HANDLING LINEAR PRECEDENCE CONSTRAINTS BY UNIFICATION

Judith Engelkamp, Gregor Erbach and Hans Uszkoreit
Universität des Saarlandes, Computational Linguistics, and
Deutsches Forschungszentrum für Künstliche Intelligenz
D-6600 Saarbrücken 11, Germany
engelkamp@coli.uni-sb.de

ABSTRACT

Linear precedence (LP) rules are widely used for stating word order principles. They have been adopted as constraints by HPSG but no encoding in the formalism has been provided. Since they only order siblings, they are not quite adequate, at least not for German. We propose a notion of LP constraints that applies to linguistically motivated branching domains such as head domains. We show a type-based encoding in an HPSG-style formalism that supports processing. The encoding can be achieved by a compilation step.

INTRODUCTION

Most contemporary grammar models employed in computational linguistics separate statements about dominance from those that determine linear precedence. The approaches for encoding linear precedence (LP) statements differ along several dimensions.

Depending on the underlying grammatical theory, different criteria are employed in formulating ordering statements. Ordering constraints may be expressed by referring to the category, grammatical function, discourse rôle, and many other syntactic, semantic, morphological or phonological features.

Depending on the grammar formalism, different languages are used for stating the constraints on permissible linearizations. LP rules, first proposed by Gazdar and Pullum (1982) for GPSG, are used, in different guises, by several contemporary grammar formalisms. In Functional Unification Grammar (Kay 1985) and implemented versions of Lexical Functional Grammar, pattern languages with the power of regular expressions have been utilized.

Depending on the grammar model, LP statements apply within different ordering domains. In most frameworks, such as GPSG and HPSG, the ordering domains are local trees. Initial trees constitute the ordering domain in ID/LP TAGS (Joshi 1987). In current LFG (Kaplan & Zaenen 1988), functional precedence rules apply to functional domains. Reape (1989) constructs word order domains by means of a special union operation on embedded tree domains.

It remains an open question which choices along these dimensions will turn out to be most adequate for the description of word order in natural language.

In this paper we do not attempt to resolve the linguistic issue of the most adequate universal treatment of word order. However we will present a method for integrating word order constraints in a typed feature unification formalism without adding new formal devices.

Although some proposals for the interaction between feature unification and LP constraints have been published (e.g. Seiffert 1991), no encoding has yet been shown that integrates LP constraints in the linguistic type system of a typed feature unification formalism. Linguistic processing with a head-driven phrase structure grammar (HPSG) containing LP constraints has not yet been described in the literature.

Since no implemented NL system has been demonstrated so far that handles partially free word order of German and many other languages in a satisfactory way, we have made an attempt to utilize the formal apparatus of HPSG for a new approach to processing with LP constraints. However, our method is not bound to the formalism of HPSG.

In this paper we will demonstrate how LP constraints can be incorporated into the linguistic type system of HPSG through the use of parametrized types. Neither additional operations nor any special provisions for linear precedence in the processing algorithm are required. LP constraints are applied through regular unification whenever the head combines with a complement or adjunct.

Although we use certain LP-relevant features in our examples, our aproach does not hinge on the selection of specific linguistic criteria for constraining linear order.

Since there is no conclusive evidence to the contrary, we assume the simplest constraint language for formulating LP statements, i.e., binary LP constraints. For computational purposes such constraints are compiled into the type definitions for grammatical categories.

With respect to the ordering domain, our LP constraints differ from the LP constraints commonly assumed in HPSG (Pollard & Sag 1987) in that they

Research for this paper was mainly carried out in the project LILOG supported by IBM Germany. Some of the research was performed in the project DISCO which is funded by the German Federal Ministry for Research and Technology under Grant-No.: ITW 9002. We wish to thank our colleagues in Saarbrücken, three anonymous referees and especially Mark Hepple for their valuable comments and suggestions.

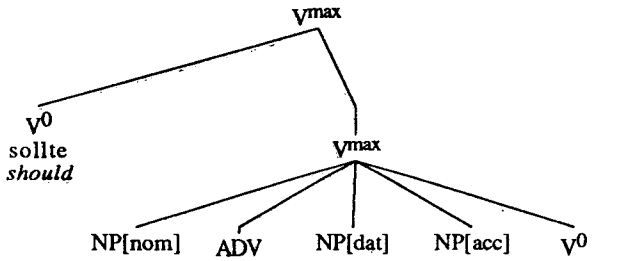
apply to nonsibling constituents in head domains. While LP constraints control the order of nodes that are not siblings, information is accumulated in trees in such a way that it is always possible to detect a violation of an LP constraint locally by checking sibling nodes.

This modification is necessary for the proper treatment of German word order. It is also needed by all grammar models that are on the one hand confined to binary branching structures such as nearly all versions of categorial grammar but that would, on the other hand, benefit from a notion of LP constraints.

Our approach has been tested with small sets of LP constraints. The grammar was written and run in STUF, the typed unification formalism used in the project LILOG.

LINGUISTIC MOTIVATION

This section presents the linguistic motivation for our approach. LP statements in GPSG (Gazdar et al. 1985) constrain the possibility of linearizing immediate dominance (ID) rules. By taking the right-hand sides of ID rules as their domain, they allow only the ordering of sibling constituents. Consequently, grammars must be designed in such a way that all constituents which are to be ordered by LP constraints must be dominated by one node in the tree, so that "flat" phrase structures result, as illustrated in figure 1.



The courier was later supposed to slip a spy the letter.
Figure 1

Uszkoreit (1986) argues that such flat structures are not well suited for the description of languages such as German and Dutch. The main reason[1] is so-called complex fronting, i.e., the fronting of a non-finite verb together with some of its complements and adjuncts as it is shown in (1). Since it is a well established fact that only one constituent can be fronted, the flat structure can account for the German examples in (1), but not for the ones in (2).

(1) sollte der Kurier nachher einem Spion den Brief zustecken
zustecken sollte der Kurier nachher einem Spion den Brief
den Brief sollte der Kurier nachher einem Spion zustecken
einem Spion sollte der Kurier nachher den Brief zustecken
nachher sollte der Kurier einem Spion den Brief zustecken
der Kurier sollte nachher einem Spion den Brief zustecken

(2) **den Brief zustecken** sollte der Kurier nachher einem Spion
einem Spion den Brief zustecken sollte der Kurier nachher
nachher einem Spion den Brief zustecken sollte der Kurier

In the hierarchical tree structure in figure 2, the boxed constituents can be fronted, accounting for the examples in (1) and (2).

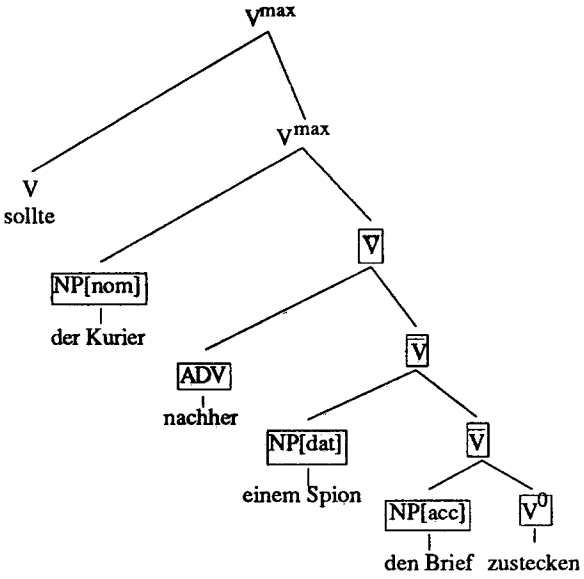


Figure 2

But with this tree structure, LP constraints can no longer be enforced over siblings. The new domain for linear order is a head domain, defined as follows:

A **head domain** consists of the lexical head of a phrase, and its complements and adjuncts.

LP constraints must be respected within a head domain.

An **LP-constraint** is an ordered pair <A,B> of category descriptions, such that whenever a node α subsumed by A and a node β subsumed by B occur within the domain of an LP-rule (in the case of GPSG a local tree, in our case a head domain), α precedes β.

An LP constraint <A,B> is conventionally written as A < B. It follows from the definition that B can never precede A in an LP domain. In the next section, we will show how this property is exploited in our encoding of LP constraints.

ENCODING OF LP CONSTRAINTS

From a formal point of view, we want to encode LP constraints in such a way that

[1] Further reasons are discussed in Uszkoreit (1991b).

- violation of an LP constraint results in unification failure, and
- LP constraints, which operate on head domains, can be enforced in local trees by checking sibling nodes.

The last condition can be ensured if every node in a projection carries information about which constituents are contained in its head domain.

An LP constraint A < B implies that it can never be the case that B precedes A. We make use of this fact by the following additions to the grammar:

- Every category A carries the information that B must not occur to its left.
- Every category B carries the information A must not occur to its right.

This duplication of encoding is necessary because only the complements/adjuncts check whether the projection with which they are combined contains something that is incompatible with the LP constraints. A projection contains only information about which constituents are contained in its head domain, but no restrictions on its left and right context[2].

In the following example, we assume the LP-rules A<B and B<C. The lexical head of the tree is X^0, and the projections are $\overline{X}$, and X^{max}. The complements are A, B and C. Each projection contains information about the constituents contained in it, and each complement contains information about what must not occur to its left and right. A complement is only combined with a projection if the projection does not contain any category that the complement prohibits on its right or left, depending on which side the projection is added.

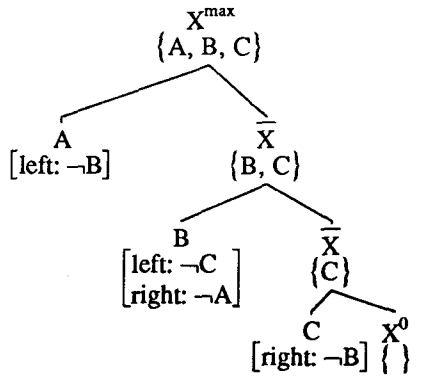


Figure 3

Having now roughly sketched our approach, we will turn to the questions of how a violation of LP constraints results in unification failure, how the information associated with the projections is built up, and what to do if LP constraints operate on feature structures rather than on atomic categories.

[2]Alternatively, the projections of the head could as well accumulate the ordering restrictions while the arguments and adjuncts only carry information about their own LP-relevant features. The choice between the alternatives has no linguistic implications since it only affects the grammar compiled for processing and not the one written by the linguist.

VIOLATION OF LP-CONSTRAINTS AS UNIFICATION FAILURE

As a conceptual starting point, we take a number of LP constraints. For the expository purposes of this paper, we oversimplifiy and assume just the following four LP constraints:

nom < dat	(nominative case precedes dative case)
nom < acc	(nominative case precedes accusative case)
dat < acc	(dative case precedes accusative case)
pro < nonpro	(pronominal NPs precede non-pronominal NPs)

Figure 4

Note that nom, dat, acc, pro and nonpro are not syntactic categories, but rather values of syntactic features. A constituent, for example the pronoun *ihn* (him) may be both pronominal and in the accusative case. For each of the above values, we introduce an extra boolean feature, as illustrated in figure 5.

[NOM *bool*
DAT *bool*
ACC *bool*
PRO *bool*
NON-PRO *bool*]

Figure 5

Arguments encode in their feature structures what must not occur to their left and right sides. The dative NP *einem Spion* (a spy), for example, must not have any accusative constituent to its left, and no nominative or pronominal constituent to its right, as encoded in the following feature structure. The feature structures that constrain the left and right contexts of arguments only use '-' as a value for the LP-relevant features.

[LEFT [ACC -]
RIGHT [NOM -
PRO -]]

Figure 6: Feature Structure for *einem Spion*

Lexical heads, and projections of the head contain a feature LP-STORE, which carries information about the LP-relevant information occuring within their head domain (figure 7).

[LP-STORE [NOM -
DAT -
ACC -
PRO -
NON-PRO -]]

Figure 7: empty LP-STORE

In our example, where the verbal lexical head is not affected by any LP constraints, the LP-STORE contains the information that no LP-relevant features are present.

For a projection like *einen Brief zusteckt* (a letter[acc] slips), we get the following LP-STORE.

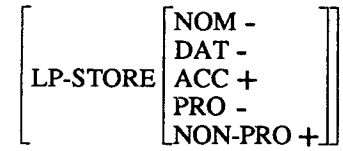


Figure 8: LP-STORE of *einen Brief zusteckt*

The NP *einem Spion* (figure 6) can be combined with the projection *einen Brief zusteckt* (figure 8) to form the projection *einem Spion einen Brief zusteckt* (a spy[dat] a letter[acc] slips) because the RIGHT feature of *einem Spion* and the LP-STORE of *einen Brief zusteckt* do not contain incompatible information, i.e., they can be unified. This is how violations of LP constraints are checked by unification. The projection *einem Spion einen Brief zusteckt* has the following LP-STORE.

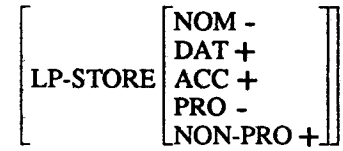


Figure 9: LP-STORE of *einem Spion einen Brief zusteckt*

The constituent *ihn zusteckt* (figure 10) could not be combined with the non-pronominal NP *einem Spion* (figure 6).

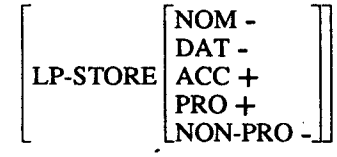


Figure 10: LP-STORE of *ihn zusteckt*

In this case, the value of the RIGHT feature of the argument *einem Spion* is not unifiable with the LP-STORE of the head projection *ihn zusteckt* because the feature PRO has two different atoms (+ and -) as its value. This is an example of a violation of an LP constraint leading to unification failure.

In the next section, we show how LP-STOREs are manipulated.

MANIPULATION OF THE LP-STORE

Since information about constituents is added to the LP-STORE, it would be tempting to add this information by unification, and to leave the initial LP-STORE unspecified for all features. This is not possible because violation of LP constraints is also checked by unification. In the process of this unification, values for features are added that may lead to unwanted unification failure when information about a constituent is added higher up in the tree.

Instead, the relation between the LP-STORE of a projection and the LP-STORE of its mother node is encoded in the argument that is added to the projection. In this way, the argument "changes" the LP-STORE by "adding information about itself". Arguments therefore have the additional features LP-IN and LP-OUT. When an argument is combined with a projection, the projection's LP-STORE is unified with the argument's LP-IN, and the argument's LP-OUT is the mother node's LP-STORE. The relation between LP-IN and LP-OUT is specified in the feature structure of the argument, as illustrated in figure 11 for the accusative pronoun *ihn*, which is responsible for changing figure 7 into figure 10. No matter what the value for the features ACC and PRO may be in the projection that the argument combines with, it is '+' for both features in the mother node. All other features are left unchanged[3].

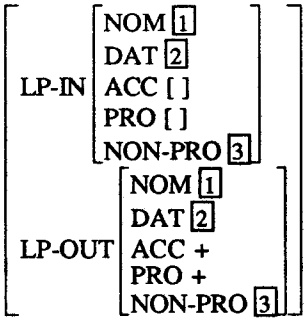


Figure 11

Note that only a '+' is added as value for LP-relevant features in LP-OUT, never a '-'. In this way, only positive information is accumulated, while negative information is "removed". Positive information is never "removed".

Even though an argument or adjunct constituent may have an LP-STORE, resulting from LP constraints that are relevant within the constituent, it is ignored when the constituent becomes argument or adjunct to some head. Our encoding ensures that LP constraints apply to all head domains in a given sentence, but not across head domains.

It still remains to be explained how complex phrases that become arguments receive their LP-IN, LP-OUT, RIGHT and LEFT features. These are specified in the lexical entry of the head of the phrase, but they are ignored until the maximal projection of the head becomes argument or adjunct to some other head. They must, however, be passed on unchanged from the lexical head to its maximal projection. When

[3]Coreference variables are indicated by boxed numbers. [ ] is the feature structure that contains no information (TOP) and can be unified with any other feature structure.

the maximal projection becomes an argument/adjunct, they are used to check LP constrains and "change" the LP-STORE of the head's projection.

Our method also allows for the description of head-initial and head-final constructions. In German, for example, we find prepositions (e.g. *für*), postpositions (e.g. *halber*) and some words that can be both pre- and postpostions (e.g. *wegen*).

The LP-rules would state that a postposition follows everything else, and that a preposition precedes everything else.

[PRE +] < [ ]

[ ] < [POST +]

Figure 12

The information about whether something is a preposition or a postposition is encoded in the lexical entry of the preposition or postposition. In the following figure, the LP-STORE of the lexical head contains also positive values.

$$\begin{bmatrix}\text{LP-STORE}\begin{bmatrix}\text{POST} & +\\ \text{PRE} & -\end{bmatrix}\end{bmatrix}$$

Figure 13: part of the lexical entry of a postposition

$$\begin{bmatrix}\text{LP-STORE}\begin{bmatrix}\text{POST} & -\\ \text{PRE} & +\end{bmatrix}\end{bmatrix}$$

Figure 14: part of the lexical entry of a preposition

A word that can be both a preposition and a postposition is given a disjunction of the two lexical entries:

$$\begin{bmatrix}\text{LP-STORE}\left\langle\begin{bmatrix}\text{POST} & -\\ \text{PRE} & +\end{bmatrix},\begin{bmatrix}\text{POST} & +\\ \text{PRE} & -\end{bmatrix}\right\rangle\end{bmatrix}$$

Figure 15

All complements and adjuncts encode the fact that there must be no preposition to their right, and no postposition to their left.

$$\begin{bmatrix}\text{RIGHT}[\text{PRE} & -]\\ \text{LEFT}[\text{POST} & -]\end{bmatrix}$$

Figure 16

The manipulation of the LP-STORE by the features LP-IN and LP-OUT works as usual.

The above example illustrates that our method of encoding LP constraints works not only for verbal domains, but for any projection of a lexical head. The order of quantifiers and adjectives in a noun phrase can be described by LP constraints.

INTEGRATION INTO HPSG

In this section, our encoding of LP constraints is incorporated into HPSG (Pollard & Sag 1987). We deviate from the standard HPSG grammar in the following respects:

- The features mentioned above for the encoding of LP-constraints are added.
- Only binary branching grammar rules are used.
- Two new principles for handling LP-constraints are added to the grammar.

Further we shall assume a set-valued SUBCAT feature as introduced by Pollard (1990) for the description of German. Using sets instead of lists as the values of SUBCAT ensures that the order of the complements is only constrained by LP-statements.

In the following figure, the attributes needed for the handling of LP-constraints are assigned their place in the HPSG feature system.

$$\begin{bmatrix}\text{SYNSEM}\mid\text{LOC}\begin{bmatrix}\text{HEAD}\begin{bmatrix}\text{LP-IN}[\ ]\\ \text{LP-OUT}[\ ]\\ \text{LEFT}[\ ]\\ \text{RIGHT}[\ ]\end{bmatrix}\\ \text{LP-STORE}[\ ]\end{bmatrix}\end{bmatrix}$$

Figure 17

The paths SYNSEM|LOC|HEAD|{LP-IN,LP-OUT,RIGHT,LEFT} contain information that is relevant when the constituents becomes an argument/adjunct. They are HEAD features so that they can be specified in the lexical head of the constituent and are percolated via the Head Feature Principle to the maximal projection. The path SYNSEM|LOC|LP-STORE contains information about LP-relevant features contained in the projection dominated by the node described by the feature structure. LP-STORE can obviously not be a head feature because it is "changed" when an argument or adjunct is added to the projection.

In figures 18 and 19, the principles that enforce LP-constraints are given[4]. Depending on whether the head is to the right or to the left of the complement/adjunct, two versions of the principle are distinguished. This distinction is necessary because linear order is crucial. Note that neither the HEAD features of the head are used in checking LP constraints, nor the LP-STORE of the complement or adjunct.

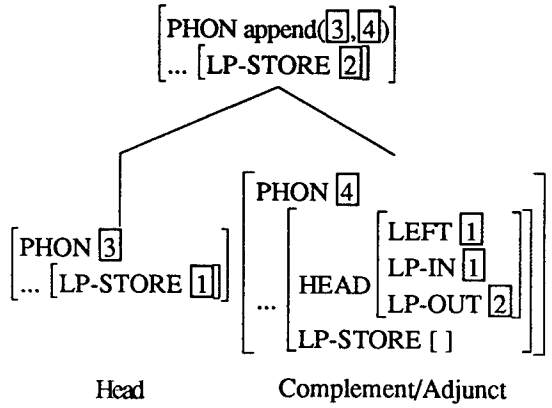


Figure 18: Left-Head LP-Principle

[4]The dots (...) abbreviate the path SYNSEM|LOCAL

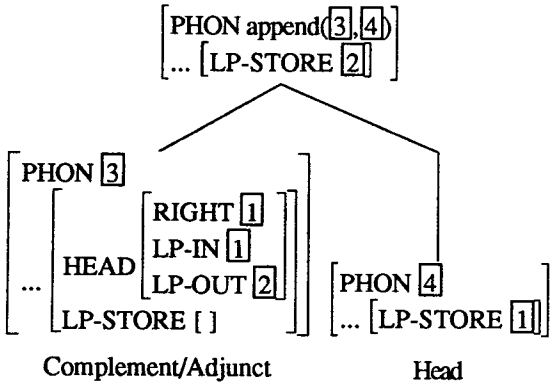


Figure 19: Right-Head LP-Principle

In the following examples, we make use of the parametrized type notation used in the grammar formalism STUF (Dörre 1991). A parametrized type has one or more parameters instantiated with feature structures. The name of the type (with its parameters) is given to the left of the := sign, the feature structure to the right.

In the following we define the parametrized types nom(X,Y), dat(X,Y), pro(X,Y), and non-pro(X,Y), where X is the incoming LP-STORE and Y is the outgoing LP-STORE.

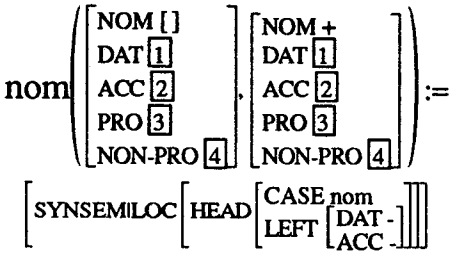


Figure 20

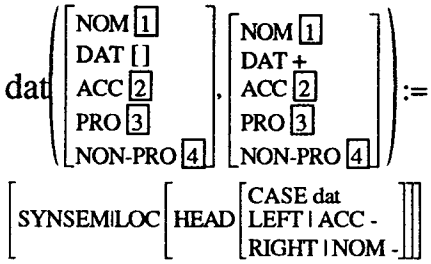


Figure 21

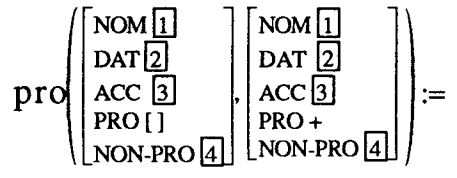


Figure 22

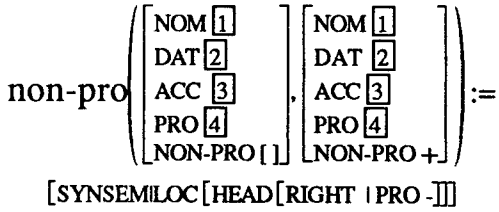


Figure 23

The above type definitions can be used in the definition of lexical entries. Since the word *ihm*, whose lexical entry[5] is given in figure 24, is both dative case and pronominal, it must contain both types. While the restrictions on the left and right context invoked by dat/2 and pro/2 can be unified[6], matters are not that simple for the LP-IN and LP-OUT features. Since their purpose is to "change" rather than to "add" information, simple unification is not possible. Instead, LP-IN of *ihm* becomes the incoming LP-STORE of dat/2, the outgoing LP-STORE of dat/2 becomes the incoming LP-STORE of pro/2, and the outgoing LP-STORE of pro/2 becomes LP-OUT of *ihm*, such that the effect of both changes is accumulated.

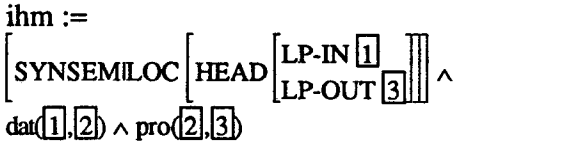


Figure 24: lexical entry for *ihm*

After expansion of the types, the following feature structure results. Exactly the same feature structure had been resulted if dat/2 and pro/2 would have been exchanged in the above lexical entry (pro(1,2) ∧ dat(2,3)), because the effect of both is to instantiate a '+' in LP-OUT.

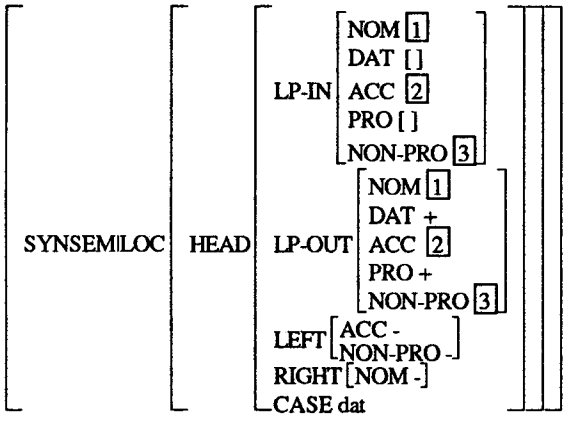


Figure 25: expanded lexical entry for *ihm*

[5]Only the information which is relevant for the processing of LP constraints is included in this lexical entry.

[6]dat/2 means the type dat with two parameters.

The next figure shows the lexical entry for a non-pronominal NP, with a disjunction of three cases.

Peter :=

$$\left[\text{SYNSEM|LOC}\left[\text{HEAD}\left[\begin{array}{l}\text{LP-IN } \boxed{1}\\ \text{LP-OUT } \boxed{3}\end{array}\right]\right]\right] \wedge$$

$$(\text{nom}(\boxed{1},\boxed{2}) \vee \text{dat}(\boxed{1},\boxed{2}) \vee \text{acc}(\boxed{1},\boxed{2})) \wedge \text{non-pro}(\boxed{2},\boxed{3})$$

Figure 26

COMPILATION OF THE ENCODING

As the encoding of LP constraints presented above is intended for processing rather than grammar writing, a compilation step will initialize the lexical entries automatically according to a given grammar including a separated list of LP-constraints. Consequently the violation of LP-constraints results in unification failure. For reasons of space we only present the basic idea.

The compilation step is based on the assumption that the features of the LP-constraints are morphologically motivated, i.e. appear in the lexicon. If this is not the case (for example for focus, thematic roles) we introduce the feature with a disjunction of its possible values. This drawback we hope to overcome by employing functional dependencies instead of LP-IN and LP-OUT features.

For each side of an LP-constraint we introduce boolean features. For example for [A: v] < [B: w] we introduce the features a_v and b_w. This works also for LP-constraints involving more than one feature such as

$$\left[\begin{array}{l}\text{PRO } +\\ \text{CASE acc}\end{array}\right] < \left[\begin{array}{l}\text{PRO } +\\ \text{CASE dat}\end{array}\right]$$

For encoding the possible combinations of values for the participating features, we introduce binary auxiliary features such as pro_plus_case_acc, because we need to encode that there is at least a single constituent which is both pronominal and accusative.

Each lexical entry has to be modified as follows:

1. A lexical entry that can serve as the head of a phrase receives the additional feature LP-STORE.

2. An entry that can serve as the head of a phrase and bears LP-relevant information, i.e. a projection of it is subsumed by one side of some LP-constraint, has to be extended by the features LP-IN, LP-OUT, LEFT, RIGHT.

3. The remaining entries percolate the LP information unchanged by passing through the information via LP-IN and LP-OUT.

The values of the features LEFT and RIGHT follow from the LP-constraints and the LP-relevant information of the considered lexical entry.

The values of LP-STORE, LP-IN and LP-OUT depend on whether the considered lexical entry bears the information that is represented by the boolean feature (attribute A with value v for boolean feature a_v).

	entry bears the information	entry doesn't bear the information
LP-STORE	+	–
LP-IN	TOP	new variable x
LP-OUT	+	coreference to x

CONCLUSION

We have presented a formal method for the treatment of LP constraints, which requires no addition to standard feature unification formalisms. It should be emphasized that our encoding only affects the compiled grammar used for the processing. The linguist does not lose any of the descriptive means nor the conceptual clarity that an ID/LP formalism offers. Yet he gains an adequate computational interpretation of LP constraints.

Because of the declarative specification of LP constraints, this encoding is neutral with respect to processing direction (parsing-generation). It does not depend on specific strategies (top-down vs. bottom-up) although, as usual, some combinations are more efficient than others. This is an advantage over the formalization of unification ID/LP grammars in Seiffert (1991) and the approach by Erbach (1991). Seiffert's approach, in which LP constraints operate over siblings, requires an addition to the parsing algorithm, by which LP constraints are checked during processing to detect violations as early as possible, and again after processing, in case LP-relevant information has been added later by unification. Erbach's approach can handle LP constraints in head domains by building up a list of constituents over which the LP constraints are enforced, but also requires an addition to the parsing algorithm for checking LP constraints during as well as after processing.

Our encoding of LP constraints does not require any particular format of the grammar, such as left- or right-branching structures. Therefore it can be incorporated into a variety of linguistic analyses. There is no need to work out the formal semantics of LP constraints because feature unification formalisms already have a well-defined formal semantics.

Reape (1989) proposes a different strategy for treating partially free word order. His approach also permits the application of LP constraints across local trees. This is achieved by separating word order variation from the problem of building a semantically motivated phrase structure. Permutation across constituents can be described by merging the fringes (terminal yields) of the constituents using the operation of sequence union. All orderings imposed on the two merged fringes by LP constraints are preserved in the merged fringe.

Reape treats clause union and scrambling as permutation that does not affect constituent structure. Although we are intrigued by the elegance and descriptive power of Reape's approach, we keep our bets with our more conservative proposal. The main problem we see with Reape's strategy is the additional

burden for the LP component of the grammar. For every single constituent that is scrambled out of some clause into a higher clause, the two clauses need to be sequence-unioned. A new type of LP constraints that refer to the position of the constituents in the phrase or dependency structure is employed for ensuring that the two clauses are not completely interleaved. Hopefully future research will enable us to arrive at better judgements on the adequacy of the different approaches.

Pollard (1990) proposes an HPSG solution to German word order that lets the main verb first combine with some of its arguments and adjuncts in a local tree. The resulting constituent can be fronted. The remaining arguments and adjuncts are raised to the subcategorization list[7] of the auxiliary verb above the main verb. Yet, even if a flat structure is assumed for both the fronted part of the clause and the part remaining in situ as in (Pollard 1990), LP constraints have to order major constituents across the two parts. For a discussion, see Uszkoreit (1991b).

Uszkoreit (1991b) applies LP principles to head domains but employs a finite-state automaton for the encoding of LP constraints. We are currently still investigating the differences between this approach and the one presented here.

Just as most other formal appraoches to linear precedence, we treat LP-rules as absolute constraints whose violation makes a string unacceptable. Sketchy as the data may be, they suggest that violation of certain LP-constraints merely makes a sentence less acceptable. Degrees of acceptability are not easily captured in feature structures as they are viewed today. In terms of our theory, we must ensure that the unification of the complement's or adjunct's left or right context restriction with the head's LP-STORE does not fail in case of a value clash, but rather results in a feature structure with lower acceptability than the structure in which there is no feature clash. But until we have developed a well-founded theory of degrees of acceptability, and explored appropriate formal means such as weighted feature structures, as proposed in (Uszkoreit 1991a), we will either have to ignore ordering principles or treat them as absolute constraints.

REFERENCES

[Dörre 1991]
Jochen Dörre. The Language of STUF. In: Herzog, O. and Rollinger, C.-R. (eds.): Text Understanding in LILOG. Springer, Berlin.

[Erbach 1991]
Gregor Erbach. A flexible parser for a linguistic experimentation environment. In: Herzog, O. and Rollinger, C.-R. (eds.): *Text Understanding in LILOG.* Springer, Berlin.

[Gazdar & Pullum 1982]
Gerald Gazdar, G. K. Pullum. Generalized Phrase Structure Grammar. A Theoretical Synopsis. Indiana Linguistics Club, Bloomington, Indiana.

[Gazdar et al. 1985]
Gerald Gazdar, Ewan Klein, G. K. Pullum, Ivan Sag. Generalized Phrase Structure Grammar. Basil Blackwell, Oxford, UK

[Joshi 1987]
A. K. Joshi. Word-Over Variation in Natural Language Generation. In: Proceedings of AAAI-87, 550-555

[Kaplan & Zaenen 1988]
R. M. Kaplan, A. Zaenen. Functional Uncertainty and Functional Precedence in Continental West Germanic. In: H. Trost (ed.), Proceedings of 4. Österreichische Artificial-Intelligence-Tagung. Springer, Berlin.

[Kay 1985]
Martin Kay. Parsing in Functional Unification Grammar. In: D. Dowty, L. Karttunen and A. Zwicky (eds.), Natural Language Parsing. Cambridge University Press, Cambidge, UK.

[Pollard 1990]
Carl Pollard. On Head Non-Movement. In: *Proceedings of the Symposium on Discontinuous Constituency*, Tilburg, ITK.

[Pollard & Sag 1987]
Carl Pollard, Ivan Sag. Information-based syntax and semantics. Vol. 1: Fundamentals. CSLI Lecture Notes No. 13, Stanford, CA.

[Reape 1989]
Mike Reape. A Logical Treatment of Semi-Free Word Order and Bounded Discontinuous Constituency. In: *Proceedings of the 4th Meeting of the European Chapter of the ACL*, Manchester, UK.

[Seiffert 1991]
Roland Seiffert. Unification-ID/LP Grammars: Formalization and Parsing. In: Herzog, O. and Rollinger, C.-R. (eds.): *Text Understanding in LILOG.* Springer, Berlin.

[Uszkoreit 1986]
Hans Uszkoreit. Linear Precedence in Discontinuous Constituents: Complex Fronting in German. CSLI Report CSLI-86-47. Stanford, CA.

[Uszkoreit 1991a]
Hans Uszkoreit. Strategies for Adding Control Information to Declarative Grammars. Proceedings of ACL '91, Berkeley.

[Uszkoreit 1991b]
Hans Uszkoreit. Linear Prededence in Head Domains. Workshop on HPSG and German, Saarbrücken, FRG (Proceedings to be published)

[7]Actually, in Pollard's proposal the subcat feature is set-valued.