Using Pattern-Action Rules for the Generation of GPSG Structures From Separate Semantic Representations

Stephan Busemann
DFKI GmbH
Stuhlsatzenhausweg 3
D-6600 Saarbrücken 11

phone: (+49 681) 302 5286
e-mail: busemann@dfki.uni-sb.de

Abstract

In many tactical NL generators the semantic input structure is taken for granted. In this paper, a new approach to multilingual, tactical generation is presented that keeps the syntax separate from the semantics. This allows for the system to be directly adapted to application-dependent representations. In the case at hand, the semantics is specifically designed for sentence-semantic transfer in a machine translation system.

The syntax formalism used is Generalized Phrase Structure Grammar (GPSG) The mapping from semantic onto syntactic structures is performed by a set of pattern-action rules. Each rule matches a piece of the input structure and guides the GPSG structure-building process by telling it which syntax rule(s) to apply. The scope of each pattern-action rule is strictly local, the actions are primitive, and rules can not call each other. These restrictions render the production rule approach both highly modular and transparent.

This report is an extended version of a paper appearing in the proceedings of 12th International Joint Conference on Artificial Intelligence 1991, Sydney. It also includes the material of a sister paper published in the proceedings of the conference of the 5th European Chapter of the ACL 1991, Berlin.

Contents

1 Introduction

In the field of unification-based computational linguistics, current research on tactical natural language (NL) generation concentrates on the following problem:

- Given a semantic representation (which is often called *logical form)* and a grammar that includes a lexicon, what are the surface strings corresponding to the semantic representation?

A variety of approaches to solving this problem in an efficient way has been put forward on the basis of unification-based grammar formalisms with a context-free backbone and complex categories. Most of this work shares a Montagovian view of semantics by assuming that the logical form be *integrated* into the grammar' rules, thus assigning to each syntactic category its semantic representation (e.g. [Dymetman/Isabelle 1988, Wedekind 1988, Russell *et al.* 1990, Shieber *et al.* 1990]). The semantic constructions are usually motivated by linguistic considerations alone; more precisely, by the interaction of syntactic and semantic constraints.

Within this *integrated-semantics approach* the generation task mainly consists of reconstructing a given logical form, thereby ensuring that the result is *complete* (all parts of the input structure are reconstructed) and *coherent* (no additional structure is built up). Thus, the surface strings then come out as a side effect. If such a generator were to be used as a front-end component of some application system, the semantics would have to be adapted to the application system's semantic representation language, which may depend on the system's purpose. To generate an utterance, a semantic representation would first have to be translated into an equivalent logical form, to which the grammar can eventually assign a syntactic structure containing the output string.

In order to avoid this adaptation, this paper suggests to *directly* relate a semantics that depends on a particular application to syntax. The semantics is not part of the grammar, but rather expressed within a separate semantic representation language. This approach, in which the grammar only covers the syntax part, is called the *separate semantics approach.* It has a long tradition in AI NL systems, but was rarely used for unification-based syntax and semantics. It will be argued that it can still be useful for interfacing a syntactic generator to some application system.

The application at hand is the Berlin machine translation (MT) system which is the first one to use an operational version of Generalized Phrase Structure Grammars (GPSG) [Gazdar *et al.* 1985] for both multilingual parsing and generation. The Berlin MT system translates sentences taken from a corpus of EC administrative texts from English to German and vice versa. It is based on a model of translation that includes several levels of transfer, the one closest to surface form of which has been implemented and tested.

The main goal of this paper is to describe a generator using a separate semantics specifically designed for transfer and to suggest a *structure-driven strategy* that is based on a system of pattern-action (PA) rules, as they are known from Al production systems (for an overview see [Davis/King 1977]). The purpose of these rules is to explicitly relate the semantic (sub)structures to possible GPSG syntactic

counterparts. The mapping process is driven by the semantic input structure that is traversed step by step. At each step PA rules are applied, which contribute to successively generating an overall syntactic structure from which the terminal string can easily be produced. This new approach allows for a carefully directed and nearly deterministic choice of grammar rules.

In Section 2, the separate semantics approach is introduced. Section 3 motivates and describes the underlying semantic representation language. Section 4 sketches the GPSG grammar formalism used and describes how it supports generation. The paper focusses, in Section 5, on the definition of PA rules and their use in the given framework of generation.

2 Keeping semantics separate from syntax

The integrated-semantics approach is often illustrated in a Prolog-like notation using DCG rules. The infix function symbol '/' is used in each category to separate the syntactic from the semantic part. Rule (1) introduces complements in an HPSG-style manner by "removing" the complement from the VP's subcategorization list (cf. [Pollard/Sag 1987]). The relation between the semantics S and the semantics of Compl is established in the lexical entry for the verb (2).

(1) vp(Subcat)/S --> vp([Compl|Subcat])/S, Compl.

(2) vp([np(_)/0bj, np(3rd-sing)/Subj])/kiss(Subj, Obj) —> [kisses].

Recent work on semantic-head-driven generation [Shieber *et al.* 1990, Calder *et al.* 1989, Noord 1990, Russell *et al.* 1990] provides a very promising step towards efficient, goal-directed reconstruction of logical form that is espescially suited for lexicon-centered grammar formalisms such as HPSG or UCG. It was observed that top-down generation may not terminate. This is illustrated in (1). If the vp node is used for top-down expansion, there is nothing to prevent the subcategorization list from growing infinitely. If the Comp node is used, the constituent to be generated must completely be guessed due to the uninstantiated semantics. Since the grammar will contain recursive rules (e.g. for relative clauses), the guessing procedure will not terminate either. In view of this problem a bottom-up approach was suggested that is guided by semantic information in a top-down fashion.

The benefits of integrated semantics are manifold. Elegant analyses of linguistic phenomena are possible that relate syntactic and semantic properties to each other (cf. the treatment of e.g. 'raising' and 'equi' constructions in [Pollard/Sag 1987]). Logical form is defined on purely linguistic grounds and as such, it is well-suited to the computational linguist's work.

However, if a generator based on an integrated semantics is to be used for conveying the results of some application system into NL, expressions of the application system's semantic representation language have to be adapted to logical form. Given that the grammar should not be rewritten, this amounts to an additional step of processing. This step may turn out to be costly since the semantic representation language will typically contain application-dependent information that must be

considered. Take, for instance, a transfer-based machine translation system (such as EUROTRA [Arnold/des Tombe 1986]). The results of the transfer (say, from German to English) are encoded in a semantic representation that is given to the system's generation component to produce the English target sentence. In a system capable of translating between a variety of languages, representations of this kind may themselves be subject to transfer and will therefore contain information relevant for translation.[1]

The effort of introducing an additional step of processing can be saved to a large extent by adopting a separate-semantics approach. The semantic representation language of some application system may directly serve as an *interface* to the generator.[2] In the case at hand, two additional components must be introduced into the generation scenario: the definition of the semantic representation language and PA rules. Instead of mapping expressions of the semantic representation language onto logical form, the semantic representation language is directly related to syntax by virtue of the PA rules.

3 Transfer results as input structures for the generator

The Berlin MT system is based on a general multi-level transfer framework of MT that has been mainly developed by Hauenschild [Hauenschild 1986, Hauenschild/Busemann 1988]. This framework assumes several succeeding levels of representation for both the source language as well as the target language text, among them a level of syntax, of sentence semantics, and of conceptual text representation. Between some of these levels transfer is assumed. Thus the complexity of the transfer step, which is viewed as the place where the divergencies between source and target language have to be bridged, is distributed between different components, and transfer will thus become more tractable than at a single level.

Within such a model, the input structures for the generator are motivated by MT considerations rather than by linguistic ones alone. The Berlin MT system, as developed and implemented so far, covers the sentence-semantic and the syntactic level with transfer being possible only at the former (cf. Figure 1). The sentence-semantic representation language family FAS (Functor-Argument Stuctures) [Hauenschild/Umbach 1988] has been designed to interface three different processes: GPSG-based analysis, sentence-semantic transfer of a source language FAS expression into a target language one, and GPSG-based generation.[3]

[1]An exception is the MiMo2 system [Noord *et al.* 1990]. The price to pay for allowing transfer at the level of logical form was to accept an "extremely poor" view of translation by just preserving the logical meaning and—as far as possible—the way in which meaning is built compositionally (quotation from [Noord *et al.* 1990]).

[2]This interface does *not* correspond to the common separation between making decisions about what to say and how to say it. Rather the interface in question must be situated somewhere in the, 'how to say it' component because it presupposes many decisions about sentence formulation (e.g. regarding pronominalization, or voice).

[3]Given that GPSG is chosen as the syntax formalism, one might wonder why the intensional logic

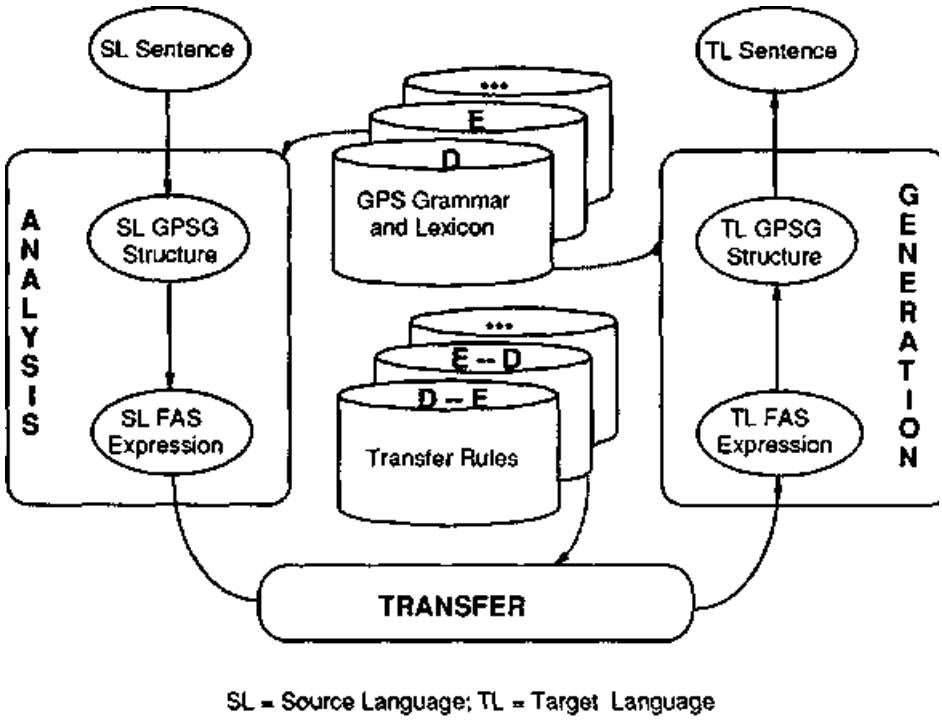

Figure 1: The Architecture of the Berlin MT System.

FAS is defined by context-free rule schemata with complex categories consisting of a main category that is associated with a fixed list of feature specifications (see Figure 2a for an example).[4] The categories are in canonical order with the functor preceding all of its arguments. FAS expressions contain almost no redundant information. For instance, number information is only located at the 'det' category. The use of semantic relations (encoded by the 'role' feature), role configurations ('conf') and semantic features ('sem') allows us to discriminate between different readings of words that result in different translational equivalents. For instance, German *verabschieden* translates to *say good-bye* if the 'affected' role[5] is a person (as in *He says good-bye to his friend),* but *to adopt* if the 'affected' role is a plan etc. (as in *The Council adopts the proposal).* This is encoded by the feature 'sem' at the category 'n.pred'.

For the kind of text envisaged, it was considered important to preserve the thematic structure of the source language sentence as far as possible during transfer. It is encoded at the level of the 'clause' daughters by virtue of the feature 'them' with the numerical values indicating which portion should preferrably be presented first, second, third etc. For instance, the English translation given for the German

representation proposed by GKPS was not adopted. On the one hand, there are intrinsic problems with the mapping scheme of GPSG structures onto intensional logic expressions [Umbach 1987]; on the other hand, MT-related information cannot be straightforwardly made explicit in intensional logic expressions [Hauenschild/Busemann 1988].

[4]In the present versions there are up to seven features in a FAS category. Many details irrelevant to the present discussion are omitted in the figure.

[5]The system of semantic roles is based on [Steiner *et al.* 1988b].

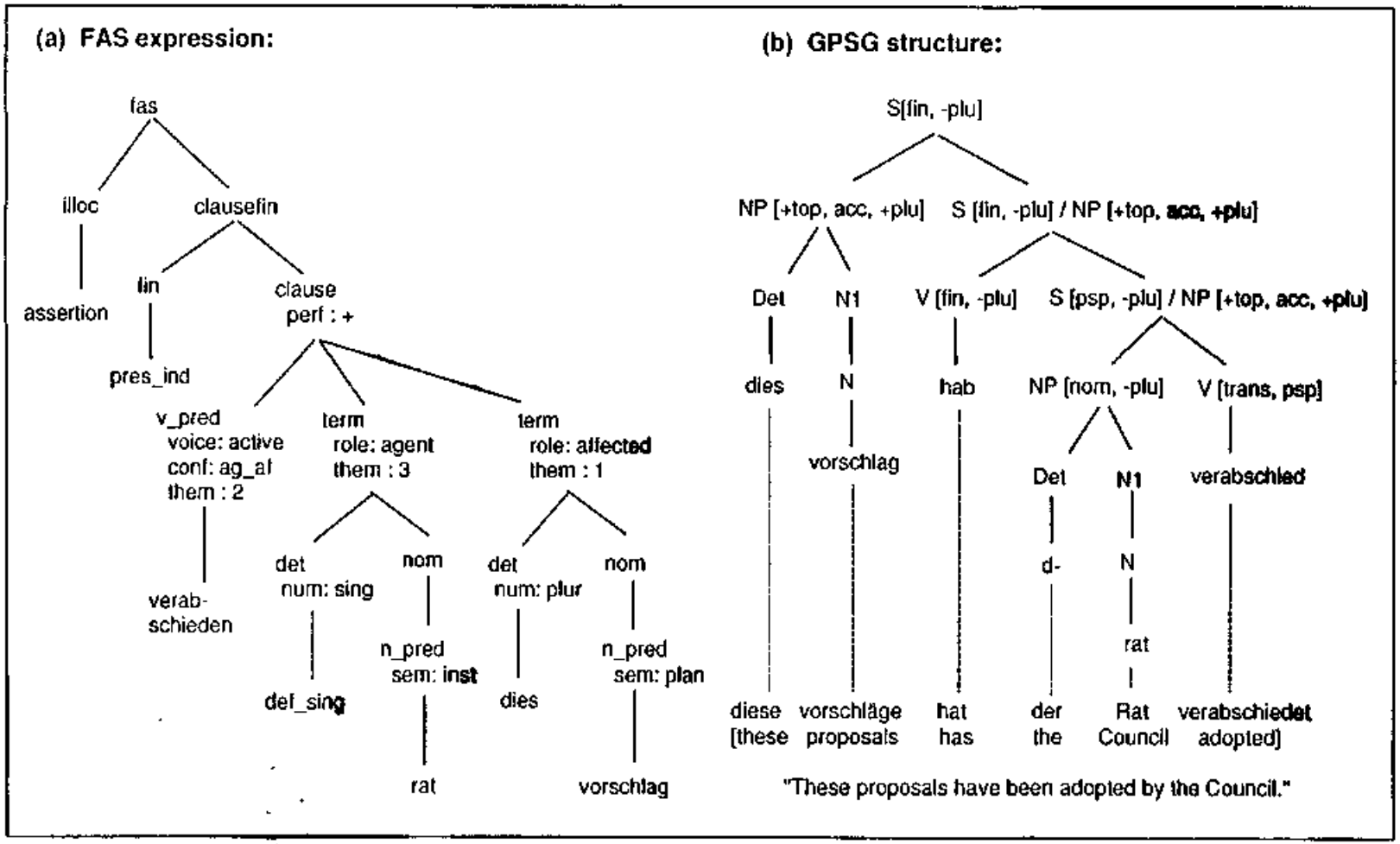

Figure 2: A Sample FAS Expression and a Corresponding GPSG Structure.

sentence in Figure 2b is passivized to reflect the source language order of the arguments.

From the point of view of generation, all decisions about style, voice, tense, or word choice are assumed to have been reached to during transfer. Thus a FAS expression reflects sufficient linguistic information for a sentence to be unambiguously assigned to it. For instance, it is possible to compute for every 'role' an NP's surface case with help of the features 'voice' and 'conf, and of the verb itself. With *verabschieden* [to adopt], active voice and the role configuration 'ag-af, which says that the verb has exactly two roles named 'agent' and 'affected' respectively, the 'agent' constituent is assigned nominative case whereas the 'affected' one yields accusative.

4 GPSG-based generation

The constructive GPSG formalism used is described in detail in [Busemann 1990]. The key ideas have been published in [Busemann/Hauenschild 1988, Hauenschild/Busemann 1988]. A major feature of the formalism is a strict application order of its components that allows the efficient implementation of different processing strategies for the construction of an admissible GPSG syntactic structure. This is different to the axiomatic formalism of [Gazdar *et al.* 1985], which assumes a simultaneous application of all components to exclude ill-formed structures.

For the present purpose, only three components will be sketched here. First of all, the concept of *complex categories* must be mentioned. Roughly, a complex GPSG category is a set of feature-value pairs with the values being allowed to be complex categories themselves.[6] Second, there is the separation between *immediate dominance* (ID) and *linear precedence* (LP). An ID rule D —> A,B,C says that in every local tree (i.e. a tree of depth one), categories A, B, and C are immediately dominated by category D. An LP statement B C says that in every local tree with categories B and C, B must precede C. Third, three *feature instantiation principles* require part of the features to be cospecified in some or all categories of a local tree.

The lexicon is a set of unary local trees consisting of a word stem dominated by a terminal GPSG category. Fully inflected word forms are provided by a separate inflection component that operates on stems and a set of morpho-syntactic features taken from the terminal categories of the GPSG structure.[7]

The construction of an admissible GPSG syntactic structure—see Figure 2b[8] for an example—consists of two subtasks that can be performed independently of each other, and each according to its own processing strategy:

Structure building: An ID rule (or a lexicon entry) licenses a local tree that contains the same amount of information. Local trees are combined with each

[6]Additional restrictions ensure that categories are finite, thus preserving context-freeness of GPSG.- Strings such as S, NP[nom], VP[inf], denote complex categories and are used for abbreviatory purposes only.

Using a root form lexicon is not just useful to keep the lexicon small, but even necessary for efficiency reasons (cf. the arguments in e.g. [Shieber *et al.* 1990]).

[8]In the context of GPSG, the symbol '/' represents the category-valued feature 'slash'.

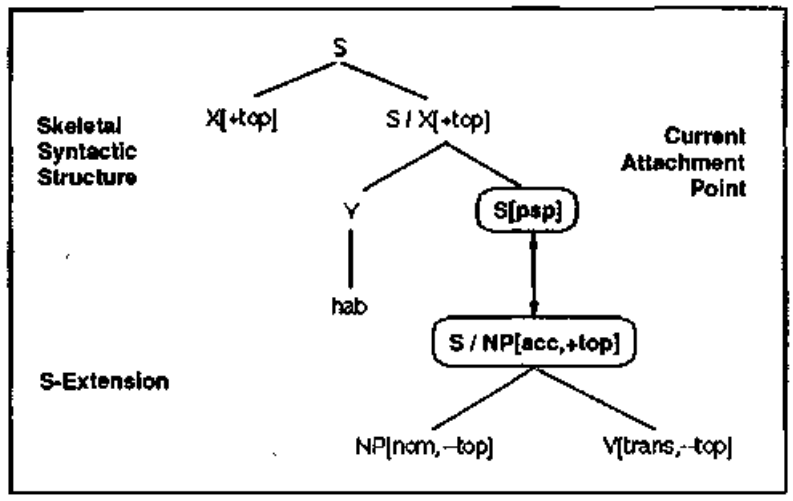

Figure 3: Introducing an S-Extension Into the Skeletal Syntactic Structure.

other to form a *skeletal syntactic structure.*

Feature instantiaton and ordering of the branches: To a (typically) strongly underspecified category, further information is successively added through the application of the feature instantiation principles and other components in a local tree. Finally, the LP statements can cause the branches to be reordered.

The question arises which strategies are best suited to efficient generation. For each subtask both a top-down and a bottom-up strategy have been investigated. As a result it turned out that structure building should occur top-down whereas feature instantiation should be performed in a bottom-up manner.[9] The structure-building strategy is justified in Section 5.1. Here we shall discuss why top-down feature instantiation may become indeterministic.

The feature instantiation principles apply to the mother and/or a subset of daughters in a local tree. In general, the more these categories are instantiated the less likely the feature instantiation principles will have to choose between alternative instantiations, which would be a source for backtracking. A top-down strategy would meet a more completely instantiated mother, but still underspecified daughters. With a bottom-up strategy, however, only the mother would be underspecified. For instance, consider the GPSG account of parasitic gaps, which are handled by the Foot Feature Principle. The 'slash' feature may occur at more than one daughter and then require all occurrences of it to unify with the mother (cf. [Gazdar *et al.* 1985, p. 162ff]). While this is easy to handle for a bottom-up process, a top-down strategy would have to guess at which daughters to instantiate a slash value.

Structure building consists of a stepwise expansion of a skeletal syntactic structure. There are non-terminal leaf categories in the skeletal syntactic structure that

[9]Modularity allows us to define a parser by adopting a bottom-up structure-building strategy fed by the input string instead of the FAS expression. In fact such a parser is part of the Berlin MT system [Weisweber 1987].

are called *attachment points*. These are the nodes that may be expanded by additional structure. Let us call such a structure *s-extension* (for structural extension). An s-extension is introduced into the skeletal syntactic structure by unifying its root category with an attachment point, which must then be removed from the current set. The skeletal syntactic structure now contains additional leaves, the categories—but not the word stems—of which become the set of current attachment points for following expansions. Let us call such an expansion step *structure-building action* (cf. Figure 3). Structure building starts with a skeletal syntactic structure consisting of a single attachment point labelled by an empty GPSG category.

Structure building alternates with feature instantiation in the following way: Top-down structure building ceases if some subtree contains no more nonterminal leaf; i.e. all of its leaves are word stems. Then bottom-up feature instantiation takes place at local trees licensed by ID rules (lexical trees are admissible by definition) until a nonterminal leaf category is encountered. The updated set of attachment points that was valid at that level becomes the current one again. The whole process terminates with a GPSG syntactic structure of some sentence as its result after the top-most local tree has passed feature instantiation.

Nothing has been said so far about how the next ID rule (or lexicon entry) is selected at a given stage of structure building. This is the topic of the following section.

5 Mapping FAS expressions onto GPSG structures

Structure building is triggered by traversing the input FAS expression and applying PA rules. Each PA rule is sensible to the particular piece of a FAS expression matched by the pattern. We shall start our discussion with the question of how much of a FAS expression should a pattern comprise. We shall then describe the PA rules and discuss their properties.

5.1 Traversing the FAS expression

In FAS, the information needed to apply some particular ID rule is not always accessible at a single FAS category or within some restricted local environment of it. Rather, information from distant portions of the FAS expression may be needed. For instance, in order to apply the ID rule for topicalization, S —> X[+top], S[fin] / X[+top][10], two distantly located specifications have to be collected (cf. Figure 2a): the FAS specification (them : 1), which is part of one of the daughter categories of 'clause', is interpreted as requiring topicalization of a syntactic constituent under the condition that a declarative sentence is being generated. This latter information is, however, available at the 'illoc' category of the FAS expression.

[10]Note that, in this ID rule, the X[+top] daughter is co-specified with the slash value of its sister, which eventually becomes more specific by virtue of the feature instantiation principles.

Two possible methods to collect the information present themselves. First, the pattern including (them : 1) could be required to cover as much of the FAS expression as would be needed to include 'illoc'. Unfortunately, the required size of the pattern is not always known in advance because the FAS syntax might allow an arbitrary number of recursively defined local trees to intervene.

The second method—which was eventually adopted—requires the patterns to cover not more than one local FAS tree. In order to gather information that is locally missing, an *intermediate storage* is used. If, for instance, the illocution is matched, information about whether or not a declarative sentence is being generated is stored. Later on, (them : 1} is encountered. If 'declarative' can be retrieved from the storage, the ID rule for topicalization can safely be triggered.

It is thus possible to guide the whole generation process by a single traversal of the FAS expression. The topicalization example above already suggests that the traversal should occur top-down rather than bottom-up: if it were bottom-up, the specification (them : 1) would have to be stored and the syntactic structuring at the sentence could only be determined when 'illoc' is matched. This delay would involve storing much additional information concerning e.g. auxiliary verbs that is not necessary otherwise.

The decision for a top-down traversal leads to the consequence that structure-building also occurs top-down: Because of a similar distribution of information in FAS expressions and in GPSG structures—for instance, lexical information is located at the terminal categories whereas much of the sentential information is found at the upper part of the structures—the strategy for traversing the FAS expression is the most efficient one for GPSG structure building.

In order to adequately restrict the power of the intermediate storage, it is defined as a two-dimensional array of order *[n,* 2] consisting of *n* pairs of the form *(key, entry).* Keys and entries are atomic symbols except for the entry to the key cat, which is a GPSG category. All keys but cat are defined by the PA rule writer. For instance, the information that a sentence is a 'declarative' is represented as (s-type:decl).

The storage *is* maintained by three kinds of *information-gathering actions* that write entries onto the storage or remove them from it:

put_store operates on a key and some information *I.* It writes *I* as the entry of key.

set_gpsg_features operates on a sequence of GPSG feature names and a sequence of GPSG feature values. It produces a GPSG category from them and unifies it with the entry of cat.

remove_store operates on a key. It returns the entry by removing it from the storage (if the key is cat, it leaves an empty GPSG category).

Note that no reading of information is possible without erasing it from the storage.

The GPSG category stored under cat serves to introduce the information collected into the syntactic structure. Translating FAS feature specifications such as

(them : 1} into GPSG feature specifications such as [top: +] is a task performed by the PA rules.

5.2 Pattern-Action rules

A PA rule is a production rule consisting of a left-hand side (the pattern) and a right-hand side (the actions). For a pattern to match local FAS trees, simple term unification suffices because FAS constituents as well as features are in canonical order. Patterns are implemented as two-element Prolog lists with the first element matching the root and the second one the list of daughters of a FAS local tree.

The actions of the right-hand side divide up into two kinds, namely a list of information-gathering actions for maintaining the intermediate storage and a list of structure-building actions for the generation of GPSG local trees. At most one of the lists may be empty. The actions are encoded as Prolog predicates.

Two sample PA rules are shown in (3) and (4). They encode the actions required for the example involving locally unaccessible information. In (3), the information-gathering action stores the fact that a declarative is being generated. The structure-building action call-id expands the skeletal syntactic structure by an s-extension according to the topicalization ID rule. The second PA rule matches a term specified by (them : 1) (which eventually will be realized e.g. as an NP). Here two information-gathering actions must be executed. The first one attempts to remove (s-type : decl) from the storage. If this succeeds, a GPSG category [+top] is generated and stored by the second information-gathering action.

How is the stored information introduced into the skeletal syntactic structure? Clearly this should be done by structure-building actions. However, rule (4) has no structure-building actions, i.e. the NP structure is built by virtue of another PA rule whose pattern matches the same local FAS tree. The definition of structure-building actions given in Section 4 is extended to include the unification of the category stored with some attachment point and the root of the s-extension (cf. Section 5.3 for a detailed example).

```
(3) pa_rule([fas() , [illoc(sem:ass) |_]] ,
        [put_store(s-type,decl)],
        [call_id("S --> X[+top], S / X[+top]")]).

(4) pa_rule([term(them:l),_] ,
        [remove_store(s-type,decl), set_gpsg_features( [top] ,[+])],
        [])-
```

Three kinds of PA rules should be distinguished according to the effects of their SBAs. Rule (4) does not generate structure at all. Rule (3) maps a FAS local tree onto a GPSG local tree by virtue of call_id. A third kind of rules to be presented later generates a more complex GPSG structure on the basis of FAS feature specifications.

Let us now turn to the control of the PA rules that must, intuitively speaking, guarantee that all relevant rules are applied in such a way that the intended effects are achieved. This we call *complete verbalization* of a local FAS tree.

A local FAS tree is completely verbalized iff a maximum number $n >= 1$ of applicable PA rules are successful. A PA rule is *applicable* to a local FAS tree *t* iff its pattern unifies with *t*. An applicable PA rule is *successful* iff all information-gathering actions can be executed in the given order without failure and at least one structure-building action—if present—is successful. A structure-building action is successful iff its s-extension as well as the stored category can be introduced into the skeletal syntactic structure.

The question of how the number of successful PA rules is guaranteed to be maximal is a matter of rule interaction and will be answered in Section 5.4.

What does it mean for an action to be *not* successful? Failure of information-gathering actions is straightforward. For instance, if the intermediate storage does not contain an appropriate entry for 's-type', the first information-gathering action of the PA rule (4) fails, and so does the rule itself. A structure-building action fails if either a stored category or the root of the extensor does not unify with an attachment point. If all structure-building actions fail, the PA rule does as well. If all PA rules applicable to a local FAS tree fail, chronological backtracking is invoked that leads to a rebuilding of the skeletal syntactic structure.

The control regime described above guarantees termination, completeness and coherence in the following way: The traversal of a FAS expression terminates since there is only a finite number of local trees to be investigated, and for each of them a finite number of PA rules is applicable. The skeletal syntactic structure generated is complete because all local FAS trees are processed and for each a maximum number of PA rules is successful. It is coherent because (1) no PA rule may be applied whose pattern is not matched by the FAS expression and (2) all attachment points must be expanded.

The algorithm described so far is summarized in Figure 4. A more detailed discussion can be found in [Busemann 1990].

5.3 An example

This section demonstrates some of the essential points of the mapping from the FAS expression in Figure 2a onto the GPSG structure in Figure 2b, which involves the topicalization of the direct object.

The first step is taken by virtue of the PA rule (3) above. The daughters of the topicalization ID rule are the current attachment points. The investigation of the 'illoc' and the 'clausefin' local FAS trees yields information about verb inflection (tense, mood), which is stored and used later by the morphological inflection component.

At the 'clause' level, two PA rules are applicable. (5) matches a (perf : +} specification at the 'clause' category and introduces a perfect auxiliary by its structure-building action. Note that by call_id_lex, a different kind of structure-building action is used here that also provides, in addition to the task of call_id, for the auxiliary's expansion into the lexicon. This is necessary since FAS does not represent perfect auxiliaries but by a feature, whereas on the GPSG side, a terminal local tree must be generated.

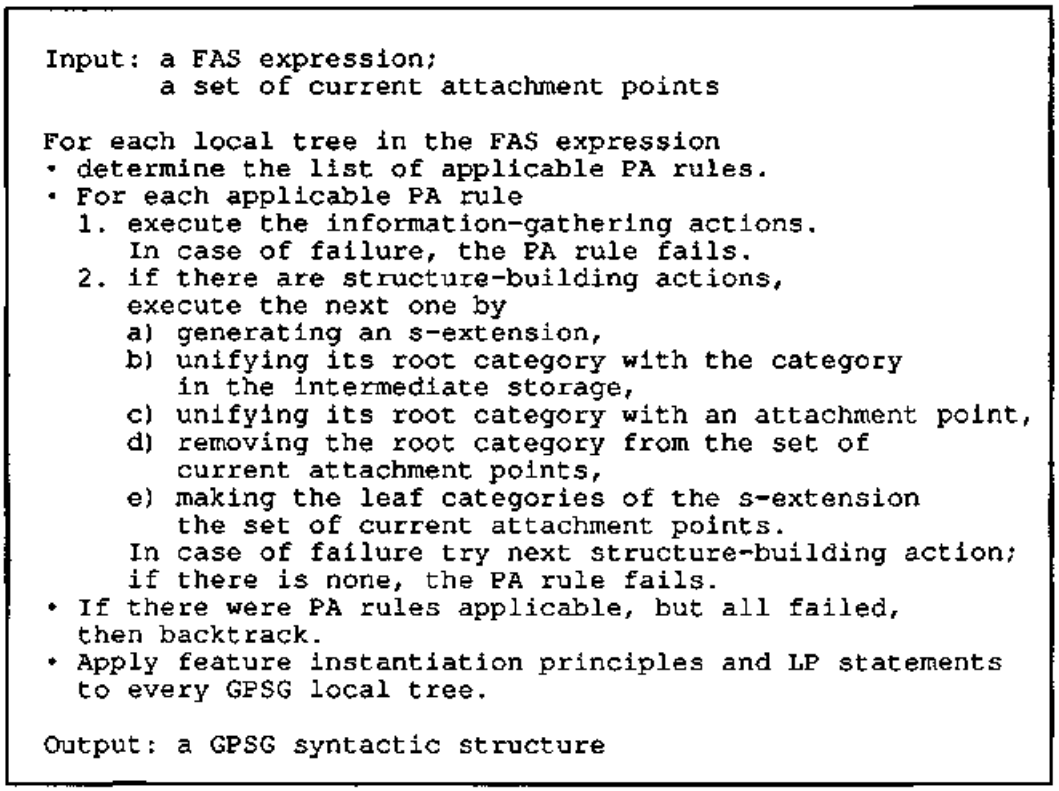

Figure 4: The Generation Algorithm.

The root of the s-expansion generated by call_id_lex unifies with S/X[+top], which is then removed from the set of attachment points. The new set has as its only element S[psp]. The skeletal syntactic structure built so far is illustrated in Figure 3.

(5) pa_rule([clause(perf:+),_],
[],
[call_id_lex("S —> V[+aux], S[psp]")]).

The second rule applicable at the 'clause' level is (6). It offers several possibilities to introduce an s-extension, given that the first daughter of the FAS tree is a 'v_pred' with active voice and role configuration 'ag-af. Since this meets the case at hand, the first structure-building action is successfully executed though this will eventually turn out to be wrong. The current set of attachment points consists of the daughters of the ID rule.

(6) pa_rule([_, [v_pred(conf:ag-af, voice:active)|_]],
[],
[call_id("S —> NP[nom,-top], NP[acc,-top], V[trans,-top]"),
call_id("S/NP[nom,+top] --> NP[acc,-top], V[trans,-top]"),
call_id("S/NP[acc,+top] --> NP[nom,-top], V[trans,-top]")]).

Note that applying the two rules the other way round would have prevented the auxiliary from being introduced into the skeletal syntactic structure due to lack of a suitable attachment point. In that case, the number of successful PA rules would

not have been maximal.

'In a next step, the verb is generated from 'v_pred' using PA rule (7). The assignment of surface case to roles is stored, and a GPSG lexicon entry is called. Since Vftrans] unifies with the attachment point V[trans, -top], the local tree is inserted into the skeletal syntactic structure. As there are no new attachment points and the local tree is admissible, the current attachment points are NP[nom, -top] and NP[acc, -top].

```
(7) pa_rule([v_pred(conf:ag-af,voice:active), [verabschieden]],
        [put_store(agent,nom), put_store(affected,acc)],
        [call_lex("V[trans] —> verabschied")]).
```

The next local FAS tree to be verbalized is rooted by a 'term' with (role : agent). Note that it is specified by (them : 3} which causes rule (8) to store a GPSG category [-top], saying that the NP must not be topicalized.

Another PA rule is applicable that is similar to rule (9) but handles singular number. Its first information-gathering action removes (agent : nom) from the storage. The second one stores a GPSG category containing the case information just retrieved as well as number information taken from the pattern. The s-extension is successfully introduced into the skeletal syntactic structure since NP[-top, -phi, nom] unifies with the attachment point NP[-top, nom]. Note again, that an application of the two rules in different order would cause the [-top] specification to be introduced into the skeletal syntactic structure by a structure-building action that verbalizes a different part of the FAS expression.

```
(8) pa_rule([term(them:3),_],
        [set_gpsg_features([top],[-])],
        []).

(9) pa_rule([terra(role:Role), [det(def:+,num:plur)|_]],
        [remove_store(Role, Case),
         set_gpsg_features([plu,cas], [+,Case])],
        [call_id("NP —> Det, N1")])-
```

Let us skip the straightforward verbalization of the term's descendants and turn to the second 'term' with (role : affected). The only remaining attachment point is NP[acc, -top]. Applying the PA rule (4) here causes a GPSG category [+top] to be stored. Furthermore, PA rule (9) adds accusative case and plural number to it and attempts to introduce another NP s-extension[11] into the skeletal syntactic structure. This, however, fails because of the incompatible 'top' specifications.

In this case, backtracking leads to a new choice of the S expansion in PA rule (6) by using the second structure-building action. With the new s-extension introduced into the skeletal syntactic structure, however, the NP[nom] cannot be introduced anymore, again because of incompatible 'top' specifications (values of the GPSG slash feature also count as attachment points). Thus a second revision of the S expan-

[11]Note that the plural determiner is generated because it is definite ((def : +) at the 'det' category in the pattern). An indefinite plural determiner is not expressed in German (and English)-hence a PA rule with (def : —) would base an s-extension on the ID rule NP —> N1.

sion becomes necessary, and the third structure-building action in rule (6) is used (cf. the s-extension in Figure 3). This time, both the verb and the NP[nom] previously generated can be attached, and the remaining attachment point NP[acc, +top] unifies with NP[+top, +plu, ace]. After the generation of the NP, which we also skip, all current attachment points are expanded.

This is the moment for the feature instantiation principles to operate on the local tree under consideration (i.e. the lowest one with mother S in Figure 2b). At the next higher level in the skeletal syntactic structure, the same situation arises: no more current attachment points. The feature instantiation principles cause, among other things, the S categories to share their slash values. As a consequence, the only remaining attachment point at the top level of the skeletal syntactic structure, X[+top], is further instantiated by the NP[acc] structure and erased from the set (remember that it is cospecified with the slash value of its sister). Thus generation terminates successfully.

Finally the terminal local trees of the admissible GPSG structure are fed to the morphological inflection component in order to eventually produce the output string.

5.4 On the interaction of PA rules

There are some important properties of PA rules known from production systems that must hold for the modular encoding of the mapping to pay off [Davis/King 1977]. Though the generation system presented uses productions, it is *not* a production system: There is no common database to be modified by the productions and consequently, known conflict resolution strategies such as the RETE algorithm [Forgy 1979] do not apply.

Conflicts arise in the present system only if more than one rule matches a given local FAS tree. As the matching is free of side-effects and the actions are primitive (i.e. no calls to other actions are allowed), the PA rules can communicate with each other only indirectly, i.e. by modifying the content of the intermediate storage or by successfully applying a structure-building action, thereby creating a situation in which another PA rule becomes applicable (or cannot be applied anymore).

As should be evident from the example, conflicting rules must be applied in a certain order to guarantee that a maximal number of them will be successful. This requirement is formalized as follows: Due to the restricted power of the PA rules, possible conflicts are detected and resolved a *priori.* All PA rules matching the same local FAS tree are identified with help of the FAS rule schemata. These PA rules are members of the local FAS tree's conflict set. The elements of every such conflict set are partially ordered according to precedence rules that determine for each pair of PA rules whether or not the first one must be applied before the second one.

For instance, the conflict that arose with the NP s-extension is resolved by requiring that PA rules without a structure-building action are applied first. The conflict regarding the perfect auxiliary is resolved with help of a precedence rule that checks the ID rules that would be invoked by the respective structure-building actions. If the mother of the second ID rule can be unified with a daughter of the first one,

but not vice versa, then the first PA rule must be applied before the second one. Thus a PA rule with a structure-building action invoking the ID rule S —> V, S[psp] will apply before another one whose structure-building action involves the ID rule S/NP[acc] —> V,NP[nom].

However, GPS grammars can be linguistically justified where this method alone does not suffice. For instance, Uszkoreit [Uszkoreit 1984] suggested that the perfect and the passive auxiliaries both be introduced by rules of type S —> V[+aux],S. Which one should licence a local tree higher up in the skeletal syntactic structure? The answer is hidden in the lexicon: the passive auxiliary can have a past participle *(Er ist gegessen worden* [it has been eaten]), but the perfect auxiliary may not be passivized (* *Er wurde gegessen gewesen* [* it was had eaten]). Hence, the perfect auxiliary should come first in a top-down strategy. This can, however only be verified during word inflection.

The most obvious way out is to force the two PA rules to belong to different conflict sets: The perfect auxilary can be introduced while processing the 'clausefin' local tree whereas the passive auxiliary is generated during the verbalization of the 'clause' tree (cf. Figure 2a). Thus the correct ordering is forced by the FAS traversion strategy.

Though this is far from being a general solution to problems of this kind, it sheds some light on the task of writing PA rules: though PA rules are highly modular and declaratively represented, their intended interaction must explicitly or implicitly be considered, too.

6 Conclusion

A new approach to multilingual, tactical generation has been presented that allows for the direct mapping of a separate, application-dependent semantic representation—the result of sentence-semantic transfer during MT—onto a GPSG syntactic structure. To build the syntactic structure, a set of pattern-action rules is used that forms a modular component of the generation system. Since it is part of the language-specific knowledge, it can be exchanged together with the grammar and the semantic representation in order to generate strings of a different language.

The strategy for constructing the skeletal syntactic structure is top-down. The problems observed with top-down generators using an integrated semantics cannot occur in the separate-semantics approach. Expansion of grammar rules are controlled by the semantic representation if each rule application is explicitly triggered by a structure-building action. Situations causing an infinite expansion due to an uninstantiated semantics (as with top-down expansion using the rule (2)) cannot arise at all since the separate semantics is fully specified.

The PA rules allow a grammar writer to express all possible syntactic realizations of a local semantic substructure. It remains open to further research how easily linguistic generalizations can be expressed by PA rules. Another research goal is to formalize conditions for a bidirectional use of PA rules, which clearly involves major modifications of the concepts presented here. The present approach opens up a new

way for a linguistically justified grammar formalism to be incorporated in different generation systems.

The generator is implemented in Waterloo Core Prolog on an IBM 4381 under VM/SP; a transported version runs as part of the Berlin MT system in Arity Prolog on an AT. The fragments of German and English covered are medium-sized (50 to 70 ID and PA rules). For the ordering of PA rules, four precedence rules sufficed. Run time for the generation of the sentence in Figure 2 is about 4.7 sec. on the AT.

Acknowledgements

This paper was written within the project DISCO, which is funded by the German Ministry for Research and Technology (BMFT) under contract ITW 9002. Most of the underlying research was carried out while the author was member of the EUROTRA-D accompanying research project KIT-FAST at the Technical University of Berlin. That work was also supported by BMFT (contract No. 1013211).

I wish to thank my colleagues Christa Hauenschild, Susanne Preuß, Carla Umbach, and Wilhelm Weisweber from TU Berlin, without whom this work would not have been possible. I am grateful to Harald Trost for reading and commenting on an earlier version of this paper.

References

[Arnold/des Tombe 1986] Doug Arnold and Louis des Tombe (1986), 'Basic Theory and Methodology in Eurotra', in S. Nirenburg (ed.), *Theoretical and Methodological Issues in Machine Translation*, Cambridge: Cambridge University Press, 114-135.

[Busemann 1990] Stephan Busemann (1990), 'Generierung mit Generalisierten Phrasenstruktur-Grammatiken', Doctoral Dissertation, Universität des Saarlandes, Computer Science Department. Also available: Technical University of Berlin, Computer Science Department, KIT-Report No. 87.

[Busemann/Hauenschild 1988] Stephan Busemann und Christa Hauenschild (1988), 'A Constructive View of GPSG or How to Make it Work', in *Proc. 12th COLING-88*, Budapest, 77-82.

[Calder *et al.* 1989] Jonathan Calder, Mike Reape, and Henk Zeevat (1989), 'An Algorithm for Generation in Unification Categorial Grammar', in *Proc. 4th Conf. of the European Chapter of the ACL*, Manchester, 233-240.

[Davis/King 1977] Randall Davis und Jonathan King (1977), 'An Overview of Production Systems', in E. W. Elcock and D. Michie (eds.), *Machine Intelligence 8*, Chichester: Ellis Horwood, 1977, 300-332.

[Dymetman/Isabelle 1988] Marc Dymetman and Pierre Isabelle (1988), 'Reversible Logic Grammars for Machine Translation', in *Proc. 2nd Int. Conf. on Theoretical and Methodological Issues in Machine Translation of Natural Languages,* Pittsburgh, PA.

[Forgy 1979] C. Forgy (1979), 'On the Efficient Implementation of Production Systems', Ph.D. Dissertation, Carnegie Mellon University, Pittsburgh, PA.

[Gazdar *et al.* 1985] Gerald Gazdar, Ewan Klein, Geoffrey Pullum und Ivan Sag, (1985), *Generalized Phrase Structure Grammar,* Oxford: Blackwell.

[Hauenschild 1986] Christa Hauenschild (1986), 'KIT/NASEV oder die Problematik des Transfers bei der maschinellen Übersetzung', in I. Batori und H.-J. Weber (eds.), *Neue Ansätze in maschineller Sprachübersetzung: Wissensreprdsentation und Textbezug,* Tübingen: Niemeyer, 167-196.

[Hauenschild/Busemann 1988] Christa Hauenschild and Stephan Busemann (1988), 'A Constructive Version of GPSG for Machine Translation', in [Steiner *et al.* 1988a], 216-238.

[Hauenschild/Umbach 1988] Christa Hauenschild and Carla Umbach (1988), 'Funktor-Argument-Struktur. Die satzsemantische Representations- und Transferebene im Projekt KIT-FAST', in J. Schütz (ed.), *Workshop 'Semantik und Transfer',* IAI Working Papers No. 6, Saarbrücken, 16-35.

[Mahr/Umbach 1989] Bernd Mahr and Carla Umbach (1989), 'Functor-Argument Structures for the Meaning of Natural Language Sentences and Their Formal Interpretation', in S. Busemann, C. Hauenschild, and C. Umbach (eds.), *Views of the Syntax/Semantics Interface. Proc. Workshop 'GPSG and Semantics', TU Berlin, Feb. 22-24, 1989,* Technical University of Berlin, Computer Science Department, KIT Report No. 74. 113-131.

[Noord 1990] Gertjan van Noord (1990), 'An Overview of Head-Driven Bottom-up Generation', in R. Dale, C. Mellish, and M. Zock (eds.), *Current Research in Natural Language Generation,* Academic, 141-165.

[Noord *et al.* 1990] Gertjan van Noord, Joke Dorrepaal, Pirn van der Eijk, Maria Florenza, and Louis des Tombe (1990), 'The MiMo2 Research System', in *Proc. 3rd Int. Conf. on Theoretical and Methodological Issues in Machine Translation,* Austin, Texas.

[Pollard/Sag 1987] Carl J. Pollard and Ivan A. Sag (1987), *Information-Based Syntax and Semantics. Volume I,* Center for the Study of Language and Information, CSLI Lecture Notes 13, Chicago: University of Chicago Press.

[Russell *et al.* 1990] Graham Russell, Susan Warwick, and John Carroll (1990), 'Asymmetry in Parsing and Generating with Unification Grammars: Case Studies from ELU', in *Proc. Conf. of the 28th Annual Meeting of the ACL,* Pittsburgh, 205-211.

[Shieber *et al.* 1990] Stuart M. Shieber, Gertjan van Noord, Robert C. Moore, and Fernando C. N. Pereira (1990), 'A Semantic-Head-Driven Generation Algorithm for Unification-Based Formalisms', in *Computational Linguistics,* 16, No. 1, 30-42.

[Steiner *et al.* 1988a] Erich Steiner, Paul Schmidt, and Cornelia Zelinsky-Wibbelt (eds.), *From Syntax to Semantics—Insights From Machine Translation,* London: Frances Pinter, 1988.

[Steiner *et al.* 1988b] Erich Steiner, Ursula Eckert, Birgit Roth, and Jutta Winter-Thielen (1988), 'The Development of the EUROTRA-D System of Semantic Relations', in [Steiner *et al.* 1988a], 40-104.

[Umbach 1987] Carla Umbach (1987), *Zur semantischen Interpretation in der Theorie der GPSG,* Technical University of Berlin, Computer Science Department, KIT Working Paper No. 19.

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

[Wedekind 1988] Jürgen Wedekind (1988), 'Generation as Structure Driven Derivation', in *Proc. 12th COLING-88,* Budapest, 732-737.

[Weisweber 1987] Wilhelm Weisweber (1987), *Ein Dominanz-Chart-Parser für generalisierte Phrasenstrukturgrammatiken,* Technical University of Berlin, Computer Science Department, KIT Report No. 45.