REVERSIBILITY AND MODULARITY IN NATURAL LANGUAGE GENERATION

Günter Neumann

Lehrstuhl für Computerlinguistik
SFB 314, Projekt N3 BiLD
Universität des Saarlandes
Im Stadtwald 15
D-6600 Saarbrücken 11, FRG
neumann@coli.uni-sb.de

ABSTRACT

A consequent use of reversible grammars within natural language generation systems has strong implications for the separation into strategic and tactical components. A central goal of this paper is to make plausible that a uniform architecture for grammatical processing will serve as a basis to achieve more flexible and efficient generation systems.

1 Introduction

In general, the goal of parsing is the derivation of all possible grammatical structures defined by a grammar of a given string α (i.e. especially the determination of all possible logical forms of α) and the goal of the corresponding generation task is the computation of all possible strings defined by a grammar of a given logical form Φ that are logically equivalent to Φ (see also (Shieber, 1988), (Calder *et al.*, 1989)). Recently, there is a strong tendency to use the same grammar for performing both tasks. Besides more practically motivated reasons - obtaining more compact systems or avoidance of inconsistencies between the input and output of a system - there are also theoretical (a single model of language behaviour) and psychological evidences (empirical evidence for shared processors or facilities, cf. (Garrett, 1982), (Frazier, 1982), (Jackendoff, 1987)) to adopt this view.

From a formal point of view the main interest in obtaining non-directional grammars is the specification of the relationship between strings and logical forms.[1] According to van Noord (1990), a grammar is reversible if the parsing and generation problem is computable and the relation between strings and logical forms is symmetric. In this case parsing and generation are viewed as mutually inverse processes.

Furthermore there are also approaches that assume that it is possible to use the same algorithm for processing the grammar in both directions (e.g. (Hasida and Isizaki, 1987), (Shieber, 1988), (Dymetman *et al.*, 1990), (Emele and Zajac, 1990)). A great advantage of a uniform process is that a discourse and task independent module for grammatical processing is available. This means that during performing both tasks the same grammatical power is potentially disposable (regardless of the actual language use).

Nevertheless, in most of the 'real' generation systems where all aspects of the generation process of natural language utterances are considered, grammars are used that are especially designed for generation purposes (cf. (Hovy, 1987), (Dale, 1990), (Horacek, 1990), (McKeown *et al.*, 1990), (Reithinger, 1991)).[2]

The purpose of this paper is to show that the use of a uniform architecture for grammatical processing has important influences for the whole generation task. A consequent use of a uniform process within a natural language generation system affects the separation into strategic and tacti-

[1] I assume a notion of grammars that integrate phonological, syntactical and semantical levels of description, e.g., (Pollard and Sag, 1987).

[2] But it is important to note here, that most of the proposed grammars are unification-based which is an important common property with respect to current parsing grammars.

cal components. On the one hand, existing problems with this separation emerge, on the other hand uniform architectures will serve as an important (linguistic) basis to achieve first solutions for the problems.

In the next section I will discuss important problems and restrictions with the modular design of current generation systems and will then show why a uniform architecture as the grammatical basis can contribute to solutions of the problems.

2 Modularity in Generation Systems

The Problem It is widely accepted to cut down the problem of natural language generation (NLG) into two subtasks:

- determination of the content of an utterance
- determination of its linguistic realization

This 'divide and conquer' view of generation is the base of current architectures of systems. With few exceptions (e.g., (Appelt, 1985)) the following two components are assumed:

- 'what to say' part (*strategic component*)
- 'how to say it' part (*tactical component*)

But, as it has been demonstrated by some authors ((Appelt, 1985), (Hovy, 1987), (Rubinoff, 1988), (Neumann, 1991), (Reithinger, 1991)) it is not possible to separate the two phases of the generation process completely, e.g., in the case of lexical gaps, choice between near synonyms or paraphrases.

Currently, in systems where the separation is advocated the problems are sometimes 'solved' in such a way that the strategic component has to provide all information needed by the tactical component to make decisions about lexical and syntactic choices (McDonald, 1983), (McKeown, 1985), (Busemann, 1990), (Horacek, 1990). As a consequence, this implies that the input for tactical components is tailored to determine a good sentence, making the use of powerful grammatical processes redundant. In such approaches, tactical components are only front-ends and the strategic component needs detailed information about the language to use.

Hence, they are not separate modules because this implies that both components share the grammar. As pointed out in Fodor (1983) one of the characteristic properties of a module is that it is *computationally autonomous*. But a relevant consideration of computationally autonomy is that modules do not share sources (in our case the grammar).

Looking for More Symmetric Architectures To maintain the modular design a more symmetric division into strategic and tactical separation is needed:

- Strategic component $\longrightarrow$ primarly concerned with conceptual decisions
- Tactical component $\longrightarrow$ primarly concerned with linguistic decisions

A consequence of this view is that the strategic component has no detailed information about the specific grammar and lexicon. This means that in general a message which is constructed precisely enough to satisfy the strategic component's goal can be underspecified from the tactical viewpoint. For example, if the strategic component specifies as input to the tactical component that 'Peter loves Maria', and 'Maria' is the current focus, then in German it is possible to utter:

1 Maria wird von Peter geliebt
'Maria is loved by Peter'

or

2 Maria liebt Peter
'Maria, Peter loves'

Of course, a 'real' generation system needs to choose between the possible paraphrases. An adequate generation system should avoid to utter 2 because for this utterance there exists also the unmarked reading that 'Maria loves Peter'. As long as the strategic component has no detailed knowledge of a specific grammar it could not express 'choose the passive form to avoid ambiguity'. But then the process can only choose randomly between paraphrases during generation and this means that the underlying message will possibly not be conveyed.

There is also psychologically grounded evidence to assume that the input to a tactical component might not be necessary and sufficient to make linguistic decisions. This is best observed in exam-

ples of self–correction (Levelt, 1989). For example, in the following utterance:[3]

> "but aaa, bands like aaa- aaa- aaa- errr- like groups, not bands, - groups, you know what I mean like aaa."

the speaker discovers two words (the near-synonymous 'group' and 'band') each of which comes close to the underlying concept and has problems to decide which one is the most suitable. In this case, the problem is because of a mis–match between what the strategic component want to express and what the language is capable to express (Rubinoff, 1988).

Current Approaches In order to be able to handle these problems, more flexible tactical components are necessary that are able to handle e.g. underspecified input. In (Hovy, 1987), (Finkler and Neumann, 1989) and (Reithinger, 1991) approaches are described how such more flexible components can be achieved. A major point of these systems is to assume a bidirectional flow of control between the strategic and the tactical components.

The problem with systems where a high degree of feedback between the strategic and the tactical components is necessary in order to perform the generation task is that one component could not perform its specific task without the help of the other. But when the mode of operation of e.g. the tactical component is continuously influenced by feedback from the strategic component then the tactical component will lose its autonomy and consequently this means that it is not a module (see also (Levelt, 1989)).

3 Integration of Parsing and Generation

A promising approach for achieving more autonomous tactical components is to integrate generation and parsing in a more strict way. By this I mean:

- the use of resulting structures of one direction directly in the other direction,
- the use of one mode of operation (e.g., parsing) for monitoring the other mode (e.g., generation).

A main thesis of this paper is that the best way to achieve such integrated approach is to use a uniform grammatical process as the linguistic basis of a tactical component.

Use of Same Structures in Both Directions If parsing and generation share the same data (i.e. grammar and lexicon) then it is possible that resulting structures of one direction could be used directly in the other direction. For example, during the generation of paraphrases of the ambiguous utterance 'Remove the folder with the system tools.' the generator can use directly the analysed structures of the two NPs 'the folder' and 'the system tools' in the corresponding paraphrases. In a similiar way parsing and generation of elliptic utterances can also be performed more efficiently. For example, consider the following dialog between person A and B:

A: 'Peter comes to the party tonight.'

B: 'Mary, too.'

In order to be able to parse B's utterance A can directly use parts of the grammatical structure of his own utterance in order to complete the elliptic structure.[4]

Adaptability to Language Use of Others Another very important argument for the use of uniform knowledge sources is the possibility to model the assumption that during communication the use of language of one interlocutor is influenced by means of the language use of the others.

For example, in a uniform lexicon it does not matter wether the lexeme was accessed during parsing or generation. This means that the use of linguistic elements of the interlocutor influences the choice of lexical material during generation if the frequency of lexemes will serve as a decision criterion. This will help to choose between lexemes which are synonymous in the actual situation or when the semantic input cannot be sufficiently specified. E.g. some drinking-devices can be denoted either 'cup' or 'mug' because their

[3] This example is taken from Rubinoff (1988) and is originally from a corpus of speech collected at the University of Pennsylvania.

[4] In this particular case, A can use the whole VP 'will come to the party'. In general the process is more complicated e.g., if B's answer would be 'Mary and John, too'.

shape cannot be interpreted unequivocally. An appropriate choice would be to use the same lexeme that was previously used by the hearer (if no other information is available), in order to ensure that the same object will be denoted. In principle this is also possible for the choice between alternative syntactic structures.

This adaptability to the use of language of partners in communication is one of the sources for the fact that the global generation process of humans is flexible and efficient. Of course, adaptability is also a kind of co-operative behaviour. This is necessary if new ideas have to be expressed for which no mutually known linguistic terms exist (e.g., during communication between experts and novices). In this case adaptability to the use of language of the hearer is necessary in order to make possible that the hearer will be able to understand the new information.

In principle this kind of adaptability means that the structures of the input computed during the understanding process carry some information that can be used to parametrize the generation process. This leads to more flexibility: not all necessary parameters need to be specified in the input of a generator because decision points can also be set during run-time.

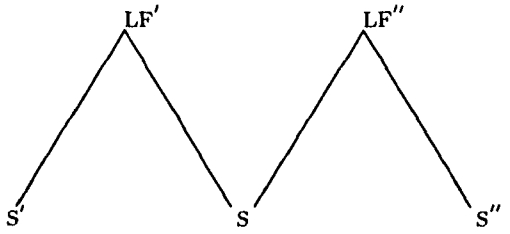
This dynamic behaviour of a generation system will increase efficiency, too. As McDonald et al. (1987) define, one generator design is more efficient than another, if it is able to solve the same problem with fewer steps. They argue that"the key element governing the difficulty of utterance production is the degree of familiarity with the situation". The efficiency of the generation process depends on the competence and experience one has acquired for a particular situation. But to have experience in the use of linguistic objects that are adequate in a particular situation means to be adaptable.

Monitoring As Levelt (1989) pointed out "speakers monitor what they are saying and how they are saying it", i.e. they monitor not only for meaning but also for linguistic well-formedness.

To be able to monitor what one is saying is very important for processing of underspecified input and hence for achieving a more balanced divison of the generation task (see sec. 2). For example to choose between the two paraphrases of the example in sec. 2, the tactical component could parse the resulting strings in order to decide to choose the less ambiguous string 'Mary is loved by Peter.' It only needs to know from the strategic component that unambiguous utterances are preferred (as a pragmactic constraint).

In Levelt's model parsing and generation are performed in an isolated way by means of two different grammars. In such flow of control the complete structure has to be generated again if ambiguities are detected that have to be avoided.

If, for example an intelligent help-system that supports a user by using an operation research system (e.g. Unix, (Wilensky *et al.*, 1984)), receives as input the utterance "Remove the folder with the system tools" then the system is not able to perform the corresponding action directly because it is ambiguous. But the system could ask the user "Do you mean 'Remove the folder by means of the system tools' or 'Remove the folder that contains the system tools' ". This situation is summarized in the following figure (LF' and LF'' symbolize two readings of S):



S: Remove the folder with the system tools

S': Remove the folder by means of the system tools

S'': Remove the folder that contains the system tools

Figure 1: Relationship between ambiguities and paraphrases

If parsing and generation are performed in an isolated way then generation of paraphrases can be very inefficient, because the source of the ambiguous utterance S is not used directly to guide the generation process.

Generation of Paraphrases In order to clarify, why an integrated approach can help to solve the problem I will consider the problem of generation of paraphrases in more detail.

If a reversible grammar is used in both directions then the links between the strings and logi-

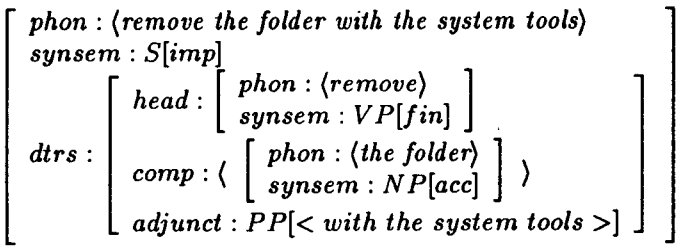


Figure 2: 'with the system tools' in modifier position of the VP

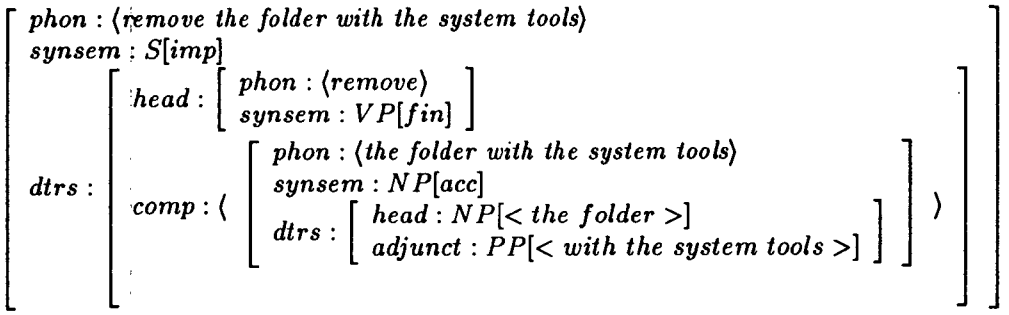


Figure 3: The same PP as a modifier of the NP 'the folder'

cal forms in fig. 1 are bidirectional.[5]

A first naive algorithm that performs generation of paraphrases using a reversible grammar can be described as follows: Suppose S is the input for the parser then the set

$$\{(S, LF'), (S, LF'')\}$$

is computed. Now LF′ respectively LF″ is given as input to the generator to compute possible paraphrases. The sets

$$\{(LF', S'), (LF', S)\}$$

respectively

$$\{(LF'', S), (LF'', S'')\}$$

result. By means of comparison of the elements of the sets obtained during generation with the set obtained during parsing one can easily determine the two paraphrases S′ and S″ because of the relationship between strings and logical forms defined by the grammar.

This algorithm is naive because of the assumption that it is possible to generate all possible paraphrases at once. Although 'all–parses' algorithms are widley used during parsing in natural language systems a corresponding 'all–paraphrases' strategy is not practicle because in general the search space during generation is much larger (which is a consequence of the modular design discussed in sec. 2).

Of course, from a formal point of view one is interested in algorithms that compute all grammatically well–formed structures – at least potentially. So most of the currently developed generators and uniform algorithms assume – more or less explictly – an all–paraphrases strategy (e.g., (Shieber, 1988), (Calder *et al.*, 1989), (Shieber *et al.*, 1989), (Dymetman *et al.*, 1990), (Emele and Zajac, 1990)). But from a practical point of view they are not directly usable in such specific situations.

[5] It is not of central role here wether the 'competence' grammar is actually compiled in two efficient parsing and generation grammars as long as the symmetry property is not affected. This inherent property of a reversible grammar is very important in the case of generation of paraphrases because it ensures that the ambiguous structure and the corresponding paraphrases are related together. If this would not be the case then this would mean that one is only able to generate the paraphrases but not the ambiguous structure.

More Suitable Strategies A more suitable strategy would be to generate only one paraphrase for each ambiguous logical form. As long as parsing and generation are performed in an isolated way the problem with this strategy is that there is no control over the choice of paraphrases. In order to make clear this point I will look closer to the underlying structure of the example's utterances.

The problem why there are two readings is that the PP 'with the system folder' can be attached into modifier position of the NP 'the folder' (expressing the semantic relation that 'folder' contains 'system tools') or of the verb 'remove' (expressing semantically that 'folder' is the instrument of the described situation). Fig. 2 and 3 (see above) show the internal grammatical structure in a HPSG–style notation (omitting details, that are not relevant in this context).

As long as the source of the ambiguity is not known it is possible to generate in both cases the utterance 'Remove the folder with the system–tools' as a paraphrase of itself. Of course, it is possible to compare the resulting strings with the input string S. But because the source of the ambiguity is not known the loop between the isolated processes must be performed several times in general.

A better strategy would be to recognize relevant sources of ambiguities during parsing and to use this information to guide the generation process. Meteer and Shaked (1988) propose an approach where during the repeated parse of an ambiguous utterance potential sources of ambiguity can be detected. For example when in the case of lexical ambiguity a noun can be associated to two semantic classes a so called 'lexical ambiguity specialist' records the noun as the ambiguity source and the two different classes. These two classes are then explicitly used in the generator input and are realized as e.g. modifiers for the ambiguous noun.

The only common knowledge source for the paraphraser is a high order intensional logic language called World Model Language. It serves as the interface between parser and generator. The problem with this approach is that parsing and generation are performed in an isolated way using two different grammars. If an ambiguous utterance S need to be paraphrased S has to be parsed again. During this repeated parse *all* potential ambiguities have to be recognized and recorded (i.e. have to be *monitored*) by means of different 'ambiguity specialists'. The problem here is that also local ambiguities have to be considered that are not relevant for the whole structure.

An Alternative Approach I will now describe the basic idea of an approach that is based on an integrated approach where both tasks share the same grammar. The advantage of this approach is that no repeated parse is necessary to compute potential ambiguity sources because the different grammatical structures determined during parsing are used directly to *guide* the generation process. By means of this strategy it is also ensured that an ambiguous utterance is not generated as a paraphrase of itself.

In principle the algorithm works as follows: During the generation of paraphrases the generation process is monitored in such a way that the monitor compares in each step the resulting structures of the generation process with the corresponding structures from parsing maintained in the alternative parse trees (I will now assume that two parse trees P_1 and P_2 corresponding to the structures given in fig. 2 and 3 are obtained during parsing). Suppose that LF' (cf. fig. 1) is specified as the input to the generator. In the case where the generator encounters alternative grammatical structures to be expanded, the monitor guides the generator by means of inspection of the corresponding parse trees. In the case where actual considered parts p_1 and p_2 of P_1 and P_2 (e.g., same NPs) are equal the generator has to choose the same grammatical structure that was used to build p_1 and p_2 (or more efficiently the generator can use the partial structure directly as a kind of compiled knowledge). In the case where a partial structure of e.g. parse tree P_1 has no correspondence in P_2 (cf. fig. 2 and 3) an ambiguity source is detected. In this case an alternative grammatical structure has to be chosen.[6]

At this point it should be clear that the easiest way in order to be able to generate 'along parsed structures' is to use the same grammar in both directions. In this case grammatical structures obtained during parsing can be used directly to restrict the potential search space during generation.

[6]Of course, the described algorithm is too restrictive, in order to handle non–structural (e.g. contextual) paraphrases. But, I assume that this approach is also applicable in the case of lexical amibiguities prerequisite word meanings are structurally described by means of lexical semantics (e.g., Jackendoff's Lexical Conceptual Structures (Jackendoff, 1990))

This approach is not only restricted in cases where the input is ambiguous and the paraphrases must contrast the different meanings. It can also be used for *self-monitoring* when it has to be checked wether a produced utterance S of an input form LF is ambiguous. In this case S will be parsed. If during parsing e.g., two readings LF and LF' are deduced LF is generated again along the parse tree obtained for S. Now an utterance S' can be generated that has the same meaning but differs with respect to the ambiguity source of S.

4 Current Work

We have now started a project called BiLD (short for Bidirectional Linguistic Deduction) at the University of Saarland (Saarbrücken) where it will be investigated how an integrated approach of parsing and generation can be realized efficiently by means of a uniform architecture and how such a model can be used for increasing flexibility and efficiency during natural language processing.

The main topic of the BiLD project is the development of a uniform parametrized deduction process for grammatical processing. This process constitutes the core of a flexible and symetric tactical module. In order to realize the integrated approach and to obtain a high degree of efficiency in both directions we will develop methods for a declarative encoding of information of control in the lexicon and grammar.

We follow a sign-based approach for the description of linguistic entities based on Head-driven Phrase Structure Grammar (Pollard and Sag, 1987) and the variant described in Reape (1989). Besides theoretical reasons there are also reasons with respect to system's design criterions to adopt this view because all levels of descriptions (i.e. phonological, syntactic and semantic structure) of linguistics entities (i.e. words and phrases) are described simultanueous in a uniform way by means of partial information structures. None of the levels of description has a privileged status but expresses possible mutually co-ocurrence restrictions of structures of different levels.

Furthermore a high degree of lexicalism is assumed so that the lexicon as a complex hierachical ordered data structure plays a central role in BiLD. As it has been shown this lexicalized view supports reversibility (cf. (Newman, 1990), (Dymetman *et al.*, 1990)) and the performing of specific processing strategies (e.g., incremental and parallel generation, (Neumann and Finkler, 1990)).

The task of the deduction process during generation is to construct the graphemic form of a specified feature description of a semantic form. For example, to yield the utterance "A man sings." the deduction process gets as input the semantic feature structure

$$\left[\begin{array}{l} rel : sing' \\ agens : \left[\begin{array}{l} quant : exists' \\ var : \boxed{1} \\ restr : \left[\begin{array}{l} pred : man' \\ var : \boxed{1} \end{array}\right] \end{array}\right] \end{array}\right]$$

and deduces the graphematic structure

$$[\ graph : \langle A\_man\_sings. \rangle\ ]$$

by means of successive application of lexical and grammatical information. In the same way the deduction process computes from the graphematic structure an appropriate semantic structure in parsing direction. A first prototype based on head-driven bottom-up strategy is now under development (cf. (van Noord, 1991)).

A major aspect of the BiLD project is that a specific parametrization of the deduction process is represented in the lexicon as well as in the grammar to obtain efficient structures of control (Uszkoreit, 1991). The main idea is that preference values are assigned to the elements (disjuncts or conjuncts) of feature descriptions. For example, in HPSG all lexical entries are put together into one large disjunctive form. From a purely declarative point of view these elements are unordered. But a preference structure is used during processing in order to guide the process of lexical choice efficiently which itself influences the grammatical process.

5 Conclusion

A main thesis of this paper was to show that existing problems with the modular design of current generation systems emerge when a reversible grammar is used. In order to maintain the modular design I have proposed an approach that is based on a strong integration of parsing and generation of grammatical structures using a reversible grammar and monitoring mechanisms.

By means of such an integrated approach performing of e.g. generation of paraphrases can be done more easier and efficently.

Acknowledgements

This research was supported by SFB 314, Project N3 BiLD.

Bibliography

Douglas E. Appelt. *Planning English Sentences.* Cambridge University Press, Cambridge, 1985.

Stefan Busemann. *Generierung natürlicher Sprache mit Generalisierten Phrasenstruktur-Grammatiken.* PhD thesis, University of Saarland (Saarbrücken), 1990.

Jonathan Calder, Mike Reape, and Henk Zeevat. An algorithm for generation in unification categorial grammar. In *Fourth Conference of the European Chapter of the Association for Computational Linguistics*, pages 233–240, Manchester, 1989.

Robert Dale. Generating receipes: An overview of epicure. In Robert Dale, Chris Mellish, and Michael Zock, editors, *Current Research in Natural Language Generation*, pages 229–255. Academic Press, London, 1990.

Marc Dymetman, Pierre Isabelle, and Francois Perrault. A symmetrical approach to parsing and generation. In *Proceedings of the 13th International Conference on Computational Linguistics (COLING)*, pages 90–96, Helsinki, 1990.

Martin Emele and Rémi Zajac. Typed unification grammars. In *Proceedings of the 13th International Conference on Computational Linguistics (COLING)*, pages 293–298, Helsinki, 1990.

Wolfgang Finkler and Günter Neumann. Popelhow: A distributed parallel model for incremental natural language production with feedback. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 1518–1523, Detroit, 1989.

Jerry A. Fodor. *The Modularity of Mind: An Essay on Faculty Psychology.* A Bradford Book, MIT Press, Cambridge, Massachusetts, 1983.

Lyn Frazier. Shared components of production and perception. In M. A. Arbib et al., editor, *Neural Models of Language Processes*, pages 225–236. Academic Press, New York, 1982.

Merrill F. Garrett. Remarks on the relation between language production and language comprehension systems. In M. A. Arbib et al., editor, *Neural Models of Language Processes*, pages 209–224. Academic Press, New York, 1982.

K. Hasida and S. Isizaki. Dependency propagation: A unified theory of sentence comprehension and generation. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, pages 664–670, Mailand, 1987.

Helmut Horacek. The architecture of a generation component in a complete natural language dialogue system. In Robert Dale, Chris Mellish, and Michael Zock, editors, *Current Research in Natural Language Generation*, pages 193 – 227. Academic Press, London, 1990.

Eduard. H. Hovy. *Generating Natural Language under Pragmatic Constraints.* PhD thesis, Yale University, 1987.

Ray Jackendoff. *Consciousness and the Computational Mind.* MIT Press, Cambridge, Massachusetts, 1987.

Ray Jackendoff. *Semantic Structures.* MIT Press, Cambridge, Massachusetts, 1990.

Willem J. M. Levelt. *Speaking: From Intention to Articulation.* MIT Press, Cambridge, Massachusetts, 1989.

David D. McDonald. Natural language generation as a computational problem: An introduction. In M. Brady and C. Berwick, editors, *Computational Models of Discourse.* MIT Press, Cambridge, Massachusetts, 1983.

David D. McDonald, Marie W. Meteer, and James D. Pustejovsky. Factors contributing to efficiency in natural language generation. In K. Kempen, editor, *Natural Language Generation: New Results in Artificial Intelligence, Psychology and Linguistics*, pages 159–182. Martinus Nijhoff, Dordrecht, 1987.

Kathleen R. McKeown. *Text Generation: Using Discourse Strategies and Focus Constraints to Generate Natural Language Text.* Cambridge University Press, Cambridge, 1985.

Kathleen R. McKeown, Michael Elhadad, Yumiko Fukomoto, Jong Lim, Christine Lombardi, Jacques Robin, and Frank Smadja. Natural language generation in comet. In Robert Dale, Chris Mellish, and Michael Zock, editors, *Current Research in Natural Language Generation*, pages 103 – 139. Academic Press, London, 1990.

Marie Meteer and Varda Shaked. Strategies for effective paraphrasing. In *Proceedings of the 12th International Conference on Computational Linguistics (COLING)*, Budapest, 1988.

Günter Neumann. A bidirectional model for natural language processing. In *Fifth Conference of the European Chapter of the Association for Computational Linguistics*, pages 245–250, Berlin, 1991.

Günter Neumann and Wolfgang Finkler. A head-driven approach to incremental and parallel generation of syntactic structures. In *Proceedings of the 13th International Conference on Computational Linguistics (COLING)*, pages 288–293, Helsinki, 1990.

Paula Newman. Towards convenient bi-directional grammar formalisms. In *Proceedings of the 13th International Conference on Computational Linguistics (COLING)*, pages 294–298, Helsinki, 1990.

Carl Pollard and Ivan Sag. *Information Based Syntax and Semantics, Volume 1.* Center for the Study of Language and Information Stanford, 1987.

Mike Reape. A logical treatment of semi-free word order and bounded discontinuous constituency. In *Fourth Conference of the European Chapter of the Association for Computational Linguistics*, pages 103–110, Manchester, 1989.

Norbert Reithinger. Popel: A parallel and incremental natural language generation system. In C. L. Paris et al., editor, *Natural Language Generation in Artificial Intelligence and Computational Linguistics*, pages 179–199. Kluwer, 1991.

Robert Rubinoff. A cooperative model of strategy and tactics in generation. In *Paper presented at the Fourth International Workshop on Natural Language Generation*, Santa Catalina Island, 1988.

Stuart M. Shieber. A uniform architecture for parsing and generation. In *Proceedings of the 12th International Conference on Computational Linguistics (COLING)*, Budapest, 1988.

Stuart M. Shieber, Gertjan van Noord, Robert C. Moore, and Fernando C.N. Pereira. A semantic-head-driven generation algorithm for unification based formalisms. In *27th Annual Meeting of the Association for Computational Linguistics*, Vancouver, 1989.

Hans Uszkoreit. Strategies for adding control information to declarative grammars. In *29th Annual Meeting of the Association for Computational Linguistics*, Berkeley, 1991.

Gertjan van Noord. Reversible unification-based machine translation. In *Proceedings of the 13th International Conference on Computational Linguistics (COLING)*, Helsinki, 1990.

Gertjan van Noord. Towards uniform processing for constraint-based categorial grammars. In *Proceedings of the ACL Workshop on Reversible Grammar in Natural Language Processing*, Berkeley, 1991.

R. Wilensky, Y. Arens, and D. Chin. Talking to unix in english: An overview of uc. *Communications of the ACM*, pages 574 – 593, 1984.