Using Pattern-Action Rules for the Generation of GPSG Structures From MT-Oriented Semantics

Stephan Busemann
Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI) GmbH*
Stuhlsatzenhausweg 3, D-6600 Saarbrücken 11, Germany
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 (PA) 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 PA rule is strictly local, the actions are primitive, and rules can not call each other. These restrictions render the production system approach both highly modular and transparent.

1 Introduction

In most tactical unification-based approaches, the meaning representation a generator starts from is taken as a given. A Montague style semantics is often used where each lexicon entry and each syntax rule is assigned a semantics in the grammar (e.g. [Dymetman and Isabelle, 1988, Shieber *et al.,* 1990]). The semantic constructions are usually motivated by linguistic considerations alone; more precisely, by the interaction of syntactic and semantic constraints.

Such a system is capable of computing a terminal string for a given logical form. If it were to be used as a front-end component of some application system,

*This work was partially funded by the German Minister for Research and Technology (BMFT) under contract ITW 9002. Most of the research was accomplished at the Technical University of Berlin in the project KIT-FAST, which was funded by the BMFT under contract 1013211. I would like to thank Christa Hauenschild for numerous valuable discussions.

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. A new approach to the syntax-semantics interface is presented that uses a set *of pattern-action* (PA) rules similar to those known from production systems. The grammar only covers syntax; the semantics is completely left to the respective 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 generator takes as input a semantic representation specifically designed for transfer. PA rules are used for extracting from it the information relevant to generation and stepwise constructing a GPSG syntactic structure. In this generator, a modern syntax formalism is for the first time coupled with Al production system techniques.

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

2 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 and Busemann, 1988]. This framework assumes several succeeding levels of representation for both the source language as well as the target language text, among them

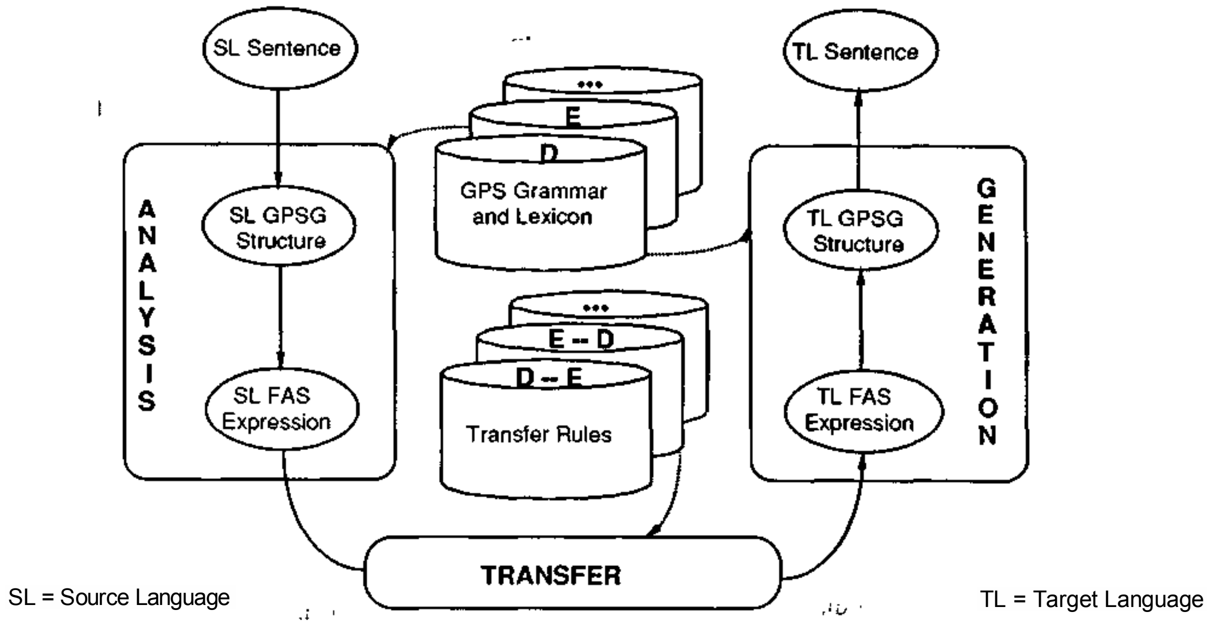

Figure 1: The Architecture of the Berlin MT System.

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 and 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.[1]

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).[2] 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.[3] For instance, German *verab-schieden* translates to *say good-bye* if the 'affected' role is a person (as in *He says good-bye to his friend),* but *to adopt* if the 'affected' role is a plan (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 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.

[1] Given that GPSG is chosen as the syntax formalism, one might wonder why the intensional logic (IL) proposed by GKPS was not adopted. On the one hand, there are intrinsic problems with the mapping scheme of GPSG structures onto IL expressions [Umbach, 1987]; on the other hand, MT-related information cannot be straightforwardly made explicit in IL expressions [Hauenschild and Busemann, 1988].

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

[3] The system of semantic roles is based on [Steiner *et a*/., 1988].

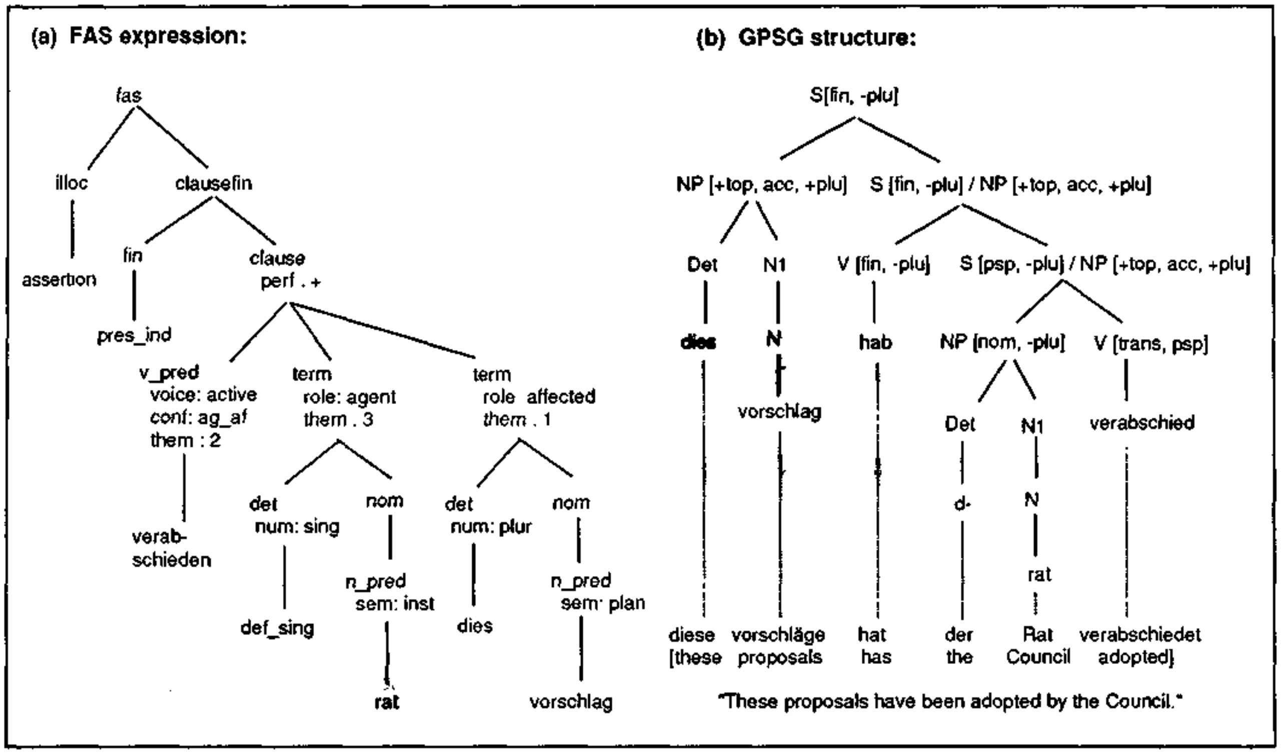

Figure 2: A Sample FAS Expression and a Corresponding GPSG Structure.

3 GPSG-based generation

The constructive GPSG formalism used is described in detail in [Busemann, 1990, Hauenschild and 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.[4] 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* (FIPs) 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.[5]

The construction of an admissible GPSG syntactic structure (cf. e.g. Figure 2b) 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 other to form a *skeletal syntactic structure* (SSS).

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

Structure is built in a top-down fashion during generation (cf. Section 4.1) whereas feature instantiation is more efficiently performed bottom-up.[6]

[4] 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.

[5]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]).

[6]Top-down feature instantiation may become indeterministic due to the definition of GPSG's FOOT feature principle, which can require several daughters to cospecify with respect to certain features [Hauenschild and Busemann, 1988].

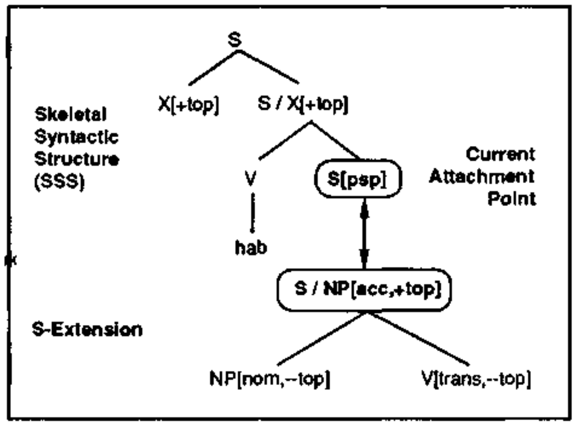

Figure 3: Introducing an S-Extension Into the SSS.

Structure building consists of a stepwise expansion of an SSS. There are non-terminal leaf categories in the SSS that 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 SSS by unifying its root category with an attachment point, which must then be removed from the current set. The SSS 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* (SBA) (cf. Figure 3). Structure building starts with an SSS 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 triggered at a given stage of structure building. This is the topic of the following sec tion.

4 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.

4.1 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][7], 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.

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).

[7]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 FIPs.

The storage is maintained by three kinds of *information-gathering actions* (IGAs) 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.

4.2 Pattern-Action rules

PA rules consist 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 local FAS tree.

The actions of the right-hand side divide up into two kinds, namely a list of IGAs for maintaining the intermediate storage and a list of SBAs 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 (1) and (2). They encode the actions required for the example involving locally unaccessible information. In (1), the IGA stores the fact that a declarative is being generated. The SBA call_id expands the SSS 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 as e.g. an NP). Here two IGAs 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 IGA.

How is the stored information introduced into the SSS? Clearly this should be done by SBAs. However, rule (2) has no SBAs, i.e. the NP structure is built by virtue of another PA rule whose pattern matches the same local FAS tree. The definition of SBAs given in Section 3 is extended to include the unification of the category stored with some attachment point and the root of the sextension (cf. Section 4.3 for a detailed example).

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

(2) pa_rule( [term(them:1),_] ,
        [ remove_store(s-type,decl),
          set_gpsg_features([top] , [+] )] ,   [] ) .
```

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.

```
Input:  a FAS expression;
        a set of current attachment points

For each local tree in the FAS expression
• determine the list of applicable PA rules.
• For each applicable PA rule
  1. execute the IGAs.
     In case of failure, the PA rule fails.
  2. if there are SBAs, execute the next SBA
     by
     a) generating an s-extension,
     b) unifying its root category with
        the category in the intermediate
        storage,
     c) unifying its root category with an
        attachment point,
     d) removing the root category from the
        set of current attachment points,
     e) making the leaf categories of the
        s-extension the set of current
        attachment points.
     In case of failure try next SBA;
     if there is none, the PA rule fails.
• If there were PA rules applicable, but all
  failed, then backtrack.
• Apply FIPs and LP statements to every GPSG
  local tree.

Output:  a GPSG syntactic structure
```

Figure 4: The Generation Algorithm.

A local FAS tree is completely verbalized iff a maximum number n $\geq$ 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 elements of IGA can be executed without failure and at least one SBA — if present — is successful. An SBA is successful iff its s-extension as well as the stored category can be introduced into the SSS.

The question of how the number of successful PA rules is guaranteed to be maximal is a matter of control and will be answered in Section 4.4.

What does it mean for an action to be *not* successful? Failure of IGAs is straightforward. For instance, if the intermediate storage does not contain an appropriate entry for 's-type', the first IGA of the PA rule (2) fails, and so does the rule itself. An SBA fails if either a stored category or the root of the extensor does not unify with an attachment point. If all SBAs 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 SSS.

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

4.3 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 applying the PA rule (1) above. The daughters of the topicalization ID rule are the current attachment points. At the 'clause' level, two PA rules are applicable. (3) matches a *(perf : +)* specification at the 'clause' category and introduces a perfect auxiliary by its SBA. Note that by call_id_lex, a different kind of SBA is used here that 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.

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 only current attachment point is S[psp].The SSS built so far is illustrated in Figure 3.

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

The second rule applicable at the 'clause' level is (4). 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 SBA 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.

```
(4)  pa_rule([_,  [v_pred(conf: ag-af,
                            voice: active) I _],  D,
       [call_id("S —> NP[nom,-top],
                    NP[acc,-top],  V[trans,-top]"),
        call_id("S/NP[nom,+top]  —>
                    NP[ace,-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 SSS 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 (5). The assignment of surface case to roles is stored, and a GPSG lexicon entry is called. After the insertion into the SSS, the current attachment points are NP[nom, -top] and NP[acc, -top].

```
(5)  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 (6) to store a GPSG category [-top], saying that the NP must not be topicalized.

Another PA rule is applicable that is similar to rule (7) but handles singular number. Its first IGA 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 SSS using 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 SSS by an SBA that verbalizes a different part of the FAS expression.

```
(6)  pa_rule([term(them:3),_] ,
        [set,gpsg_features ( [top] ,[-])],  [] ) .
```

```
(7)  pa_rule([term(role:Role),
              [det(del : + ,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 (2) here causes a GPSG category [+top] to be stored. Furthermore, PA rule (7) adds accusative case and plural number to it and attempts to introduce another NP s-extension into the SSS. This, however, fails because of the incompatible 'top' specifications.

Backtracking leads to a new choice of the S expansion in PA rule (4) by using the second SBA. With the new s-extension introduced into the SSS, 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 expansion becomes necessary, and the third SBA in rule (4) 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, acc]. After the generation of the NP, which we also skip, all current attachment points are expanded.

This is the moment for the FIPs 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 SSS, the same situation arises: no more current attachment points. The FIPs 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 SSS, 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.

4.4 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 and 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 an SBA, 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 an SBA 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 SBAs. 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 an SBA invoking the ID rule S —> V,S[psp] will apply before another one whose SBA involes the ID rule S/NP[acc] —> V, NP[nom].

5 Conclusion

A new approach to multilingual, tactical generation has been presented that allows for the direct mapping of an 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 separate 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 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.

References

[Busemann, 1990] Stephan Busemann. *Generierung natürlicher Sprache mit Generalisierten Phrasenstrukturgrammatiken.* PhD thesis, Univ. des Saarlandes, Comp. Sc. Dept., Saarbrücken, 1990. Also: Techn. Univ. Berlin, Comp. Sc. Dept., KIT report 87.

[Davis and King, 1977] Randall Davis and Jonathan King. An overview of production systems. In E. W. Elcock and D. Michie, editors, *Machine Intelligence 8,* pages 300-332. Ellis Horwood, Chichester, 1977.

[Dymetman and Isabelle, 1988] Marc Dymetman and Pierre Isabelle. Reversible logic grammars for machine translation. In *Proc. 2nd Int. Conf. on Theoretical and Methodological Issues in Machine Translation of Natural Languages,* Pittsburgh, PA, 1988.

[Forgy, 1979] C. Forgy. *On the Efficient Implementation of Production Systems.* PhD thesis, Carnegie Mellon Univ., Pittsburgh, PA., 1979.

[Gazdar *et al,* 1985] Gerald Gazdar, Ewan Klein, Geoffrey Pullum, and Ivan Sag. *Generalized Phrase Structure Grammar.* Basil Blackwell, London, 1985.

[Hauenschild and Busemann, 1988] Christa Hauenschild and Stephan Busemann. A constructive version of GPSG for machine translation. In E. Steiner, P. Schmidt, and C. Zelinsky-Wibbelt, editors, *From Syntax to Semantics—Insights From Machine Translation,* pages 216-238. Frances Pinter, London, 1988.

[Hauenschild and Umbach, 1988] Christa Hauenschild and Carla Umbach. Funktor-Argument-Struktur. Die satzsemantische Repräsentations- und Transferebene im Projekt KIT-FAST. In J. Schütz, editor, *Workshop "Semantik und Transfer",* IAI working papers no. 6, pages 16-35, Saarbrücken, 1988.

[Hauenschild, 1986] Christa Hauenschild. KIT/NASEV oder die Problematik des Transfers bei der maschinellen Übersetzung. In I. Bátori and H.-J. Weber, editors, *Neue Ansätze in maschineller Sprachübersetzung: Wissensrepräsentation und Textbezug,* pages 167-196. Niemeyer, Tübingen, 1986.

[Shieber *et al.,* 1990] Stuart M. Shieber, Gertjan van Noord, Robert C. Moore, and Fernando C. N. Pereira. A semantic-head-driven generation algorithm for unification-based formalisms. *Computational Linguistics,* 16(l):30-42, 1990.

[Steiner *et al.,* 1988] Erich Steiner, Ursula Eckert, Birgit Roth, and Jutta Winter-Thielen. The development of the Eurotra-D system of semantic relations. In E. Steiner, P. Schmidt, and C. Zelinsky-Wibbelt, editors, *From Syntax to Semantics—Insights From Machine Translation,* pages 40-104. Frances Pinter, London, 1988.

[Umbach, 1987] Carla Umbach. *Zur semantischen Interpretation in der Theorie der GPSG.* Techn. Univ. Berlin, Comp. Sc. Dept., KIT working paper 19, 1987.