A Constructive View of GPSG or How to Make It Work

Stephan BUSEMANN
Christa HAUENSCHILD

Technical University of Berlin
Institute for Software and Theoretical Computer Science
Project Group KIT
Sekr. FR 5-12
Franklinstr. 28/29
D-1000 Berlin 10
E-mail: busemann@db0tui11.bitnet

Abstract

Using the formalism of generalized phrase structure grammar (GPSG) in an NL system (e.g. for machine translation (MT)) is promising since the modular structure of the formalism is very well suited to meet some particular needs of MT. However, it seems impossible to implement GPSG in its 1985 version straightforwardly. This would involve a vast overgeneration of structures as well as processes to filter out everything but the admissible tree(s). We therefore argue for a constructive version of GPSG where information is gathered in subsequent steps to produce syntactic structures. As a result, we consider it necessary to incorporate procedural aspects into the formalism in order to use it as a linguistic basis for NL parsing and generation. The paper discusses the major implications of such a modified view of GPSG. [1]

1 Introduction

Any attempt to build a multi-lingual MT system as in EUROTRA [King, Perschke 1987] must provide for massive modularization in order to avoid developing 9 parsers, 9 generators and 72 transfer components for the 9 languages involved, not to mention the different but redundant formulations of linguistic knowledge embodied in them. The most obvious approach consists in developing one single parser, one single generator, and one single transfer component, the first two being capable of dealing with grammars for different languages and the latter with transfer rules for different pairs of languages. Moreover, an MT system must be based on a linguistically justified theory of grammar. This theory has to be implemented in the system, where it determines the construction of a syntactic representation of a sentence during the parsing of some input string as well as during the generation based on the output of the transfer component.

The theory of GPSG (see [Gazdar et al. 1985], henceforth: [GKPS]) has been tested for its usefulness for MT [Hauenschild/Busemann 1988]. It offers the high degree of modularity that is required. For instance, an implementation of the GPSG formalism would be able to run with different grammars, and linguistic generalizations would either evolve from the formalism (in the case of universals), or be expressible within the grammars (in the case of language-specific generalizations). We shall distinguish between the formalism and the grammar in the following way; the formalism consists of the Feature Instantiation Principles (FIPs), the formal definition of syntactic features, categories, Feature Co-occurrence Restrictions (FCRs), Immediate Dominance (ID) rules, Linear Precedence (LP) statements, admissible trees, etc. The grammars consist of actual sets of ID rules, LP statements, FCRs, and the lexicon.

However, a closer look at the axiomatic way GPSG has been defined reveals severe problems for an implementation of GPSG. In the next section we shall outline these problems, and in section 3 present our change in perspective towards a GPSG formalism that overcomes these problems. Some consequences of this are discussed in the last section.

The rest of the paper concentrates on GPSG and its use for processing of representations of natural language sentences. Nothing can be said here about the necessity of including textual knowledge for translation or about the transfer step itself (but cf. [Hauenschild 1986]).

1 This work has been developed in the project KIT-FAST (KIT = Künstliche Intelligenz und Textverstehen (Artificial Intelligence and Text Understanding); FAST = Functor Argument Structure for Translation), which constitutes the Berlin component of the complementary research project of Eurotra-D. It receives grants by the Federal Minister for Research and Technology under contract 1013211.

2 Problems With the Implementation of GPSG

In this section we want to justify why we had to develop a constructive version of the GPSG formalism although it might seem that the "classical" version of it (as defined in [GKPS]) can be implemented. We want to show that this is only true in theory but not in practice.

What would it really amount to if we tried to implement the axiomatic version of GPSG in a straightforward way? In order to find all admissible trees corresponding to a given sentence, we would have to do the following things for every local tree (i.e. trees of depth 1):

- build every possible extension for every category in an ID rule, which means that every feature that is not specified in the rule may be either absent or specified by any of its values,
- filter out the illegal categories with the aid of the FCRs,
- build all the possible projections of ID rules with the remaining legal categories, thereby creating every possible order of the daughters,
- filter out those combinations of categories that are inadmissible according to the Foot Feature Principle (FFP), Control Agreement Principle (CAP) or Head Feature Convention (HFC),
- filter out those projections that are unacceptable because of some category contradicting a Feature Specification Default (FSD),
- filter out all those projections that contradict any LP statement applicable to the daughters.

After this, the subset of admissible local trees has to be identified which yields the desired complex structures in the following way: two (locally) admissible trees may be combined into a larger tree iff one of the daughters of one of them is identical with the mother of the other one.

The whole process can be regarded as divided up into three major steps. The first step consists in constructing all the possible projections (possible according to ID rules and FCRs). The second step consists in filtering out local trees that are not admissible according to the restrictions imposed on them by the FIPs, the FSDs and the LP statements. Though these devices are not filters in the Chomskyan sense,[2] they behave in an analogous way by preventing previously generated structures from becoming locally admissible trees. The last step consists in forming complex structures out of locally admissible trees.

In order to show the complexity of such an approach, it is necessary to give a rough idea of what the first step really amounts to; it yields a combinatorial explosion of the set of categories. Assuming the 25 atomic and the 4 category-valued features defined for the English grammar in [GKPS], a lower bound for the number of categories to be checked by the FCRs is 10^{774} [Ristad 1986].

The second of the above mentioned steps is not trivial either, though its problems might be solvable after all. For a purely axiomatic view of the GPSG formalism it may be permissible to neglect the order in which the different filtering components are to be applied, although there seem to be some problems with the definitions of the different FIPs with respect to their logical independence of each other. For an effective implementation however, the ordering problem becomes crucial. There are some hints in [GKPS] referring to interdependencies between the different filters, but they are not fully specified. The most problematic case is the order in which the HFC and the CAP have to be applied:

- the HFC seems to presuppose the effects of the CAP (and of the FFP) because it must not force feature specifications that are excluded by the CAP on categories in local trees;
- the CAP presupposes the HFC in the sense that it is based on semantic types, which are dependent on HEAD features, the distribution of which is in turn governed by the HFC.

One possible way out of this dilemma is suggested in [Shieber 1986], but it is based on the assumption that HEAD features may be split up into two disjoint sets: those HEAD features which are prerequisites for the assignment of semantic types and thus for the applicability of the CAP, and those HEAD features that can safely be applied after the CAP has done its work. However, it is not clear whether such a distribution is possible. Of course, you can always make your ID rules much more informative with respect to feature specifications than is suggested in [GKPS] and thereby guarantee a proper functioning of the FIPs; but that would probably not be in the spirit of GPSG, where the main point is to capture the universal as well as the language-specific generalizations.

There are a number of problems with the CAP; we want to outline just one of them, which has led us to modify this principle. The definition of control in [GKPS] implicitly restricts the functioning of CAP to structures where the functor has no more than one argument (with the exception of those very special cases of control mediators). This cannot be seen from the definition of control [GKPS:88] alone, but may be derived from the interaction of this definition with the conditions on correct type assignment that are imposed on syntactic structures by the principle of functional realization [GKPS, chapter 10]: it follows from both parts of the theory taken together that a functor can be controlled by its argument only in the case where there is no further argument; otherwise the functor would have to be of a type that differs from what is assumed in the definition of control (intuitively, the type of a functor depends on how many arguments the functor takes).

2 This was pointed out to us by John Nerbonne (electronic mail).

This difficulty seems quite hard to cope with; if we assume rather flat structures (as we do, on independent grounds, in our German syntax [Preuß 1987], see also [Uszkoreit 1984]), then it is not clear which of the different arguments of a functor is to control it; in the case of subject-predicate agreement in German, the subject would have to be marked as the controller, which can hardly be done on the basis of the semantic types alone (because there seems to be no semantic reason to distinguish subjects and objects by their semantic type unless we treat subjects as functors operating on VPs as arguments, which would reverse the control relation between them and thus cause all sorts of other problems). The only possibility we can conceive of would be analogous to the concept of argument order as defined in [GKPS] in order to obtain correctly the interpretations of direct and indirect objects, but this is a language-particular concept (cf. [GKPS:214], which would not fit into a universal principle.

3 A Constructive View of GPSG

As the previous section shows, the GPSG formalism in its original version is not suitable for computer implementation. From a processing point of view, it is an obvious requirement that the components of GPSG should only construct the well-formed categories and trees, i.e. no garbage should be produced. In order to utilize GPSG for parsing and generation in a computer system, a change in perspective becomes necessary; instead of deciding for all fully specified categories and all local trees whether they are legal or admissible respectively, we start from a highly underspecified local tree that is admitted by an ID rule and gather information by subsequently applying FCRs and FIPs. Eventually we shall have a fully specified local tree that is admissible by definition.

We shall call this view of GPSG *constructive* since it allows for the construction rather than the selection of a syntactic structure. In a constructive version of GPSG, FCRs and FIPs mainly act as principles of feature transport rather than of feature distribution.

One of the most important questions for the constructive version is in what order the components of GPSG have to be applied. Since each of them may add further feature specifications to a category in a local tree, the order of application ought to depend on what information must be present for a component to work properly. This can be determined in general by using a monotonic operation such as unification for making categories more and more specific.

This has led us to dispense with any assertions about categories as they are often used in [GKPS]. For instance, the predicate ~ with the meaning that some feature is undefined (i.e. it is not contained in the category) is replaced by a feature value, ~, which is subject to unification. We shall thus say that a feature *f* is undefined if it is specified as <*f*, ~>.

The predicative character of FCRs is also modified towards a functional one by including the assignment of values to features. Formally, an FCR is written *cat1* ⊃ *cat2*, where *cat1* and *cat2* are categories. An FCR applies to a category *C* iff *C* is an extension of *cat1*. *C* must unify with *cat2*, otherwise *C* is not legal.

Let us now discuss the role of the FIPs in a constructive version. We shall start with HFC. In [GKPS], HFC is based on the free feature specification sets, which are utilized to prevent HFC from rejecting local trees because of HEAD features specified differently at the mother and the head daughter(s) by virtue of ID rules, FCRs, the FFP, or the CAP. To generate these sets would again require all possible projections from an ID rule to be produced. As was shown in the previous section, this must be avoided if a computer implementation is to be supplied.

From the constructive point of view we suggest that the effect of using the free feature specification sets can be attained by ensuring that for a local tree, the work of the FCRs, the FFP and the CAP has been completed before HFC comes into play. HFC then assures that the so far unspecified HEAD features at the mother are identical with the corresponding HEAD feature specifications at the head daughter(s) and vice versa thereby never rejecting a local tree[3]. HFC proceeds as follows; every head daughter that can unify with its mother with respect to the set of HEAD features will do so. Typically, HEAD includes features for verb form or clause structure. A constituent is marked as head by a binary feature, *head*, which is specified in the ID rules, thus replacing the meta-notation H in [GKPS], the meaning of which is completely dependent on its context.

This way HFC is supposed to work in an equally general, but much simpler, fashion than it was possible with the definition in [GKPS]. Moreover, HFC is capable of coping with multiple heads used for the treatment of certain coordination phenomena; feature specifications are found in the coordinated head daughters, the HEAD feature in question has to be undefined at the mother. This parallels the way multiple heads are treated in [GKPS].

The requirement that the CAP be prior to HFC raises, however, the problem that the CAP cannot be based on semantic types anymore because it is HFC which might provide the major feature specifications necessary to determine the type of a constituent. Moreover, to be applicable to local trees with more than one argument (in those cases where no control mediator is present), the CAP had

[3] After HFC has been applied to a local tree, FCRs may become applicable that were not before, which in turn should cause the HFC to resume its work etc. until nothing is specified anymore. Whether this repetition must actually occur, depends on how the grammar is formulated.

to be reformulated, and its place is taken by a purely syntactic mechanism, the Agreement Principle (AP), which is defined as follows [Weisweber 1987]; every daughter in a local tree that is marked for agreement must unify with its mother with respect to a subset of features, called AGR. If an AGR feature is undefined, it is ignored by the AP. Any local tree violating the AP is rejected. AGR typically contains features for case, gender, person, or number. A constituent is marked for agreement by a binary feature, *agr*, that is specified through FCRs, e.g. {<*cas*, nom>} ⊃ {<*agr*, +>} and {<*vform*, fin>} ⊃ {<*agr*, +>}. The AP together with HFC provides for subject-verb agreement on the basis of these FCRs. This way of coping with agreement phenomena foregoes with any notion of control. There are no semantic types involved; what agrees with what need not be stated explicitly, it is simply the consequence of the interplay of FCRs, AP, HFC, and the definition of the feature sets AGR and HEAD. Note that the AP does not presuppose HEAD feature specifications and can thus be prior to HFC.

However, the AP as defined above cannot account for the fact that a category may participate in some agreement relations, but not in others (in 'raising' constructions a direct object may have to agree with a reflexive pronoun, but not with the finite verb). A more sophisticated version of the AP, which is presently being developed, is based on different kinds of *agr* values (e.g. agr1 and agr2 instead of +). A direct object, as well as the reflexive, is then specified with <*agr*, agr2> whereas subject and finite verb both have <*agr*, agr1>. The revised AP requires categories containing the same *agr* specification to unify with respect to AGR as described above.

This approach allows a category to contain feature specifications arising from different agreement relations. An important hypothesis underlying the revised AP is that this will only be necessary if that category contains, by virtue of an ID rule, category-valued features, which can by themselves be specified for *agr*. These features are also inspected by the revised AP in order to find members of some agreement relation in a local tree. Figure 1 contains a local tree, (3), with the feature *slash* (denoted by '/') specified at the mother as an accusative NP by an ID rule. This expresses the fact that a direct object is missing in local tree (3). The revised AP uses the AGR specifications of the *slash* value to establish agreement between the direct object and the reflexive pronoun. The AGR specifications of the S, on the other hand, are used to ensure subject-verb agreement.

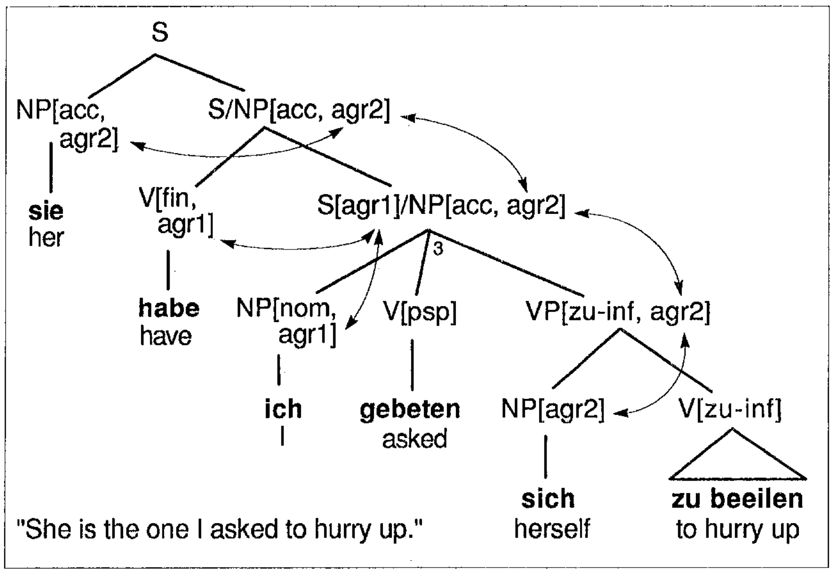

Fig. 1: Establishing Different Agreement Relations

Note that this way of including category-valued features specified in ID rules is independent of which syntactic structures are used to describe a language, rather the function of category-valued features as indicators of long distance relations is utilized.

The feature *agr* can still be specified by virtue of FCRs, though there seem to be some characteristic exceptions where the value is better provided within the ID rules. For instance, a VP should not always contain <*agr*, agr2>, as in figure 1, because in the case of 'equi' verbs it would have to agree with the subject. [4]

[4] This relational information cannot be derived from the different subcategorizations of 'raising' and 'equi' verbs alone.

Let us conclude the discussion of the FIPs with the FFP, the functioning of which has by and large been taken over from [GKPS]. A special treatment is necessary for the value ~. All daughters unify with the mother with respect to a set of FOOT features, provided that the values are not specified in the ID rules. Daughters that are undefined with respect to some FOOT feature are ignored by the FFP unless the FOOT feature is undefined at every daughter or at the mother; in that case the FFP requires all constituents to be undefined with respect to that FOOT feature. If a local tree violates the FFP it is rejected.

The FFP is only dependent on the ID rules and is thus able to be the first FIP to apply. It is in fact prior to the AP since its

point, we shall look at two rather obvious strategies, on which is used in the Berlin GPSG system [Hauensc Busemann 1988] for parsing and the other for generation.

The first one constructs the tree in a bottom-up man thereby *reducing* admissible local trees by unifying t mothers with the daughters of another local tree. The botto up strategy starts from lexical categories, which are admissi by the lexicon. Each reduction step is followed by application of the FIPs to the newly created local tree. Thus information contained in the lexical categories is percolated higher levels of the tree, thereby constraining the set of furt reduction steps allowed by the grammar. This strategy is us within the parser in the GPSG system [Weisweber 1987].

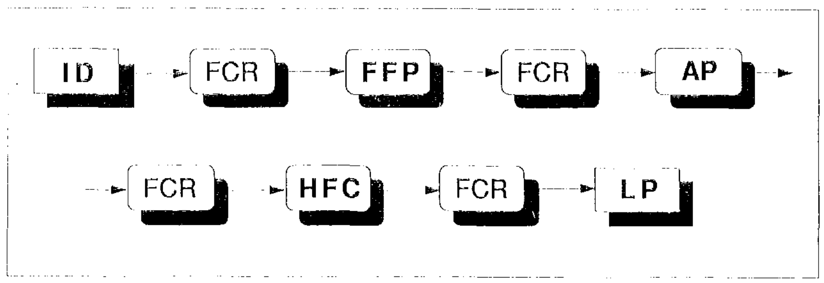

Fig. 2: Sequence of Application in a Constructional Version of GPSG

results may trigger FCRs that specify the *agr* feature. FCRs have to be applied at each step where a feature might have been specified in a local tree, namely after the FFP, the AP, and the HFC. LP statements can only be guaranteed to apply [5] properly on fully specified categories. Thus they operate in the last place (cf. figure 2) [6].

The next question to be addressed is how complex structures are built from local trees. Since in the constructional version nothing forces a daughter of one local tree and the mother of another one to have the same set of features specified with the same values, the two categories are not required to be identical, as in [GKPS], rather they must *unify* in order to be combined into a larger tree.

For each of the two categories involved in the unification, additional features may be specified. This *specification by construction*, when combined with the application of FCRs and FIPs, makes the results of transporting feature specifications within local trees immediately available to other local trees. The precise way of interaction with FCRs and FIPs depends on the strategy adopted for tree formation. In order to clarify the

The second strategy consists of top-down tree formation With this type of process, local trees are *expanded* by unifying their daughters with mothers of other local trees. The top-down strategy starts from a local tree (with mother S, fo instance), the categories of which have feature specification by virtue of an ID rule only. FCRs and FIPs cannot be applied during tree expansion because there is too little information available for deciding upon e.g. the value of *agr* (for the same reason, FCRs and FIPs are not applied to ID rules directly), rather they apply in a bottom-up manner as with the first strategy after the lexical insertion has been completed.

The latter strategy is utilized within the generation component [Busemann 1987] in the GPSG system, which has to introduce, for instance, number and case information into the structure that it is about to generate. This takes place in the course of tree expansion by adding relevant feature specifications to categories in the tree (to an NP mother, for instance). This information is usually not available in local trees at a deeper level, especially at local trees with lexical categories. Therefore the lexicon should contain word stems (rather than word forms) and, correspondingly, categories that are unspecified for e.g. number and case. [7]

This makes a situation possible that has not been discussed yet; namely, that when FIPs apply to local trees at these deeper levels they may have to cope with unspecified features. There

[5] For parsing, LP statements work as filters whereas for generation, they constructively order the branches in a local tree [Busemann 1987].

[6] Note that a similar ordering discovered by Shieber [Shieber 1986] results from investigations of underlying assumptions of [GKPS].

is indeed no requirement that AGR or HEAD features must have a value in order to unify. We should like the FIPs to work properly even if features have not yet received a value. In these cases, the feature values in question are *co-specified*, i.e. they will have the same value as soon as one of them is specified. In our example, number and case specifications are spread over the sub-structure dominated by the NP as soon as the FIPs apply to the local tree where they have been introduced. However, such a delayed specification makes it more difficult to maintain control over whether a category is still legal and whether a local tree still complies with the LP statements. For an elegant solution see [Weisweber 1988], in this volume.

In our present version of GPSG, we use neither metarules nor FSD. However, the linguist ought to still have the possibility of expressing elegantly language-particular generalizations with the aid of metarules. They will be realized in a preprocessing component in order to avoid having to apply them during parsing or during generation.

As for FSDs, we adopt the working hypothesis that they are superfluous if lexical entries are sufficiently specified and free feature instantiation (in the sense of [GKPS]) is not allowed. FSDs are needed in the GPSG version of [GKPS] because free feature instantiation may assign nonsensical values to features, which would never occur if the structure had been built orderly on the basis of sufficient lexical information. In the long run it might be desirable to use the device of FSDs in a constructional version of GPSG, too; namely, for those cases where features have not been specified, though the whole structure has been completed. However, we shall have to avoid the complexity of FSDs as defined in [GKPS]; a simplified solution might be analogous to our version of HFC for HFC, too, is a default device in the final account.

The constructional version of GPSG presented here constitutes the linguistic basis for parsing and generation of English and German sentences within the Berlin GPSG system. The system is fully implemented in Waterloo Core Prolog using the set of predicates defined by the KIT-CORE Prolog standard [Bittkau et al. 1987], which makes it possible to run it with several other Prolog dialects, too (e.g. Symbolics Prolog). At present, it runs on an IBM 4381 under VM/SP, on a Symbolics 3640 Lisp machine, and on an IBM AT.

4 Conclusion

It has been shown how our constructive version of GPSG avoids the problem of combinatorial explosion that would have arisen if we had tried to implement the GPSG formalism in its axiomatic version [GKPS] in a straightforward way. Our change in perspective also leads to an important simplification of the HFC because it is no longer necessary to build all the projections of an ID rule for the determination of the free feature specification sets.

The dilemma over the ordering of the CAP and the HFC has been removed too, which is crucial for any implementation of the formalism. But, for this to be achieved, we had to sacrifice part of the generality that characterizes the treatment of control in [GKPS]; i.e. although the question of which constituents have to agree with one another is not answered in a purely idiosyncratic way by the ID rules (because most of the cases can be accounted for by FCRs, which are, as it were, language-specific generalizations), the fact that agreement depends on functor-argument structures is no longer integrated into the formalism.

This loss, however, is compensated for by the fact that we can treat agreement in cases which the original CAP could not account for (as in the case where a functor is controlled by one of several arguments).

Although we have had to concentrate our presentation on just a few aspects of the constructive view of GPSG, we hope to have made plausible that our modified formalism is, in contrast to the original one, suitable for parsing and generation within an NL processing system.

References

Bittkau, Oliver; Haider, Christian; and Kietz, Jörg-Uwe (1987), *KIT-CORE-PROLOG Description (Version 1.0)*, KIT Internal Working Paper No. 17, Fachbereich Informatik, Technische Universität Berlin.

Busemann, Stephan (1987), Generierung mit GPSG, in K. Morik (ed.), *Procs. 11th German Workshop on Artificial Intelligence*, Berlin, Springer, 355-364.

Gazdar, Gerald; Klein, Ewan; Pullum, Geoffrey; and Sag, Ivan (1985), *Generalized Phrase Structure Grammar*, Oxford, Blackwell.

Hauenschild, Christa (1986), KIT/NASEV oder die Problematik des Transfers bei der maschinellen Übersetzung, in I. S. Bátori and H. J. Weber (eds.), *Neue Ansätze in maschineller Übersetzung. Wissensrepräsentation und Textbezug*, Tübingen, Niemeyer, 167-185.

Hauenschild, Christa, and Busemann, Stephan (1988), A Constructive Version of GPSG for Machine Translation, to appear in E. Steiner, P. Schmidt, and C. Zelinsky-Wibbelt (eds.), *From Syntax to Semantics - Insights From Machine Translation*, London, Frances Pinter, 1988.

King, Margaret, and Perschke, Sergej (1987), EUROTRA, in M. King (ed.), *Machine Translation Today: The State of the Art*, Edinburgh, Edinburgh University Press, 373-391.

Preuß, Susanne (1987), *GPSG-Syntax für ein Fragment des Deutschen*, KIT Internal Working Paper No. 20, Fachbereich Informatik, Technische Universität Berlin.

Ristad, Eric Sven (1986), Computational Complexity of Current GPSG Theory, in *Procs. 24th Annual Meeting of the ACL*, New York, 30-39.

Shieber, Stuart M. (1986), A Simple Reconstruction of GPSG, in *Procs. 11th COLING-86*, Bonn, 211-215.

Uszkoreit, Hans (1984), *Word Order and Constituent Structure in German*, Ph.D. Dissertation, University of Texas, Austin.

Weisweber, Wilhelm (1987), *Ein Dominanz-Chart-Parser für Generalisierte Phrasenstrukturgrammatiken*, KIT Report 45, Fachbereich Informatik, Technische Universität Berlin.

Weisweber, Wilhelm (1988), Using Constraints in a Constructive Version of GPSG, in *Procs. 12th COLING-88* (this volume), Budapest.

[7] A stem-form lexicon complemented with lemmatization and inflection procedures is better suited to NL processing anyway, at least if strongly inflecting languages such as German are involved.