HYPERBUG: A Scalable Natural Language Generation Approach

Martin Klarner
University Erlangen-Nuremberg
klarner@cs.fau.de

Abstract

A scalable natural language generation (NLG) system called HYPERBUG [1] embedded in an agent-based, multimodal dialog system is presented. To motivate this presentation, several scenarios (including a domain shift) are identified where scalability in dialog systems is really needed, and NLG is argued to be one way of easing this desired scalability. Therefore the novel approach to hybrid NLG in the HYPERBUG system is described and the scalability of its parts and resources is investigated. Concluding with a few remarks to discourse generation, we argue that NLG can both contribute to and benefit from scalability in dialog systems.

1 Introduction

Scalability in dialog systems is, of course, not only a matter of the natural language understanding (NLU) component, but also of the NLG part of the system.[2] We nevertheless see a lot of the effort spent in designing and implementing spoken dialog systems go into analysis of speech and language; generation is often left aside or squeezed in afterwards. Therefore an NLG component must fit into the existing dialog system framework and answer to the preset priorities in dialog system development.

We present a scalable NLG system embedded in an agent-based, multimodal dialog system. To this end, we first address common scenarios for scalability in dialog systems and describe the role of NLG in such systems before focusing on a classification of NLG systems in general and our hybrid linguistic realization approach in particular. After giving an overview of our NLG system we will be able to show how our notion of reversibility with respect to internal and external resources and the interaction and combination of shallow and deep NLG in HYPERBUG work towards and profit from scalability in our dialog system.

2 Scalability in Dialog Systems

Scalability for spoken dialog systems is needed in several situations, including the following scenarios:

1. Enlarging the domain **content** modifies and extends the thematic orientation of the domain.
2. Refining the domain **language** extends the linguistic coverage and expressibility of the domain.
3. Changing the **application domain** refers to usually both of the first two and can lead to completely new requirements for a dialog system and its parts.
4. Changing the **discourse domain** alters the discourse type within the same domain.

The common consequence of these four scenarios is their impact on both NLU and NLG: If scalability is not biased between these two parts, one cannot expect the system to be scalable as a whole. Especially in the situation of a domain shift, the degree of automated knowledge acquisition is an important issue to minimize costly (both in terms of time and money) manual efforts. If these are unavoidable for the NLU component, as it will often be the case in a real-world scenario, at least the NLG module must automatically benefit from them.

3 NLG in Dialog Systems

NLG itself and for its own can be seen as a way to improve the scalability of a dialog system. (Reiter, 1995) analyze the costs and benefits of NLG in comparison to other technologies and approaches, such as graphics, mail merging and human authoring, and argue for a hy-

[1] The acronym stands for **hy**brid, **p**ragmatically **e**mbedded **r**ealization with **b**ottom-**u**p **g**eneration.

[2] In fact, for a working spoken dialog system even more components have to be scalable, including the speech recognizer, the speech synthesizer and, most important of all, the dialog manager. But for now we concentrate on the opposition of NLU and NLG.

brid approach of combining NLG and templates in the IDAS system. Generally speaking, the application of NLG techniques in a real-world system must be justified with respect to linguistic and economic requirements.
If templates are used, a smart approach is needed in all scenarios mentioned in section 2 to avoid being forced to completely redesign at least the data part of the system output component. Nevertheless, fielded dialog systems often settle for a shallow generation module which relies on canned text and templates because of limited financial means and linguistic knowledge.

3.1 NLG and Dialog Management

In our spoken dialog system (Bücher et al., 2001), the dialog manager (DM) is responsible for the integration of user utterances into the discourse context. Moreover, the DM initiates system answers to user's questions and system error messages if a user's goal is unsatisfiable or a system task cannot be fulfilled. But these system utterances must be verbalized as well, i.e. translated from abstract semantic representations to natural language sentences. This task is not placed within the DM, but "outsourced" to a component with adequate linguistic competence, the NLG module. This way, the DM can be designed completely amodal, i.e. it does not need to have any linguistic knowledge. Moreover, we can see that scenario 4 in section 2 can be separated into a linguistic and a dialog part. The first part corresponds to scenario 2, the second is covered by the DM in our system; hence we may skip scenario 4 for the remainder of this paper.

3.2 Reversibility in dialog systems

Given a spoken dialog system in general and an NLG component in such a system in particular, we consider reversibility a central means to allow for scalability. Considering reversibility, we want to introduce two distinctions: We discriminate between reversibility of **algorithms** and reversibility of **data** on the one hand and between **static** (at developing time or at compile time) and **dynamic** reversibility (at runtime) on the other hand. In this terminology, reversibility of data means reusing existing system resources by the NLG component. We can classify NLG resources into two groups: The language analysis part contains the (syntactic and semantic) lexicon, the morphology component, and the grammar, while the DM part comprises the discourse memory, the domain model, and the user model.[3]

[3] For a different classification of knowledge resources for text planning, see (Maier, 1999).

4 Scalability in tactical generation

We focus on a special part of generation, the tactical generation or linguistic realization, according to the classification in (Reiter and Dale, 2000)[4]. We are able to do so mainly because, as mentioned in 3.1, the DM is responsible for content determination in our system. This leaves the realization task for the NLG component (besides some microplanning, which has to be performed as well).

4.1 A taxonomy of existing systems

In this section we will classify existing tactical generation systems into three groups and address problems with scalability in each one of them.
Shallow generation systems form the first group; e.g. COMRIS (Geldof, 2000). The approach taken there relies on canned text and templates, and the domain dependency of these constructs is inherent. Therefore, in scenario 3 of section **Fehler! Verweisquelle konnte nicht gefunden werden.**, once a domain shift is projected, the whole data part of the NLG component must be redesigned from scratch. In scenarios 1 and 2 the existing resources must be extended only, but even this can become a hard task if the existing template database is large enough.
Deep generation systems make up the second group, e. g. KPML (Bateman 1997); they often suffer from large overgenerating grammars and slow processing time. Also often well-founded linguistic knowledge is required to create and maintain the grammars needed. Their problems with scalability arise primarily in scenarios 1 and 2, when thematic or linguistic coverage must be increased.
The third group, **"modern" generation**[5] systems ideally avoid the shortcomings of both of the above mentioned classical approaches. We distinguish between three types here: NLG with **XSLT** (Wilcock, 2003), which is basically template-based generation from XML input; **stochastic** approaches like (Oh and Rudnicky, 2000), where the deep generation grammar is replaced by a stochastic language model, and **hybrid generation** approaches like D2S (Theune et al., 2000), which bridges the gap between NLG and speech synthesis by a prosody module.

[4] Dale and Reiter distinguish between linguistic and structure realization, the former corresponding to the content and the latter to the structural part of tactical generation. We find this distinction somewhat artificial, because the content must be already determined for realization, but want to use it anyway to further clarify the task carried out by our system.
[5] For these systems, the term "hybrid" is normally used in the literature, but we want to spare it for hybridization between shallow and deep generation; see 4.2.

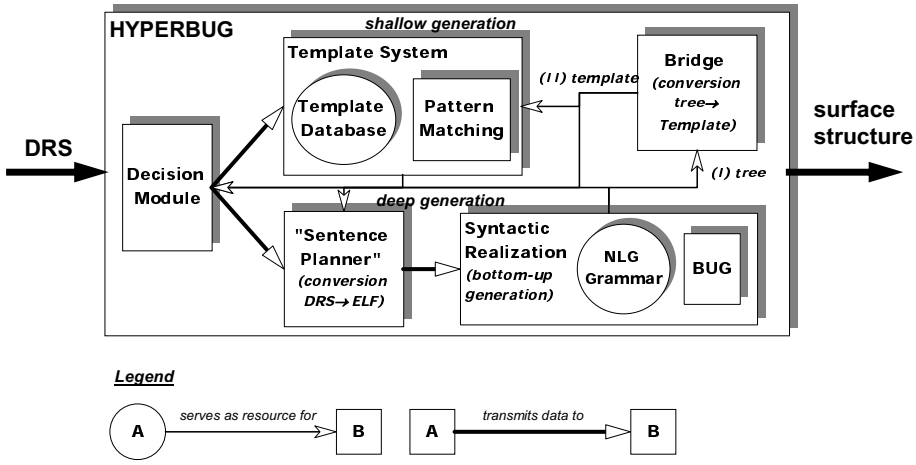

Figure 1: System core of HYPERBUG

4.2 Hybrid tactical generation

In our terminology, hybrid generation means the combination of shallow and deep generation in a single system; therefore, hybrid systems are a special case of the "modern" approaches, which were mentioned in the preceding section and do not necessarily contain any of the two "classical" approaches. For practical needs, we focus on how to combine deep (grammar-based) and shallow (template-based) generation techniques under the term hybrid NLG. We distinguish three types of such hybrid NLG systems:

- Type I: Shallow NLG with deep elements
- Type II: Deep NLG with shallow elements
- Type III: Concurring deep and shallow NLG

Here are two examples of existing systems to illustrate the classification just given: D2S fills slots in syntactic templates containing derivation trees and therefore can be classified as a type I system. (Wahlster, 2000) has separate shallow and deep generation modules resulting in a system of type III.[6]

5 The HYPERBUG Approach

5.1 System core functionality

Our approach to realization is to combine all three types of hybrid tactical generation mentioned in section 4.2 in a single system called HYPERBUG. The goals for its design and implementation were:

1. To **re-use** existing system resources originally designed for parsing
2. To generate templates **at runtime** rather than mere sentences
3. To dynamically reduce the workload on deep generation and gradually let shallow generation take over
4. To learn the domain-dependent part of the tactical generation task while the dialog is running and, ultimately, enable automatic adaptation to domain-shifts

Figure 1 shows the system core of HYPERBUG. As a shallow generation component, we implemented a powerful template engine with recursion and embedded deep generation parts, including a lexicon, a morphology component, inflection, and constituent aggregation, resulting in a system of type I in our classification.

For the deep generation branch, we decided to settle for a combination of a microplanning and a realization component: The first module, a "sentence planner", in essence converts the input discourse representation structure (DRS, Kamp and Reyle, 1993) which is provided by the DM into a different semantic representation, the extended logical form (ELF)[7]. This ELF structure serves as input for the second module, a modified and improved version of bottom-up generation (BUG, van Noord, 1990) in Java, using a unification-based grammar with feature structures. We also incorporated elements of shallow generation in our version of BUG: Surface text parts, e.g. proper nouns, may occur in the semantic input structure, the ELF. The precom-

[6] Type II was, though theoretically sound and possible, difficult to find in existing systems.

[7] The extensions allowed (as compared to a conventional LF) include syntactic functions like tense and mode, topicalization information and subordination clause type.

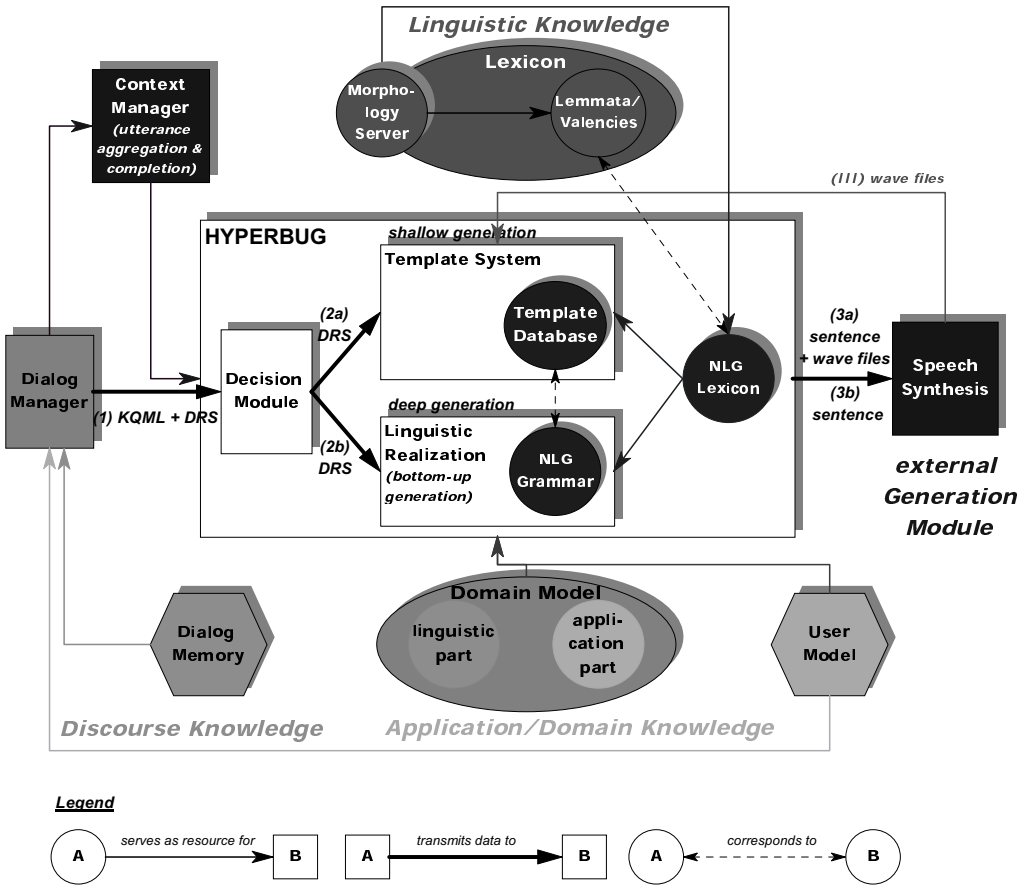

Figure 2: System resources of HYPERBUG

piled exception lexicon is searched first, whenever a proper noun occurs in the LF[8]. Thus, we get a type II system.

Links between shallow and deep generation are provided in two ways: At first, a decision module analyzes the input and invokes the appropriate realization branch, making HYPERBUG a system of type III. Stage 1 of this decision module is to use shallow generation as default and deep generation only as a fallback strategy. Stage 2 uses keyword spotting techniques in the input XML structure with XPATH and a lookup in an index table containing references to available canned text and templates. For stage 3, a planning procedure makes use of the speech act, the discourse situation and the user model to ensure that the most appropriate processing branch is selected; in extension of the approach for text planning presented in (Stent, 2001), we have applied Conversation Acts Theory (Poesio, 1994) to linguistic realization here.

Not only do we combine all three types of hybrid realization in our system, but we also interleave shallow and deep generation in another way: At last, after the complete utterance has been generated, a "bridge" between shallow and deep generation implements a feedback loop from BUG to the template system. This type of "bootstrapping"[9] is mainly responsible for the novel approach taken in HYPERBUG.

Figure 1 depicts, besides the system core, also the first two parts of the "bootstrapping" procedure developed in HYPERBUG.

[8] Proper nouns are indicated simply by capitalizing them.

[9] We use the term "bootstrapping" mainly as an analogy to classical bootstrapping procedures, hence the quotation marks.

(I) BUG passes the generated derivation tree to the bridge, where it is converted into a template.
(II) The bridge sends the template to the template system where it is stored for further use and to the decision module where a reference is saved in the index table. This way, the next dialog turn with similar semantic input can be realized faster using a template without having to invoke the deep generation branch again.

5.2 Resources and reversibility

Extending our description from the core functionality explained in the previous section, we now turn our attention to the resources used in HYPERBUG. Figure 2 gives an overview of the internal and external resources of HYPERBUG and their usage.
The linguistic knowledge is contained in the system lexicon and the morphology component. The former can be distinguished further into a valency lexicon containing case frames and a lemmata lexicon for domain exceptions[10]. These exceptions are compiled into the NLG lexicon used by both the shallow and deep generation branch, resulting in a static reversibility of data. The morphology component is separated from the core lexicon and implemented as a server which can both analyze and generate and is consequently used for parsing and generation. As a server with a uniform query interface, from the outside it looks like it were algorithmically reversible, but internally the algorithms for parsing and generation are implemented differently, resulting in a dynamic reversibility of data, because identical data are used and the processing direction is decided at runtime in the query sent to the server.
A separated module called context manager (CM) is responsible for sentence-spanning processing: It performs macro-aggregation[11] and completion of underspecified utterances from the DM. For this task, access to the dialog memory is required, which is performed indirectly via a DM request[12].
The templates are stored in a database which corresponds to the grammar in the deep generation branch. Unfortunately, the chunk grammar used for analysis is insufficient for generation[13], resulting in a lack of even static reversibility: In this case, the requirements for robust parsing differ too much from the ones for variable NLG.

[10] The domain independent lexicon entries are derived from WORDNET synsets.
[11] By this term we mean aggregation on the sentence level, as opposed to micro-aggregation which occurs between constituents.
[12] In the current implementation of our system, the dialog memory is only accessible this way.
[13] The chunk grammar does, of course, not contain phrase rules up to the sentence level.

The domain model is divided into a linguistic and an application-specific part. Only the linguistic part is used in HYPERBUG, mainly for substitution of synonyms, hyperonyms and hyponyms to enlarge variability in lexical choice, a part of microplanning also handled by our approach to NLG.
The processing steps in the system are as follows:
(1) The dialog manager sends to HYPERBUG a KQML[14] message with the semantic representation of the system utterance to be realized (a DRS encoded in an XML structure), enriched with pragmatic information such as sender, KQML speech act, and pragmatic dialog act.
(2) The decision module determines, based on linguistic and pragmatic information (such as the speech act and the user model), whether shallow or deep generation is (more) appropriate to process the message and feeds it (a) to the template system or (b) to the deep generation branch (or to both of them, as a fall-back strategy).
(3 a) The template system passes the generated surface structure with additional information[15] to the speech synthesis agent[16], including wave files of utterance parts which were already synthesized before.
(3 b) BUG passes the generated surface structure with additional information to the speech synthesizer.
There is also a third bootstrapping aspect depicted in Figure 2:
(III) The synthesizer agent returns the synthesized wave files to the template system to enhance the stored templates. This way, the synthesizer does not have to process identical utterance parts more than once, thus increasing the efficiency in synthesis and the real-time-capacity of the system. [17]

5.3 Scalability in HYPERBUG

After giving an overview of our NLG system, we will now address scalability issues in its parts and resources.
Template system. Templates in HYPERBUG are at least algorithmically unproblematic: The pattern matching algorithm is linear complex with the number of entries. But normally, template systems are poorly maintainable and need to be rewritten from scratch after a domain shift. We try to overcome this difficulty by isolating a considerably large domain-independent part, such as metadialog (ambiguity, coherence state, and plan/action

[14] The language KQML is currently used for agent communication in our system, but we are in the process of transition to FIPA-ACL.
[15] sentence accent to influence prosody generation in the synthesizer and deixis to synchronize textual output with the avatar in our multimodal system
[16] in essence a wrapper agent around the open-source synthesizer MBROLA
[17] System profiling has shown a considerable amount of processing time of the generation component going into synthesis with MBROLA.

state), greeting, and default messages[18], avoiding much of the effort needed in scenario 3. Scenarios 1 and 2, on the other hand, are treated in our template system by using modularity via inclusion: The templates are recursive so that we can easily extend and refine the existing database. All in all, we can state that relatively few new entries are required for our template system in all three scenarios 1-3.

Lexicon and morphology component. The lexicon and the morphology component are also algorithmically unproblematic, as they are linear complex with the number of entries. Furthermore, we were able to re-use a large part of the existing system resources initially designed for parsing (i.e. the exception lexicon and the morphology server). We can summarize that, for these two components, NLG automatically grows with NLU, and that no NLG-specific effort is required in the first three scenarios mentioned above.

Grammar. For the deep generation branch, the grammar can lead to algorithmic problems: The algorithm has exponential complexity, but only in the number of categories within the rules. However, this number is finite with a low upper bound. The algorithm has linear complexity in the number of words, just like the underlying lexicon does. Disjunctive unification can cause problems, but not if it is restricted to simple features, as it is in our system. Anyway, a large part of the rules can be re-used in all three scenarios[19], but a proper grammar organization is required for the inevitable manual maintenance.

Hybrid approach. The central argument for the scalability of HYPERBUG, however, lies in its special hybrid design: The decision module before and the bridge after the two generation branches constitute the bootstrapping approach which continually improves the system performance in terms of efficiency and linguistic coverage at runtime (useful for scenarios 1 and 2) and enables automatic adaptation to domain shifts (scenario 3).

Speech synthesis. HYPERBUG has a built-in feature enabling intrasentential multilingual speech synthesis[20], rendering the system scalable in terms of language changes. The second aspect of scalability within speech synthesis is the other "bridge" between this external module and the template system which gradually improves the system response time in all three scenarios.

Pragmatic resources. The pragmatic resources comprise the dialog memory, the domain model and the user model. For the domain model, it is possible that new NLG-specific entries are required in the scenarios above. But these entries are not a critical factor in terms of scalability. The complexity of the discourse memory mainly depends on the dialog length. This is a largely domain-independent factor and not affected by our scenarios. All we can say about the user model is that as a primarily non-linguistic resource it is not in the focus of NLG. If it needs to be enriched or refined in scenario 1 and 2 or even redefined in scenario 3, its usage in our NLG system retains its complexity.

We conclude that the external pragmatic resources can be extended and re-used without any impact on HYPERBUG and do therefore not influence the scalability of our NLG component.

Discourse generation. Finally, we want to briefly address some aspects of discourse generation as a way of scalability in terms of linguistic expressibility. Deictic expressions are currently hard-coded in special templates, because they are highly domain-dependent. In our multimodal system, they must be synchronized with the other output modalities, such as the avatar performing deictic gestures. The expected place for anaphora generation in our system is the CM. As a pragmatic resource, the dialog memory is used for this task via the DM. Pronominal references are enabled by the CM which checks for appropriate discourse referents to be pronominalized; they are executed by the sentence planner which substitutes nouns by matching pronouns in the LF. The conditions for appropriate (i.e. unambiguous) anaphora and replacements of nouns by pronouns are not easy to meet and check. Our current idea involves a generate-and-test approach, i.e. we want to tentatively generate an anaphor or a pronoun and use the analysis part of our dialog system to determine whether they are ambiguous or not.

6 Related Work

Generally, the method known as explanation-based generalization from machine learning is comparable to the bootstrapping approach described here; but normally learning is achieved by offline training.

In (Neumann, 1997) a training phase with an appropriate corpus is needed, while we perform generation at runtime without such a corpus. Furthermore, Neumann extracts complex subgrammars; we generate annotated surface sentences instead, which are less expressible, but faster to instantiate. And finally, Neumann performs a static template choice as opposed to our runtime decision module which can opt for deep generation based on pragmatic constraints, even if a semantically appropriate template is already available.

(Corston-Oliver. 2002) has a machine learning approach for realization similar to (Neumann, 1997): Transformation rules are learned offline from examples in a corpus. Again, a separate training phase is needed beforehand.

[18] i.e. ok and error messages, the latter tending to be rather domain-specific, though

[19] This is, of course, the inherent advantage of deep over shallow NLG.

[20] Basically, lexical information about the language of a proper noun (e.g. a person's name) is included in the output to the synthesizer which uses this information to switch between target languages, even within a single sentence.

(Scott, 1998) can be seen as offline interface generation using a GUI and therefore as a manual version of the bootstrapping approach described here, but her system is used for content determination, not for realization.

7 Conclusion and Further Work

We have presented a hybrid NLG system that can both contribute to and benefit from the scalability in its embedding multimodal dialog system. Various scenarios requiring a scalable NLG system where identified and applied to our system components and resources in order to analyze their scalability.

A prototype of the system is implemented and used in several different domains, namely home A/V and car audio management (Bücher, 2001), B2B e-procurement (Kießling, 2001), and model train controlling (Huber and Ludwig, 2002), but we need further evaluation of the requirements for a domain shift and of the user acceptance to improve the quality of our output language and speech.

What remains to do on the implementation side? Technically, we still lack a fully implemented "sentence planner" with in-depth analysis of the semantic input structure (which is only processed in a shallow manner by now), and a separation of pure canned text from templates for efficiency. Also, the interface to the linguistic part of the domain model which is represented in description logics must be implemented using an appropriate inference machine. Conceptually, we want to broaden the bridge between shallow and deep generation, refine the specification of stage 3 in the decision module, and work out a way to access the user model directly (currently, it is accessed indirectly via the DM, just like the discourse memory).

Acknowledgements

The author wants to thank Bernd Ludwig and Peter Reiss for fruitful discussions and interesting ideas contributing to the research reported in this paper.

References

John Bateman. 1997. *Enabling technology for multilingual natural language generation: the KPML development environment.* Journ. Natural Language Engineering 3 (1):15-55.

Kerstin Bücher et al. 2001. *Discourse and Application Modeling for Dialogue Systems.* Proc. KI-2001 Workshop on Applications of Description Logics.

Simon Corston-Oliver. 2002. *An overview of Amalgam: A machine-learned generation module.* Proc. Int. Natural Language Generation Conference, New York:33-40.

Sabine Geldof. 2000. *Context-sensitivity in advisory text generation.* PhD Thesis, University of Antwerp.

Alexander Huber and Bernd Ludwig. 2002. *A Natural Language Multi-Agent System for Controlling Model Trains.* Proc. AI, Simulation, and Planning in High Autonomy Systems (AIS-2002):145-149.

Hans Kamp and Ulrich Reyle. 1993. *From Discourse To Logic.* Kluwer, Boston/Dordrecht/London.

Werner Kießling et al. 2001. *Design and Implementation of COSIMA - A Smart and Speaking E-Sales Assistant.* Proc. 3rd International Workshop on Advanced Issues of E-Commerce and Web-Based Information Systems (WECWIS '01):21-30.

Günther Neumann. 1997. *Applying Explanation-based Learning to Control and Speeding-up Natural Language Generation.* Proc. ACL/EACL-97.

Gertjan van Noord. 1990. *An Overview of Head-Driven Bottom-Up Generation.* In: Robert Dale et al. *Current Research in Natural Language Generation.* Springer, Berlin/Heidelberg/New York.

Alice Oh and Alexander Rudnicky: *Stochastic language generation for spoken dialogue systems.* Proc. ANLP/NAACL 2000 Workshop on Conversational Systems: 27-32.

Ehud Reiter. 1995. *NLG vs. Templates.* Proc. 5th European Workshop on Natural Language Generation (EWNLG-1995).

Ehud Reiter and Robert Dale. 2000. *Bulding Natural Language Generation Systems.* Cambridge University Press, Cambridge, UK.

Donia Scott et al. 1998. *Generation as a Solution to its Own Problem.* Proc. 9th Int. Workshop on Natural Language Generation (INLG-98).

Amanda J. Stent. 2001. *Dialogue Systems as Conversational Partners: Applying Conversation Acts Theory to Natural Language Generation for Task-Oriented Mixed-Initiative Spoken Dialogue.* PhD Thesis, Rochester, NJ.

Marie Theune et al. 2000. *From Data to Speech: A General Approach.* Natural Language Engineering 7(1):47-86.

Wolfgang Wahlster. 2000. *Verbmobil: Foundations of Speech-to-Speech Translation.* Springer, Berlin/Heidelberg/New York.

Graham Wilcock. 2003. *Generating Responses and Explanations from RDF/XML and DAML+OIL.* Proc. IJCAI-2003:58-63.