STRUCTURE-DRIVEN GENERATION FROM SEPARATE SEMANTIC REPRESENTATIONS

Stephan Busemann
Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI) GmbH
Stuhlsatzenhausweg 3, D-6600 Saarbrücken 11
uucp: busemann@dfki.uni-sb.de

ABSTRACT

A new approach to structure-driven generation is presented that is based on a separate semantics as input structure. For the first time, a GPSG-based formalism is complemented with a system of pattern-action rules that relate the parts of a semantics to appropriate syntactic rules. This way a front end generator can be adapted to some application system (such as a machine translation system) more easily than would be possible with many previous generators based on modern grammar formalisms.[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* (LF)) 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 (for some discussion see e.g. [Shieber *et al.* 1990]). Most of this work shares a Montagovian view of semantics by assuming that LF be *integrated* into the grammar rules, thus assigning to each syntactic category its semantic representation.

Within this *integrated-semantics approach* the generation task mainly consists of reconstructing a given LF, 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.

This paper describes a different use of semantics for generation. Here the semantics is not part of the grammar, but rather expressed within a separate semantic representation language (abbrev.: SRL). 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 main goal of this paper is to describe a generator using a separate semantics and to suggest a *structure-driven strategy* that is based on a system of pattern-action (PA) rules, as they are known from AI production systems (for an overview see [Davis/King 1977]). The purpose of these rules is to explicitly relate the semantic (sub)structures to possible 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 producing 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.

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] This work was partially funded by the German Minister for Research and Technology (BMFT) under contract ITW 9002. Most of the research underlying this article was accomplished within the EUROTRA-D accompanying research project KIT-FAST at the Technical University of Berlin and funded by the BMFT under contract 1013211.

I wish to thank Christa Hauenschild, John Nerbonne, and Hans Uszkoreit for commenting on earlier versions of this paper.

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

(2) `vp([np(_)/Obj, 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 LF 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]). LF 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 SRL have to be adapted to LF. 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 SRL will typically contain application-dependent information that must be considered. Take, for instance, a transfer-based machine translation (MT) 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.[2]

[2] An exception is the MiMo2 system [Noord *et al.* 1990]. The price to pay for allowing transfer at the level of LF 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]).

The effort of introducing an additional step of processing can be saved to a large extent by adopting a separate-semantics approach. The SRL of some application system may directly serve as an *interface* to the generator.[3] In the case at hand, two additional components must be introduced into the generation scenario: the definition of SRL and PA rules. Instead of mapping SRL onto LF, SRL is directly related to syntax by virtue of the PA rules.

A STRUCTURE-DRIVEN GENERATOR

The generator to be described in this section is a module of the Berlin MT system [Hauenschild/Busemann 1988], which translates sentences taken from administrative texts in an EC corpus from German into English and vice versa.[4] The syntax formalism used is a constructive version of GPSG [Gazdar *et al.* 1985] as described in [Busemann/Hauenschild 1988]. The semantic representation language FAS (Functor-Argument Stuctures) [Mahr/Umbach 1990] is employed as an interface between three different processes: it is the target of GPSG-based analysis, for sentence-semantic transfer, and as the source for GPSG-based generation.

FAS is defined by context-free rule schemata with complex categories consisting of a main category (e.g. 'clause' in Figure 1a), which is associated with a fixed list of feature specifications.[5] The categories are in canonical order with the functor preceding all of its arguments. In contrast to syntactic structures where agreement relations are established by virtue of feature propagation, FAS categories 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 allows us to discriminate between different readings of words that result in different translational equivalents. Moreover, part of the thematic structure of the source language sentence is preserved during transfer and encoded by virtue of the feature 'them' with the numerical values indicating which portion should preferrably be presented first, second, third etc. The definitions of FAS for the German and English fragments mainly differ with regard to their terminal symbols.

[3] This interface does *not* correspond to the common separation between making decisions about what to say and how to say it (cf. [McKeown/Swartout 1988]). 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).

[4] The underlying view of MT is described in [Hauenschild 1988].

[5] In the present versions there are up to seven features in a FAS category. For sake of simplicity many details irrelevant to the present discussion are omitted in the examples.

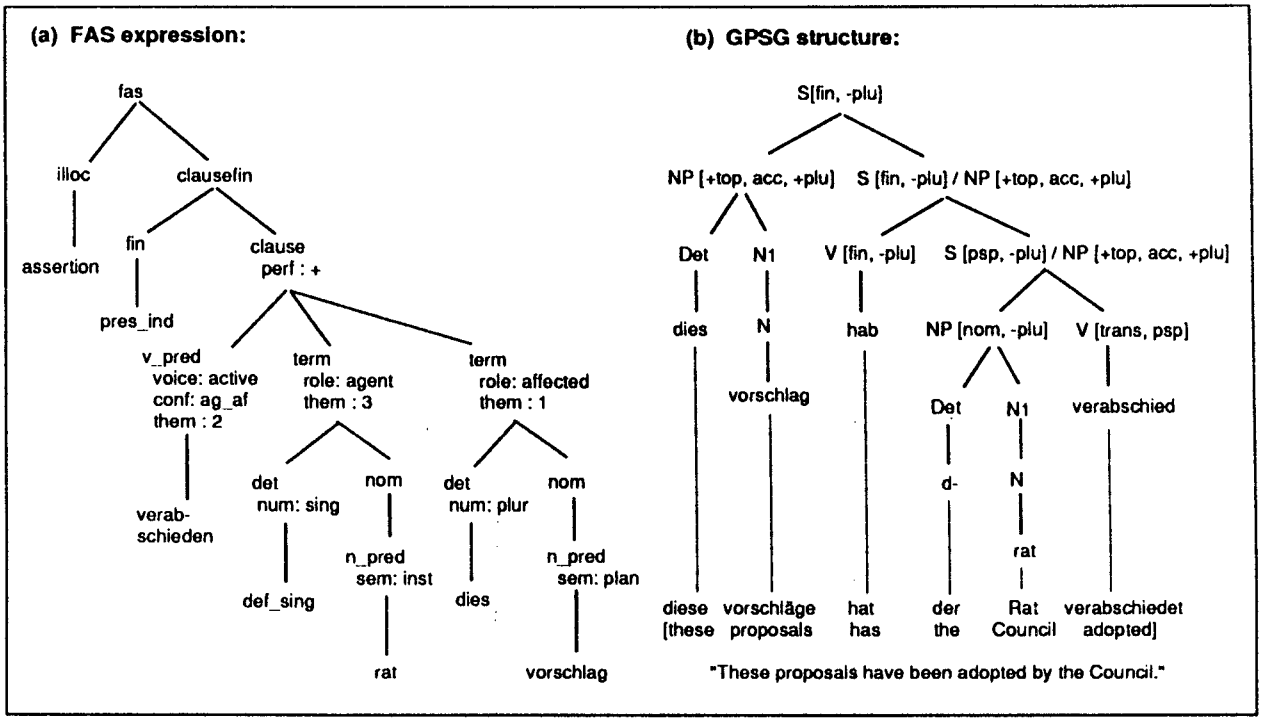

Figure 1: Sample FAS Expression (a) and Corresponding GPSG Structure (b).

The GPSG formalism used includes the ID/LP format, feature co-occurrence restrictions (FCRs) and universal principles of feature instantiation (FIPs). The ID rules are interpreted by the generator as providing the basic information for a local tree. The categories of each generated local tree are further instantiated by the FIPs and FCRs. Finally, the branches are ordered by virtue of the LP statements.

Strategies for structure building and feature instantiation. The task of constructing an admissible GPSG syntactic structure can be divided up into the following subtasks that can be performed independently of each other, and each according to its own processing strategy:

- Structure building (by virtue of PA rules, which in turn use ID rules)
- Feature instantiaton and ordering of the branches (by virtue of FIPs, FCRs and LP statements)

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.

Before justifying the result let us have a closer look at the structure-building algorithm. The *overall syntactic structure* (OSS) is successively construed in a top-down manner. At each level there is a set of nonterminal leaf nodes available serving as *attachment points* for further expansion steps (initially the empty category is the only attachment point). An expansion step consists of

1. generating a local tree t by virtue of an ID rule,
2. unifying its mother node with one of the attachment points,
3. removing the attachment point from the current set,
4. defining the daughters of t as the new current set of attachment points.

Since lexical entries terminate a branch of the OSS, the fourth of the above points is dropped during expansion of lexical categories: processing continues with the reduced set of attachment points.

Feature instantiation and the ordering of branches take place in a bottom-up manner after a local tree has no further attachment points associated with it (i.e. all of its daughters have been expanded). Then processing returns to the next higher level of the OSS examining the set of attachment points. Depending on whether or not it is empty, the next step is either feature instantiation or structure building. Given this interlinking of the two subtasks, an OSS is admitted by the grammar if

its top-most local tree has passed feature instantiation.

The effects of feature instantiation with respect to the German example in Figure 1b[6] can be better understood with the help of the S-expansion rules used; cf. (3)–(5).[7] Rule (3) causes topicalization, (4) introduces a perfect auxiliary, and (5) requires a transitive verb whose object is topicalized.

(3) S ⟶ X[+top], S[fin] / X[+top]

(4) S ⟶ V, S[psp]

(5) S / NP[+top, acc] ⟶ NP[nom], V[trans]

The solution will now be justified. First of all, note that the top-most part of an FAS expression is related to the top-most part of the GPSG structure, and that the leaves of a FAS expression usually correspond to GPSG lexicon entries. As a consequence, the order the FAS expression is traversed determines the order in which the structure-building subtask is performed. Why should then, in the case of FAS, the traversal occur top-down?

The answer is motivated by the distribution of information in FAS expressions. In order to apply a certain ID rule deterministically, information from distant portions of the FAS expression may be needed. For instance, the FAS specification ⟨them : 1⟩, which is part of one of the daughters of `clause` in Figure 1a, is interpreted as requiring topicalization of a syntactic constituent under the condition that a declarative sentence is being generated. This latter information is, however, only available at the `[illoc [assertion]]`[8] part of the FAS expression (cf. Figure 1a).

Two possible methods for collecting this 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`. In that case, all the information needed is present, and the traversal of the FAS expression could occur bottom-up as well as top-down. 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 *auxiliary storage* is needed. 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. Now, the ID rule for topicalization (3) is triggered iff 'declarative' can be retrieved from the storage.

If the necessary information is not available yet, one must accept either a delay of a mapping or backtracking. With a top-down traversal of FAS expressions, however, such cases are sufficiently restricted to ensure efficiency. Note that a bottom-up traversal or a mixed strategy could be more efficient if the distribution of information in the SRL were different.

The problems observed with top-down generators using an integrated semantics cannot occur in the separate-semantics approach. Expansion of grammar rules can be controlled by the semantic representation if each rule application is explicitly triggered. 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.

Let us now discuss why feature instantiation should be a bottom-up process. The FIPs 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 FIPs 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.

Pattern-action rules. A PA rule is a production rule with a *pattern* for local FAS trees as its left-hand side and two sets of actions as its right-hand side. The *information-gathering actions* (IGAs) maintain the auxiliary storage. The *structure-building actions* (SBAs) generate GPSG trees. Either one of these sets may be empty.

In order to minimize the power of PA rules, the inventory of IGAs and SBAs is restricted. There are only three IGAs for storing information into and removing from the auxiliary storage. The auxiliary storage is a two-dimensional array of a fixed size. It may contain atomic values for a set of features predetermined by the PA rule writer as well as a single GPSG category. There are only five SBAs for different kinds of mapping, three of which are explained below; cf. [Busemann 1990] for a comprehensive discussion. Any SBA will remove the stored category

[6] These are not shown for the constituents of NPs.

[7] Note the different use of the symbol '/': here it denotes the category-valued feature 'slash'.

[8] Square brackets are used here to indicate tree structure.

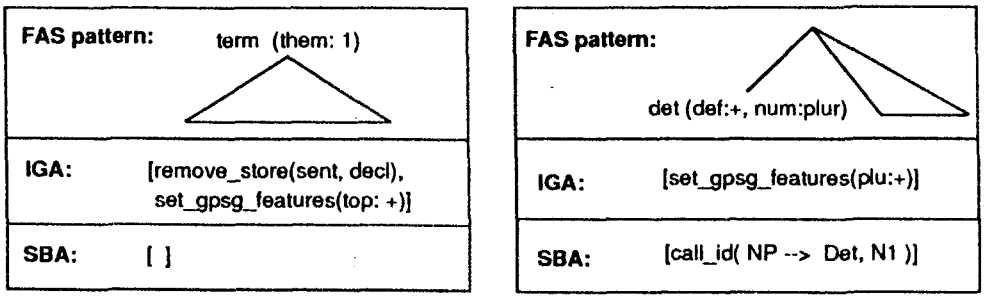

Figure 2: Two Pattern-Action Rules for NP-Topicalization.

from the storage and unify it with the mother of the local tree it is about to generate.

To illustrate this let us return to the topicalization example. The responsible PA rules are shown in Figure 2. The pattern of the first one matches any local FAS tree whose mother is a `term(them:1)`. The IGAs work as follows: If a specification ⟨sent : decl⟩ can be removed from the storage, the GPSG feature specification [+top] will be added to the stored category (by virtue of the IGA `set_gpsg_features`). The SBA set is empty. The second PA rule matches any local FAS tree whose first daughter is a definite determiner with plural number followed by zero or more daughters. Note that both patterns match the same local tree of the FAS expression in Figure 1a. There is only one IGA, which adds the number information to the stored GPSG category. The single SBA, `call_id`, states that a local GPSG tree is generated by virtue of the ID rule indicated and added to the OSS. Since the mother of the local tree (NP) now contains the specification [+top], it can only unify with the 'slash' value introduced by the mother of rule (5). Fronting of the NP is achieved in accordance with the FIPs and LP statements.

Three kinds of PA rules should be distinguished according to the effects of their SBAs. Figure 2 shows two of them; the first one doesn't create structure at all while the second one transduces a (FAS) local tree into a (GPSG) local tree. A third type of rules generates GPSG structure out of FAS feature specifications. Figure 1 shows its use to generate the non-local subtree including the perfect auxiliary `[s [v [hab], s(psp)]]` from the local FAS tree dominated by `clause(perf:+)`. Note that this PA rule must be applied before an attempt is started to attach the subtree `[s/np(acc) [np(nom), v(trans)]]`. This latter subtree is generated by a PA rule whose pattern matches the same FAS tree as the previous one. We shall return to this problem in the following section.

Controlling the mapping procedure. First of all note that PA rules can communicate with each other only indirectly, i.e. by modifying the content of the auxiliary storage or by successfully applying an SBA, thereby creating a situation in which another rule becomes applicable (or cannot be applied anymore). PA rules do not contain any control knowledge.

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 and an SBA—if present—is successful. An SBA is successful iff a syntactic subtree can be attached to the OSS as described above.

Since the set of PA rules is not commutative, the order of application is crucial in order to ensure that n is maximal. Due to the restricted power of the PA rules possible conflicts can be detected and resolved *a priori*. A conflict arises if more than one pattern matches a given FAS tree. All FAS trees matched by more than one pattern can be identified with help of the FAS grammar. The respective PA rules are members of the same conflict set. The elements of a conflict set can be partially ordered by virtue of precedence rules operating on pairs of PA rules.

For instance, the conflict regarding the perfect auxiliary is resolved by making a precedence rule check the ID rules that would be invoked by the respective SBAs. If the mother of the second one can be unified with a daughter of the first one and not vice versa, then the first PA rule must be applied before the second one. Thus a PA rule with an SBA invoking ID rule (4) will apply before another one with an SBA invoking ID rule (5).

Note that, in this example, the number of successful PA rules would not be maximal if the order of application was the other way around since the SBA invoking ID rule (4) would not succeed anymore.

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 OSS 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.

CONCLUSION

The adaptation of a GPSG-based generator to an MT system using FAS as its SRL was described as an instance of the separate-semantics approach to surface generation. In this instance, the OSS is most efficiently built top-down whereas feature instantiation is performed bottom-up.

The mapping based on PA rules has proved to be efficient in practice. There are only a few cases where backtracking is required; most often the local FAS tree being verbalized allows together with the contents of the auxiliary storage and the current set of attachment points for a deterministic choice of grammar rules.

The generator has been fully implemented and tested with middle-sized fragments of English and German. It is part of the Berlin MT system and runs on both an IBM 4381 under VM/SP in Waterloo Core Prolog and a PC XT/AT in Arity Prolog.

Compared to algorithms based on an integrated semantics the separate-semantics approach pursued here is promising if the generator has to be adapted to the SRL of some application system. Adaptation then consists in modifying the set of PA rules rather than in rewriting the grammar.

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 natürlicher Sprache mit Generalisierten Phrasenstruktur-Grammatiken*, Doctoral Dissertation, Universität des Saarlandes, Saarbrücken. Also available: TU Berlin, Dept. of Computer Science, KIT Report 87.

[Busemann/Hauenschild 1988] Stephan Busemann and 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, 300–332.

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

[Hauenschild 1988] Christa Hauenschild (1988), 'Discourse Structure—Some Implications for Machine Translation', in D. Maxwell, K. Schubert und A. P. M. Witkam (eds.), *New Directions in Machine Translation*, Dordrecht: Foris, 145–156.

[Hauenschild/Busemann 1988] Christa Hauenschild and Stephan Busemann (1988), 'A Constructive Version of GPSG for Machine Translation', in E. Steiner, P. Schmidt, and C. Zelinsky-Wibbelt (eds.), *From Syntax to Semantics—Insights From Machine Translation*, London: Frances Pinter, 216–238.

[Mahr/Umbach 1990] Bernd Mahr and Carla Umbach (1990), 'Functor-Argument Structures for the Meaning of Natural Language Sentences and Their Formal Interpretation', in K.-H. Bläsius, U. Hedstück, and C.-R. Rollinger (eds.), *Sorts and Types in Artificial Intelligence*, Berlin, New York: Springer (Lecture Notes in Artificial Intelligence 418), 286–304.

[McKeown/Swartout 1988] Kathleen R. McKeown and William R. Swartout, 'Language Generation and Explanation', in M. Zock and G. Sabah (eds.), *Advances in Natural Language Generation. An Interdisciplinary Perspective. Vol. 1*, London: Frances Pinter, 1–52.

[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, Pim 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(1), 30–42.