}]$".

Summing up this brief overview we note that only computing transitivity under identity seems to capture the linguistic intuitions about linear precedence.

4.1.2 A denotational semantics for LP statements

After having looked at the different possibilities for precomputing transitive closure from a linguistic point of view in this section we will analyze the LP formalism as such. We will show that the formulation of a denotational semantics for LP statements based on an order relation on feature structures automatically entails transitivity of descriptions of the type of item 4.1.1.1 above. In the last section however this type of transitivity was shown not to be linguistically adequate.

[17] The example (19) only involves negation of types. However negating a feature term works in a parallel fashion, with one extra issue to be mentioned: If attributes are partial functions from types to types, negating something like "$_t[\text{X} \quad r]$" is equivalent to "$\neg t \vee \text{X}\uparrow \vee [\text{X} \ \neg\ \text{R} \quad]$", which is trivial except for the $\text{X}\uparrow$ meaning X is undefined, which needs a special treatment. By having attributes as total functions and introducing a new type *undefined*, we avoid that complication and can simply state "$\neg t \vee [\text{X} \quad \neg\ r]$" since $\neg r$ will include *undefined*. However having to have every attribute defined on every type clearly also is a disadvantage.

(20) a. If $LP = \{A_1 \prec B_1, \ldots, A_n \prec B_n\}$ is a finite set of LP statements, $[\![A_i]\!] \cap [\![B_i]\!] = \emptyset$[18], then $\langle U, < \rangle$ is the smallest transitive asymmetric[19] relation on a universe U such that $\forall u_1, u_2 \in U$:
If $u_1 \in [\![A_i]\!]$ and $u_2 \in [\![B_i]\!]$ for some $A_i \prec B_i \in LP$, then $u_1 < u_2$.[20]
b. If x and y occur in some domain and $x < y$ then x precedes y in time (in speech) and space (in text).

Anti–symmetry seems desirable, since symmetry, demanding that if "$F \prec G \in LP$" then "$G \prec F$" has to be in LP too, produces conflicting statements.

Another point to discuss is the motivation of the condition that $[\![A_i]\!] \cap [\![B_i]\!] = \emptyset$. It seems reasonable to demand that the intersection of two elements of an LP statement be empty if you consider the anti–symmetry condition[21]. Consider $[\![A_i]\!] = \{a, b, c\}$ and $[\![B_i]\!] = \{b, c, d\}$. Here we are faced with the problem of ordering b in front of c and vice versa. This clearly violates anti–symmetry. Therefore as long as there is no real linguistic motivation to assume the intersection to be nonempty, we want to maintain the property.

Note that in (20) the term domain is not specified, since we believe the question which kind of domain should be chosen to be an empirical one. One could incorporate any appropriate domain, such as local trees (as in Gazdar et al. 1985), head domains (Engelkamp et al. 1992, Kiss 1992), or seperate word order domains (Reape's work).

A further issue should at least be mentioned in passing. What properties of linguistic objects are ordered, that is precede each other as specified in (20b)? The phonology only, whole signs, or something like constituent structure units (for type B statements). This is relevant to processing, since one needs to have the information which can be referred to by an LP statment available in the representation of linearized constituents, if one wants to incrementally add the rest of the elements of the linearization domain.

What effect does the definition (20) have on our discussion about transitivity? The ordering relation on feature structures " $<$ " is defined to be transitive. The following example shows that the corresponding criterion for descriptions is of the unification type discussed in (4.1.1.1).

(21)

$$\begin{array}{ccc} F & \prec & G \\ H & \prec & I \end{array} \quad \text{with} \quad \begin{array}{ccc} [\![F]\!] & = & \{a, b\}, \\ [\![G]\!] & = & \{c, d\}, \\ [\![H]\!] & = & \{d, e\}, \\ [\![I]\!] & = & \{f, g\} \end{array}$$

[18]This excludes $A \prec A$ as well.

[19]The relation is irreflexive because of the demand that the intersection of the denotations has to be empty.

[20]We use " $<$ " as notation for a relation on sets of feature structures. In these cases we intend the following interpretation:
$\mathcal{S}_1 < \mathcal{S}_2 := \forall x \in \mathcal{S}_1$ and $\forall y \in \mathcal{S}_2$ $x < y$.

[21]It could be a set with just one element though. What happens in that case? How would you determine in an example that there is indeed only one feature structure matching the description? "$A \prec A$" would be allowed if $[\![A]\!]$ is a singleton set. The relation would become non–reflexive and anti–symmetric. The rest of the argument remains unchanged.

We can spell the relation " $\prec$ " out so that we derive the following statements on feature structures (relation " $<$ ") from which by transitivity of " $<$ " some new statements can be concluded which in turn can be simplified (in this case) to one LP statement.

(22)

$$\begin{array}{ccc} a & < & c \\ a & < & d \\ b & < & c \\ b & < & d \\ d & < & f \\ d & < & g \\ e & < & f \\ e & < & g \end{array} \Rightarrow \begin{array}{ccc} a & < & f \\ a & < & g \\ b & < & f \\ b & < & g \end{array} \Rightarrow \{a, b\} \quad < \quad \{f, g\} \quad \Rightarrow F \prec I$$

Therefore we can, just by transitivity of " $<$ ", deduce that "$\{a, b\} < \{f, g\}$", which leads to the conclusion "$F \prec I$" which corresponds to computing the transitive closure as in (4.1.1.1) via $[\![B]\!] \cap [\![C]\!] \neq \emptyset$, with the problems discussed there.

Wanting to keep the linguistic motivation, one cannot maintain the idea of transitivity of LP with the straightforward formalization of LP semantics given.

Consequently the idea of Gazdar et al. 1985 to have LP as an anti–symmetric transitive relation cannot be sustained. Furthermore the assumption that $[\![A_i]\!] \cap [\![B_i]\!] = \emptyset$ made in (20a) makes it impossible to use LP rules of the form "$A \prec A$" to express that only one element meeting the description A can appear in a local tree.

> 'An LP statement of the form "$A \prec A$"' has the effect of preventing two nodes labeled A from occurring as sisters, in view of its interpretation as 'a node labeled A cannot appear to the left of a node labeled A in a local tree'.' (Gazdar et al. 1985, p.55)

Since this usage would corrupt LP statements to do things which are clearly not linear precedence but something entirely different, we will not count this as real loss.

4.2 Negation and the application of LP rules

This section serves to show that applying LP statments requires negation. According to the semantics given above, an LP statement "$F \prec G$" translates to:

$$\forall A, B : match(A, F) \wedge match(B, G) \rightarrow [\![A]\!] < [\![B]\!]$$

$$\Leftrightarrow \forall A, B : \neg match(A, F) \vee \neg match(B, G) \vee [\![A]\!] < [\![B]\!]$$
$$\Leftrightarrow \forall A, B : [\![B]\!] < [\![A]\!] \rightarrow \neg match(A, F) \vee \neg match(B, G)$$

The operation *match* is defined as follows:

Definition: Match[22]

Let $\mathcal{F}$ be a set of feature terms.

$$match : \mathcal{F} \times \mathcal{F} \rightarrow \{T, F\}$$
$$match(A, B) := [\![A]\!] \cap [\![B]\!] \neq \emptyset$$

Checking whether an LP statment "$F \prec G$" allows the feature structures described by B to precede those described by A therefore boils down to the following predicate "$check_{F \prec G}$(A,B)".

$$check_{F \prec G}(B, A) := \neg match(A, F) \vee \neg match(B, G)$$

Note that in the above definition the *relation* match is negated, not just a *feature term*. Since it is not clear whether it is possible to handle full negation of relations with feature logics, it is interesting to note that the negation in this case can be pushed down to the feature term under certain conditions[23]:

$$[\![A]\!] \neq \emptyset \ \wedge \ ([\![A]\!] \subseteq U \backslash [\![F]\!] \ \vee \ [\![A]\!] \subseteq [\![F]\!]) \ \models \ \neg match(A, F) \leftrightarrow match(A, \neg F)$$

Applied to linguistics the first condition $[\![A]\!] \neq \emptyset$ means that the description of the constituent being matched to an LP–element has to be consistent, i.e. describe some feature structure. This restriction is not problematic. The other conjunct is more interesting: The description is either subsumed by[24] the LP–element or by the negation of the LP–element. Note that this is a real constraint on descriptions that needs to be satisfied every time a LP statment is checked.

4.3 Further problems

One problem of the ID/LP format remains to be mentioned. It seems to be linguistically motivated to allow conflicting ordering statements and resolve the conflict via introducing a hierarchy among ordering rules. This has been adressed in depth by Uszkoreit 1984. To give a brief example: If one considers a situation where a pronominal accusative NP has to be ordered relative to a nonpronominal dative NP in the German Mittelfeld and the relavant LP statemants demand that pronominals precede nonpronominals and dative NPs precede accusative NPs. Since there are structures in German allowing these to occur, but the LP statements would forbid them, one has to find a way to weigh LP rules so that degrees of acceptability are possible instead of forcing a failure in these cases of clashing LP statements. Otherwise one has to write very explicit LP statements, to cope with each possible constellation separatly thereby loosing generalizations.

[22]Match could be definied in three ways: as unification, test unification, or subsumption check. Since we decide to use test unification this would forces us to check the completed parse tree again after completion, since usage of unification elsewhere in the grammar might lead to nonlocal flow of information (cf. the discussion of Seiffert 1987 in section 3.1). The same holds true for checking subsumption. Defining match via unification is the only possibility to automatically obtain a failure in the inadequate cases during the whole parse.

[23]The small proof of this is included in the appendix for convenience sake.

[24]A is subsumed by $F \Leftrightarrow A$ is more specific that $F \Leftrightarrow A$ describes less feature structures than F

5 Motivating change

5.1 Summing up the motivation for leaving the LP format

In section 2 we discussed two types of LP statments. While type A LP rules order signs according to sign inherent properties, type B LP rules make reference to the function of a sign in a tree. Type A rules correspond to the traditional view of LP rules as word order generalizations over the whole grammar. Type B rules on the other hand, are more construction specific, and - as noted in section 2.2 - it therefore seems natural to understand them as constructing a phrase structure like backbone on which the type A rules can then be applied to order the elements not fixed by type B rules.

While viewing some formal properties of LP rules in section 4.2, it turned out that the semantics of LP rules makes use of negation to rule out unacceptable structures. However section 3 showed that approaches to process ID/LP grammars can avoid the enormous complexity involved in generating all possible immediate dominance trees before ruling out the ones violating the LP relations. Nonetheless all of these approaches were shown to have their problems, for example in dealing with nonlocal information flow of LP relevant information. One is therefore tempted to investigate other possibilities for expressing word order generalizations, allowing straight forward algorithms to licence acceptable structures, instead of ruling out unacceptable ones.

Before turning to the exploration of an alternative formulation for word order generalizations which takes into account the two points just given, a further, linguistic motivation for leaving LP format should be mentioned.

5.2 Some motivation from linguistics

In linguistics one is confronted with the phenomenon of "marked word orders": While a certain construction type with an "unmarked" word order is acceptable within a wide range of grammatical variations, the same construction type is far more constrained if the word order is changed, this more constrained word order being called the "marked" order.
Let us illustrate the difference between marked and unmarked word orders using an observation regarding the linearization of nominal elments in the German "Mittelfeld" discussed in Lenerz 1977.
The following four examples show sentences[25] in which the direct object (DO) preceedes the direct object (IO). Since the topic-focus structure playes a role, questions are supplied to situate the sentences in a context.

(23) a. Wem hast du ein Buch (DO indef.) geschenkt?
Who did you give a book?
b. * Ich habe ein Buch dem Schüler geschenkt.
I have a book (DO indef.) the student (IO) give
I gave a book to the student.

(24) a. Wem hast du das Buch (DO def.) geschenkt?
Who did you give the book?

[25] The example sentences in this section are taken from Lenerz 1977, pp. 43-44 and 52-54.

b. Ich habe das Buch dem Schüler geschenkt.
I have the book (DO def.) the student (IO) give
I gave the book to the student.

The examples (23) show that an indefinite direct object cannot precede a definite indirect object. Example (24) demostrates that this word order is possible if the direct object is definite.

In (25) and (26) we note a further constraint: a direct object can only precede the indirect object if it is more thematic than the indirect object.

(25) a. Was hast du dem Kassierer gegeben?
What did you give to the cashier? (IO thematic)
b. ?* Ich habe das Geld dem Kassierer gegeben.
I have the money (DO) the cashier (IO) give
I gave the money to the cashier.

(26) a. Wem hast du das Geld gegeben?
Who did you give the money? (DO thematic)
b. Ich habe das Geld dem Kassierer gegeben.
I have the money (DO) the cashier (IO) give
I gave the money to the cashier.

In sentences in which the direct object follows the indirect object, the constraints observed with DO IO order do not appear. All of the sentences are grammatical.

(27) a. Hier sind mehrere mögliche Geschenke zur Auswahl; was würdest du einem Schüler (DO) schenken?
Here are several possible presents to chose from; what would you give to a student?
b. Ich würde einem/dem Schüler ein/das Buch schenken.
I would a/the student (IO indef./def.) a/the book (DO indef./def.) give
To a/the student I would give a/the book.

(28) a. Was hast du dem Kassierer gegeben?
What did you give to the cashier? (IO thematic)
b. Ich habe dem Kassierer das Geld gegeben.
I have the cashier (IO) the money (DO) give
I gave the money to the cashier.

(29) a. Wem hast du das Geld gegeben?
Who did you give the money? (DO thematic)
b. Ich habe dem Kassierer das Geld gegeben.
I have the cashier (IO) the money (DO) give
I gave the money to the cashier.

Summing up we can state that the IO DO order is the unmarked order of objects in the German Mittelfeld. The DO IO order is the marked order, which is possible as well, but subject to additional constraints.

It is hard to see, how the observation of marked word orders can be captured in the ID/LP formalism, since it is impossible to distinguish between the different LP acceptable orders. To be able to capture the differences between marked and unmarked word orders, one needs a formalism which can attach different constraints to constructions, which differ only with respect to linearization.

6 Exploring successive permutation format (SUP)

6.1 An alternative idea: allowing beats ruling out

In the section 4 we discussed problems with LP stemming from two main areas: The assumption of transitivity demanding computation of the transitive closure, and the need for negation to test LP statements. We have shown that a denotational semantics for LP based on the common assumptions is not available. Therefore we now turn to another idea for lineariztion, which replaces the restricting approach of LP statements by a permitting one.

Summing up once more, the idea of ID/LP–format is to allow all possible local trees from an ID rule in a first step. This means all linear orders of elements on the right hand side of a rule has to be considered. In a second step the trees not conforming to the LP statements were ruled out. Under this approach free word order is the easiest to deal with, fixed word order requires many LP statements. SUP format on the opposite starts out from a fixed order of elements on the rhs stemming from the application of a phrase structure rule. Then further orders are allowed by successive permutation according to a set of PERM rules. This makes fixed order languages the easiest, free word order languages the hardest to describe.

We will note PERM rules as $F \lhd G$, sometimes speak of F as the "buck" and G as the "vaulter".

6.2 Possible ways of exploration

After talking about the general idea behind SUP format we now turn to possible varieties of such a formalism.

6.2.1 The effect of PERM rules

What is the effect of the application of a PERM rule? One option is to have the PERM rule forcing a new order. For example the ps rule "A →B C" together with the PERM statment $B \lhd C$ would allow only trees in which Cs are ordered in front of Bs. The other possibility is to use PERM statments to increase possibilities for building structures. Here a PERM rule can be applied, but does not have to be. By forcing order changes we do not licence any "free order" subdomains loosening up the phrase structure backbone, but rather replace one rule (in certain applications) by another one. Therefore it seems that allowing permutations instead of forcing them is closer to our aim of loosening up the rigid linearization of the PS backbone.

6.2.2 The Domain of PERM rules

The next choice concerns the domain for a single permutation. On the most permissive view one could allow the permutation of any pair of daughters in a local tree. A stricter notion of domain permits PERM rules to apply to adjacent daughters only. To evaluate these approaches, take the following example for a PERM rule applying to non–adjacent daughters:

(30) a. PS rule 1: $A \rightarrow BC_1D$
b. PS prule 2: $A \rightarrow BC_2D$
c. PERM rule 1: $B \lhd D$

In (30) it is not possible to express that D can permute in such a way that it precedes B if applied to constructions licensed by (30a), but not by (30b).[26] Nevertheless we can express this, if we assume the restrictive view using the following PERM rules:

(31) a. PERM rule 1: $C_1 \lhd D$
  b. PERM rule 2: $B \lhd D$

Permutation of non–adjacent daughters can be achieved via successive application of PERM rules.[27]

6.2.3 The Range of PERM rules

The next decision has to be made on the range of the permutation. Does a PERM rule allow the exchange of two elements, or should it permit arbitrary insertion of the vaulter into the string of elements preceding the buck? The effects of both approches are illustrated in the example (32) where (32b) shows the result of applying the PERM rule under the exchange assumption and (32c) the ones resulting from arbitrary insertion.

(32) a. A →B C D E
  $D \lhd E$
  b. "exchange" adds 1 rule: A →B C **E** D.
  c. "arbitrary" adds 3 rules:
    A →B C **E** D.
    A →B **E** C D.
    A →**E** B C D.

The drawbacks of the arbitrary insertion approach are parallel to those of permuting non-adjacent elements. Exchanging elements allows us to differentiate more.

6.2.4 Formal properties of the PERM relation

We want PERM to be an irreflexive, anti–symmetric, and not transitive relation on feature terms.

Irreflexive: Reflexive PERM rules are excluded to ensure termination of the successive application of PERM rules.

Anti–symmetry: As with irreflexivity we must assume PERM to be anti–symmetric to get a least fix point (in the non–technical sense) for successive permutation. Furthermore we need to guarantee that no cycles can occur even in indirect cases. Therefore the following must hold for all PERM statments:
$F \lhd G,\ H \lhd I \rightarrow ([\![F]\!] \cap [\![I]\!] \neq \emptyset) \wedge ([\![G]\!] \cap [\![H]\!] \neq \emptyset)$?

Transitivity: PERM is not a transitive relation as can be seen in (33).

(33) $F \lhd G$
  $G \lhd H$
  linear order of daughters from ps rule: $F\,G\,H$
  PERM results: $G\,F\,H$, $F\,H\,G$

[26]Naturally this could be expressed by adding one more PS rule and eliminating PERM rules alltogether. However this allows no generalizations over linear precedence in PS rules, which was the purpose of PERM rules to begin with.

[27]Note that successive permutation must be licensed each time by (possibly different) PERM rules.

6.2.5 Summary

The discussion in the last sections leads us to formulate SUP grammars with PERM rules in the follwing way:

- PERM rule application increases order possibilities.
- PERM rules apply to adjacent elements.
- PERM rules exchange the elements they apply to.
- A set of PERM rules does not denote an order, since PERM rules are not transitive.

Note that this entails that nonlocal reordering can only be achieved via successive local permutation.

7 Formalizing SUP grammars

Let $\mathcal{F}$ be a set of feature terms.

7.1 Extending basic feature based ps grammars

Definition: SUP Grammar
A SUP grammar is a quintuple

$$G = \langle N, T, \Pi, \Psi, S \rangle$$

with

- *N a set of feature terms representing the non–terminals (metalanguage),*
- *T another set of feature terms representing the terminals (= lexicon, object language), $N \cap T = \emptyset, N \cup T = \Sigma$,*
- *Π a set of rules of the form $F_0 \rightarrow F_1, \ldots, F_n$ with F_i feature terms,*
- *Ψ a set of PERM rules of the form $F_2 \lhd F_1$, F_1 and F_2 feature terms.*
- *S the start feature term, $S \in N$.*

Definition: Compatibility of local trees
A local tree[28] θ is compatible with a ps rule $p \in \Pi$ iff the mother of θ unifies with the lhs–element of p and the daughters of θ in the linear order of realization unify with the rhs of p.

7.1.1 Allowing Permutations

Let G = $\langle N, T, \Pi, \Psi, S \rangle$ be a SUP grammar, F, G and H (with subscript variants) elements of Σ. Furthermore assume Θ a set of local trees, and $\theta \in \Theta$.

Definition: Application of PERM rules
A PERM rule is a function from local trees into local trees ($H \lhd G$: $\Theta \rightarrow \Theta$). Let G be a SUP grammar and θ_1 a local tree. A PERM rule $H \lhd G$ applies to θ_1 iff

[28] We use the term local tree to denote trees with depth one.

$$\theta_1 = \begin{array}{c} F_0 \\ F_1 \quad \dots \quad F_i \quad F_{i+1} \quad \dots \quad F_n \end{array} \qquad \textit{and}$$

$$match(H, F_i) \wedge match(G, F_{i+1})$$

yielding a second local tree

$$\theta_2 = \begin{array}{c} F_0 \\ F_1 \quad \dots \quad F_{i+1} \quad F_i \quad \dots \quad F_n \end{array}$$

Definition: SUP acceptability
A local tree θ_1 is acceptable by a SUP grammar iff

- $\exists\, p \in \Pi$ *of the form* $F_0 \rightarrow F_1 \dots F_n$ *and* θ_1 *is compatible with* p*, or*
- $\exists$ *a PERM rule* $\psi \in \Psi$ *such that* $\psi(\theta_2) = \theta_1$ *and* θ_2 *is SUP acceptable.*

A tree T is acceptable by a SUP grammar iff every of its local trees is SUP acceptable.

Definition: Language of a SUP grammar
The language of a SUP grammar $L(G)$ contains all words w which are formed by the leaves of a tree T which is SUP acceptable in G.

8 Summing up

The purpose of this paper was to discuss linguistic and formal properties of the LP formalism, pointing out problems in the formalization of a semantics for LP statements and in computational issues involving transitivity and negation.
We then turned to formalizing a theory based on feature based PS grammars, capturing generalizations about additional possible orders via so called PERM rules.
Two things remain to be done: On the computational side a generation and a parsing method for SUP grammars has to be develloped which takes the possible nonlocal flow of information into account (cf. Seiffert 1987), while on the empirical, linguistic side the usefulness of the new formalism needs to be studied.

Appendix

The proof for section 4.2

Under what conditions C can we push the negation from the relation down to the feature term? That is: $C \models \neg match(A,F) \leftrightarrow match(A,\neg F)$.

Transforming the first part:

$$
\begin{array}{lll}
& \neg match(A,F) & \\
\Leftrightarrow & \neg((\llbracket A \rrbracket \cap \llbracket F \rrbracket) \neq \emptyset) & \text{Def.: match} \\
\Leftrightarrow & (\llbracket A \rrbracket \cap \llbracket F \rrbracket) = \emptyset & \text{double-negation elimination} \\
\Leftrightarrow & (\llbracket A \rrbracket \cap U \setminus \llbracket F \rrbracket) = \llbracket A \rrbracket &
\end{array}
$$

Transforming the second part:

$$
\begin{array}{lll}
& match(A,\neg F) & \\
\Leftrightarrow & (\llbracket A \rrbracket \cap \llbracket \neg F \rrbracket) \neq \emptyset & \text{Def.: match} \\
\Leftrightarrow & (\llbracket A \rrbracket \cap U \setminus \llbracket F \rrbracket) \neq \emptyset & \text{Def.: Negation of feature terms}
\end{array}
$$

Determining the conditions for one implication:

$$
\begin{array}{ll}
& (\llbracket A \rrbracket \cap \llbracket F \rrbracket) = \emptyset \ \rightarrow \ (\llbracket A \rrbracket \cap U \setminus \llbracket F \rrbracket) \neq \emptyset \\
\Leftrightarrow & (\llbracket A \rrbracket \cap \llbracket F \rrbracket) \neq \emptyset \ \vee \ (\llbracket A \rrbracket \cap U \setminus \llbracket F \rrbracket) \neq \emptyset \\
\Leftrightarrow & \llbracket A \rrbracket \neq \emptyset \\
& \text{Thus the following holds:} \\
\text{I.} & \llbracket A \rrbracket \neq \emptyset \ \models \ (\llbracket A \rrbracket \cap \llbracket F \rrbracket) = \emptyset \ \rightarrow \ (\llbracket A \rrbracket \cap U \setminus \llbracket F \rrbracket) \neq \emptyset
\end{array}
$$

Determining the conditions for the second implication:

$$
\begin{array}{ll}
& (\llbracket A \rrbracket \cap \llbracket F \rrbracket) = \emptyset \ \leftarrow \ (\llbracket A \rrbracket \cap U \setminus \llbracket F \rrbracket) \neq \emptyset \\
\Leftrightarrow & (\llbracket A \rrbracket \cap \llbracket F \rrbracket) = \emptyset \ \vee \ (\llbracket A \rrbracket \cap U \setminus \llbracket F \rrbracket) = \emptyset \\
\Leftrightarrow & \llbracket A \rrbracket \subseteq U \setminus \llbracket F \rrbracket \ \vee \ \llbracket A \rrbracket \subseteq \llbracket F \rrbracket \\
& \text{Thus the following holds:} \\
\text{II.} & \llbracket A \rrbracket \subseteq U \setminus \llbracket F \rrbracket \ \vee \ \llbracket A \rrbracket \subseteq \llbracket F \rrbracket \ \models \ (\llbracket A \rrbracket \cap \llbracket F \rrbracket) = \emptyset \ \leftarrow \ (\llbracket A \rrbracket \cap U \setminus \llbracket F \rrbracket) \neq \emptyset
\end{array}
$$

From I. and II. we can conclude:

$$\llbracket A \rrbracket \neq \emptyset \ \wedge \ (\llbracket A \rrbracket \subseteq U \setminus \llbracket F \rrbracket \ \vee \ \llbracket A \rrbracket \subseteq \llbracket F \rrbracket) \ \models \ \neg match(A,F) \leftrightarrow match(A,\neg F)$$

References

Barton, G. 1985. The Computational Difficulty of ID/LP Parsing. In *Proccedings of the 23rd Annual Meeting of the ACL*, 76–81.

Blache, P. 1992. Using Active Constraints to Parse GPSGs. In *Proceedings of COLING-92*, 81–86.

Blache, P., and J.-Y. Morin. 1990. Bottom-Up Filtering: a Parsing Strategy for GPSG. In *Proceedings of COLING-90.*

Earley, J. 1970. An Efficient Context-Free Parsing Algorithm. *Comm. ACM 13.2* 94–102.

Engelkamp, J., G. Erbach, and H. Uszkoreit. 1992. Handling Linear Precedence Constraints by Unification. In *ACL Proceedings, 23th Annual Meeting*, 201—208.

Erbach, G. 1991. A flexible parser for a linguistic experimentation environment. In *Text Understanding in LILOG*, ed. O. Herzog and C.-R. Rollinger. Berlin: Springer.

Gazdar, G., E. Klein, G. Pullum, and I. Sag. 1985. *Generalized phrase structure grammar.* Cambridge, Massachusetts: Harvard University Press.

Höhle, T. N. 1986. Der Begriff 'Mittelfeld'. Anmerkungen über die Theorie der topologischen Felder. In *Kontroversen alte und neue. Akten des VII. Internationalen Germanistenkongresses Göttingen 1985*, ed. A. Schöne, 329–340. ? Bd. 3.

King, P. J. 1992. Unification Grammars and Descriptive Formalisms. unpublished lecture notes, Seminar für Sprachwissenschaft, Universität Tübingen.

Kiss, T. 1992. *Infinite Komplementation. Neue Studien zum deutschen Verbum infinitum.* PhD thesis, Bergische Universit"at–Gesamthochschule Wuppertal.

Lenerz, J. 1977. *Zur Abfolge nominaler Satzglieder im Deutschen.* Tübingen: TBL Verlag Gunter Narr.

Meknavin, S., M. Okumura, and H. Tanaka. 1992. A Chart-based Method of ID/LP Parsing with Generalized Discrimination Networks. In *ACL Proceedings, 23th Annual Meeting*, 401—407.

Morawietz, F. ms. A Procedural Treatment for Parsing of HPSG. Master's thesis, Seminar für Sprachwissenschaft, Eberhard-Karls-Universität Tübingen, Tübingen.

Oliva, K. 1992. The proper treatment of word order in HPSG. In *Proceedings of COLING-92*, 184–190.

Partee, B. H., A. ter Meulen, and R. E. Wall. 1990. *Mathematical Methods in Linguistics.* Vol. 30 of Studies in Linguistics and Philosophy. Dordrecht/Boston/London: Kluwer Academic Publishers.

Pollard, C., and I. Sag. in press. *Head-Driven Phrase Structure Grammar.* To be published by University of Chicago Press and CSLI Publications.

Pollard, C. in press. On Head Non Movement. In *Discontinuous Constituency*, ed. H. Arthur and W. Sijtsma. Berlin: Mouton de Gruyter.

Reape, M. 1990. A Theory of Word Order and Discontinous Constituency in West Continental Germanic. In *Parametric Variation in Germanic and Romance: Preliminary Investigations*, ed. E. Engdahl and M. Reape, 25–39.

Edinburgh: Centre for Cognitive Science and Department of Artificial Intelligence: ESPRIT Basic Research Action 3175 DYANA R1.1.A.

Reape, M. ms. Clause Union and Word-Order variation in Germanic. Univerity of Edinburgh.

Seiffert, R. 1987. Chart-Parsing of Unification-Based Grammars with ID/LP-Rules. In *Categories, Polymorphism and Unification,* ed. E. Klein and J. van Benthem, 335–54. Edinburgh/Amsterdam. CCS/ILLI.

Shieber, S. M. 1984. Direct Parsing of ID/LP Grammars. *Linguistics and Philosophy* 7:135–154.

Shieber, S. M. 1985. Using Restriction To Extend Parsing Algorithms for Complex–feature–based Formalisms. In *Proccedings of the 22nd Annual Meeting of the ACL.*

Uszkoreit, H. 1984. *Word Order and Constituent Structure in German.* PhD thesis, University of Texas, Austin. published in revised form in 1987 as Lecture Notes N0. 8, CSLI Lecture Notes Series.

Uszkoreit, H. 1986. Linear precedence in discontinuous constituents: complex fronting in German. Report 86–47, CSLI, Stanford, Calif.