

An Extended Architecture for Robust Generation*

Tilman Becker, Anne Kilger, Patrice Lopez, Peter Poller

DFKI GmbH
Stuhlsatzenhausweg 3
D-66123 Saarbrücken, Germany
{becker,kilger,lopez,poller}@dfki.de

Abstract

Based on our experiences in VERBMOBIL, a large scale speech-to-speech translation system, we identify two types of problems that a generation component must address in a realistic implementation and present relevant examples. As an extension to the architecture of a translation system, we present a module for robustness preprocessing on the interface between translation and generation.

1 Introduction

Based on our experiences with VERBMOBIL, a large scale speech-to-speech translation system, we identify two types of problems that a generation component must address in a comprehensive implementation. Besides general *task-inherent* problems like, e.g., the processing of spontaneous speech input, the translation step itself, and real-time processing, we found that an implementation of such a large scale system additionally exhibits *technical problems* that are caused by various faults in the steps prior to generation.

The task of VERBMOBIL is the multi-lingual (German, English, Japanese) speaker-independent translation of spontaneous speech input that enables users to converse about the scheduling of a business appointment including travel, accommodation, and leisure time planning in a multi-lingual dialogue. The system covers 10,000 words in each language with the corresponding knowledge bases (grammars, translation rules, etc.). In contrast to a text translation system, the processing of spontaneous speech requires extended functionalities in almost every module because the system has to be able to deal with, e.g., ill-formed and disfluent (hesitations, repetitions, repairs) speech input. In a dialogue system, there is also an apparently simple but very strong constraint on the system to achieve its task: For each user input the system has to produce a translation result.

Due to the high complexity of this task, the system is subdivided into 24 separate subtasks (implemented modules).

For the translation step the system contains four different parallel translation “tracks” consisting of three “shallow” (case based, statistical, and dialogue-act based (Reithinger, 1999)) and one “deep” translation track (see figure 1) for each language. The individual translation results are partly associated with confidence values reflecting their quality and then sent to a special selection component to choose the most appropriate one. Our practical experience shows that there are cases in which the input to the generation component is impossible to process. Here the shallow translation paths serve as a fall-back in order to fulfill the strong necessity of a translation result as far as possible.

Although some parts of the analysis task (e.g., resolving scopal ambiguities) can actually be left unsolved when they are not necessary for the translation task, in general, problems in some module result in an accumulated inadequacy of the final translation.

Since the translation task is distributed to a set of cooperating modules, there is a choice of solving the task inherent and technical problems either locally inside the individual modules or handing them to problem specific correction modules. We found that robustness must be included in every module. For the architecture of the generation module, we have devised a submodule for robustness that preprocesses the input data. This proved an elegant and promising extension to achieve the required local module robustness without touching the core generation module directly. A similar module also exists for analysis (see ‘Robust Semantics’ in figure 1), (Worm, 1998).

In this paper, we focus on the generation component of our system. Besides the general robustness requirements, the mentioned inadequacy accumulation reaches its maximum since generation is positioned at the end of the translation process. In the following sections, we show how the strong robustness requirement influenced the architecture of our

* The research within VERBMOBIL presented here is funded by the German Ministry of Research and Technology under grant 01IV101K/1.

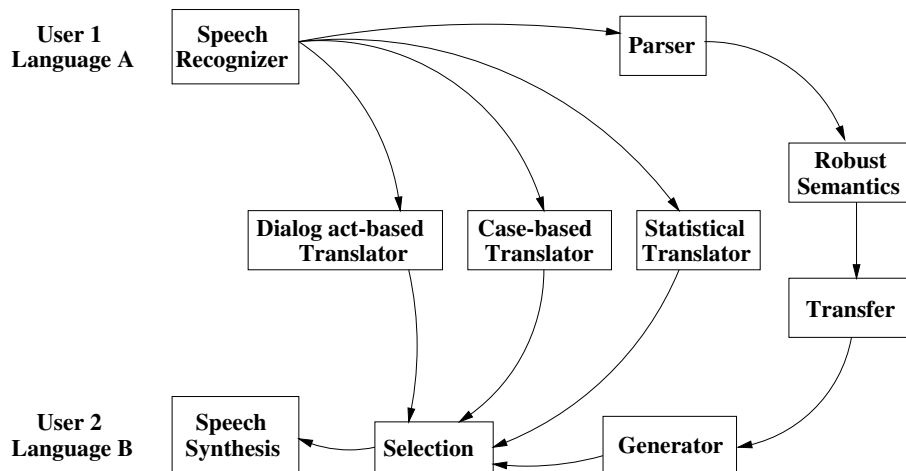


Figure 1: Simplified system architecture of the speech-to-speech translation system VERBMOBIL.

generation module. We classify the above mentioned problems from the point of view of generation and present our solutions to these problems, mainly under the aspect of *robust* generation with problematic input.

2 Task-inherent and Technical Problems

The problems for generation that arise in a speech-to-speech translation system fall into two main classes: as in any large-scale system, there will be software-engineering problems which we will call *technical* problems and there are *task-inherent* problems that are particular to the translation task and the highly variable input in spontaneous speech. Since it is impossible to draw a clear line between technical and task-inherent problems, we will present a short classification and then go into more detail without much discussion whether a certain problem should be viewed as technical or task-inherent.

One would hope to be able to eliminate technical problems completely. However, in a large system, where development is distributed over many modules (implemented at different sites), some robustness against certain technical problems can become a necessity, as our experiences have shown. This is even more important during the development phase—which a research system never leaves. Most technical problems have to do with violations of the interface definitions. This ranges from simple things such as using unknown predicates in the semantic representation to complex constructions that cannot be generated (the generation gap). We actually regard the latter as a task-inherent problem.

Secondly, the task-inherent problems can be divided into problems that are caused by (i) spon-

aneous speech input and (ii) insufficiencies in the analysis and translation steps.

2.1 Robustness in Analysis

The problems in (i) are quite varied and many cases are dealt with in analysis (and translation), some cases are dealt with in our robustness preprocessing submodule, a few in the classical submodules of generation. For example, there is a separate module on the level of speech recognition which deals with hesitations and self-corrections. Phenomena like ellipsis, phrasal and other incomplete utterances are handled by analysis, so generation must be able to deal with the corresponding semantic representations too. Agreement errors are handled (i.e., corrected) in analysis. But some serious syntactic errors cannot be corrected. However, at least the maximal analyzable segments are determined so that ungrammatical utterances are translated as sequences of several meaningful segments.

2.2 Robustness in Generation

The problems in (ii) are caused by an accumulation of problems which result in (semantic) input to the generator that cannot be processed. Robustness in our system concentrates on this type of problem which is and should be handled as a separate step between analysis/transfer and generation. (See the discussion of the architecture in section 3.)

The list below contains some examples that are picked up again in section 4.

- Problems with the *structure* of the semantic representation:
 - unconnected subgraphs
 - multiple predicates referring to the same object

- omission of obligatory arguments
- Problems with the *content* of the semantic representation:
 - contradicting information
 - missing information (e.g. agreement information)

3 Architecture

As described in section 1, the deep processing in VERBMOBIL is based on a pipeline of modules which use a unique interface language (VIT¹) that incorporates a semantic representation. Since this semantic representation is obviously grammar-independent and could reflect the effects of spoken, spontaneous language, we have no guarantee that the grammar covers the semantic representation given by the transfer module. Consequently we have chosen to extend the classical generation architecture with a new module dedicated to robust preprocessing. We first present our classical generator architecture (see also (Becker et al., 1998; Becker et al., 2000)) in terms of the RAGS architecture and then discuss its extension to the task-inherent problems.

The RAGS architecture (Cahill et al., 1999) is a reference architecture for natural language generation systems. Reflecting the common parts of natural language generation systems, this proposal aims to provide a standard architecture allowing the identification of some important generation subtasks and resources. By presenting our system in the light of the RAGS specifications, our goal is to propose general solutions that could be used by other researchers who need to extend their own generation architecture to similar tasks.

While the *macro-planning* task is important and mandatory in text generation, it is limited in dialogue translation. Most of the related problems, for instance the sentence segmentation and the pronoun choices, have been solved by the user in the source language. Considering the RAGS architecture, conceptual and rhetorical levels of representation are also outside the scope of our system. Our architecture consists of four main modules (see figure 2). For an easy adaptation to other domains and languages, we have emphasized an organization based on a general kernel system and the declarativity of knowledge sources (Becker et al., 1998). All but the first modules are captured by the RAGS architecture. However, the first module is dedicated solely to robustness in the specific speech-to-speech translation task and will be presented and discussed last in this section. It can easily be added to a RAGS-like system whose *whiteboard* is perfectly suited for

¹Verbmobil Interface Term, (Bos et al., 1996; Dorna, 1996)

the transformations that the robustness preprocessing module performs.

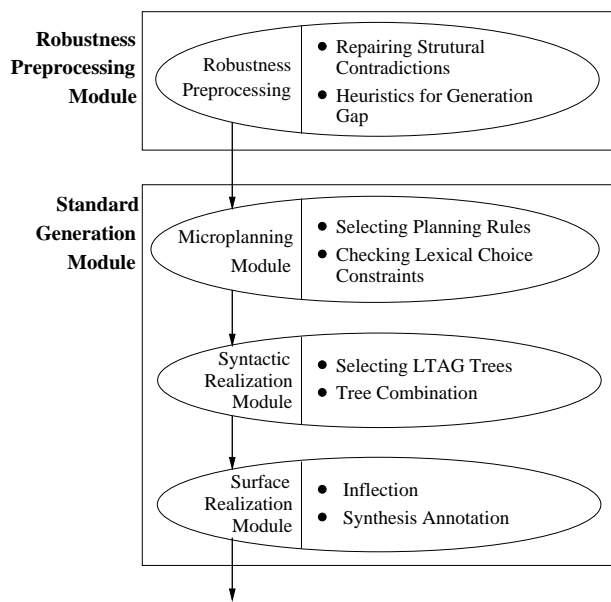


Figure 2: An extended generation system architecture

Microplanning Module At the level of sentence generation, the quality of the planning process depends on the interdependencies between conceptual semantics, predicative semantics and syntax. A particular lexical choice can imply constraints on other lexical items. The role of the microplanner is to realize lexical choices in a way that a syntactic realization is possible and costly backtracking is prevented.

The microplanning task can be viewed as a constraint solving problem and implemented using an adapted constraint solving mechanism in order to achieve efficiency, flexibility, and declarativity of knowledge. The microplanner produces a dependency tree representation indicating for each node a set of syntactical constraints to be fulfilled by the corresponding lexical syntactic units (predicate, tense, aspect, mood, etc.).

Syntactic Realization Module This module is in charge of the concrete syntax generation. The processing is based on a fully lexicalized Tree-Adjoining Grammar derived from the HPSG grammar used in the deep-processing parser module (Kasper et al., 1995; Becker, 1998).

Surface Realization Module The syntactic realization module produces a derived tree from which the output string is extracted. The morphological features in preterminal nodes are used for inflection. The surface string is also annotated by syntactic information (phrase boundary, aspect, sentence mood)

that are exploited by the speech synthesis module.

Robustness Preprocessing Module We have described three modules corresponding to classical tasks of generation systems and pointed out at the beginning of this section the necessity for robustness. Where can we integrate the required robustness in such a generation architecture? One approach could be the relaxation of constraints during the syntactic realization (relaxing word order or/and dependency relations). One can argue against this approach that:

- There is no straightforward way to limit the relaxation of syntactic constraints only to the robustness problematic structures.
- We must be sure that the microplanning module can deal with problematic semantic input.

These points suggest to check and repair the inconsistencies of the semantic representation as early as possible, i.e., before sentence microplanning. Moreover we show in the next section that most of the problems presented in section 2 can be identified based on the microplanner input.

We now present more concretely the robust preprocessing module.

4 Robustness

In this section we describe the types of problems defined in section 2 using examples from our system and discuss how our module is made robust enough to handle a lot of these problems.

Before the semantic representation is handed to microplanning, the *robustness preprocessing* module of the generator checks the input, inspecting its parts for known problems. For each problem found, the preprocessor lowers a *confidence value* for the generation output which measures the reliability of our result. In a number of cases, we use *heuristics* to fix problems, aiming at improved output.

As discussed in section 2, problems in the input to the generator can be technical or task-inherent. Technical problems manifest themselves as faults wrt. the interface language definition, whereas the task-inherent problems concern mismatches between a specific semantic expression and the coverage of the natural language grammar used in the generator. These mismatches are known as the generation gap (Meteer, 1990).

4.1 Declarativity

In our implementation, the robustness module is partly integrated into the constraint solving approach of the microplanning module. Using the constraint solving approach allows for a strict separation of algorithms (i.e., some constraint solving algorithm) and *declarative* knowledge sources. On this level, the rules (constraints) for robustness can be

clearly separated from the microplanning rules, justifying our presentation of robustness as a separate module.

4.2 Conforming to the Interface Language Definition

The definition of the interface language² comprises only its syntax and some semantic constraints. There is an implementation of expressions in the interface language as an abstract data type which can at least check syntactic conformity (Dorna, 1996). But we also have to deal with semantic faults.

The first example illuminating robust preprocessing is on the connectedness of the semantic input graph. Our interface language describes an interface term to contain a connected semantic graph plus an index pointing to the root of the graph. Two types of problems can occur according to this definition:

Disconnectedness of the Graph: The *robustness preprocessor* checks whether the input graph is in fact connected. If there are several disconnected parts, a distinct generation call is made for each subgraph. In the end, all sub-results are connected to produce a global result. We are currently working on a better heuristic to order the sub-results, taking into account information about the word order in the source language.

Wrong Index: The robustness preprocessor tests whether the index points to the root of the graph or one of the subgraphs. For each subgraph without an adequate index, we compute a local root pointer which is used for further processing. This turned out to be an easy and reliable heuristic, leading to good results.

There are several types of technical problems which cannot be repaired well. Minimally, these cases are detected, warning messages are produced, and the confidence value is lowered. We apply heuristics where possible. Examples are *uniqueness of labels* (every semantic predicate must have a unique identifier), the use of *undefined predicate names*, and *contradicting information* (e.g., the use of a DEFINITE and an INDEFINITE quantifier for the same object). In the case of *incorrect predicate classes*, i.e., where a predicate is used with an undefined argument frame, only those parts of the input are handled which are analyzed as correct.

4.3 Falling into the Generation Gap

The robustness preprocessor even does more than checking for structural contradictions between input and interface language. Based on analyses of

²A further complication in a research system like ours stems from the fact that the interface language itself is developed, i.e., changed over time.

a large amount of test-suites it is fed with some heuristics which help to bridge the generation gap that reflects the unpredictability whether a specific semantic structure can be mapped to an acceptable utterance in the target language. Some examples of heuristics used in our system are as follows:

Conflicting Information: Often it is inconsistent to allow several predicates to include the same depending structure in their argument frames, e.g., two predicates describing different prepositions should not point to the same entity. We have to pick one possibility heuristically.

Gaps in Generation Knowledge: There are input configurations that have no reflection within the generator’s knowledge bases, e.g., the DISCOURSE predicate defining a sequence of otherwise unrelated parts of the input. The robustness preprocessor removes this predicate, thereby subdividing the connected graph into several unconnected ones and continuing as for disconnected graphs described above.

Other examples for generation constraints that can conflict with the input are the occurrence of some specific cyclic subparts of graphs, self-referring predicates, and chains of predicates which are not realizable in generation.

Robustness inside the Microplanner and the Syntactic Generator additionally helps to get rid of some generation gap problems:

Contradictions to Generation Constraints:

The knowledge bases of the generator (microplanning rules, grammar and lexicon) describe constraints on the structure of the output utterance that might conflict with the input. A common problem occurring in our system is the occurrence of subordinating predicates with empty obligatory arguments. Here the microplanner relaxes the constraint for argument completeness and hands over a structure to the syntactic generator that does not fulfill all syntactic constraints or contains elliptical arguments. In these cases, the grammar constraints for obligatory arguments are relaxed in the syntactic generator and elliptical arguments are allowed beyond the constraints of the grammar. The result is often output that reflects the spontaneous speech input which we accept for the sake of robustness.

Missing attributes: Often there are obligatory attributes for the semantic predicates missing in the input, e.g., statements about the directionality of prepositions, agreement information, etc. The generator uses heuristics to choose a value for its own.

Contradictions on the Semantic Level: Some attributes may lead to conflicts during generation, e.g., if a pronoun is given as SORT≠HUMAN and TYPE=SPEAKER. The generator uses a heuristics to set the value of SORT in this case.

Solving Part of the Analysis Task: Sometimes the input to the generator is underspecified in a way that it can be improved easily by using simple heuristics to “continue analysis.” A common example in our system is an input expression like “*on the third*” which often is analyzed as (ABSTR_NOM ∧ ORD(3)), i.e., an elliptical noun with ordinal number 3. We add the sort TIME_DOFM³ to the attributes of ABSTR_NOM so that, e.g., a semantic relation TEMPORAL_OR_LOCAL is correctly mapped to the German preposition “*an*.”

4.4 How much robustness?

There is a limit to the power of heuristics that we have determined using a large corpus of test data. Some examples for possible pitfalls:

- When realizing “empty nominals” ABSTR_NOM as elliptical nouns, guessing the gender can cause problems: “*Thursday is my free day t_i* ” as FREE ∧ DAY ∧ ABSTR_NOM (with a reading as in “*day job*”) might result in “**Donnerstag ist mein freies Tag t_i* .”
- Conflicts between sort and gender of a pronoun might be resolved incorrectly: “*Es (English: ‘it’) trifft sich ganz hervorragend*” with PRON (GENDER=NTR, SORT=HUMAN) should not be translated as “*#He is really great*.”

Although the boundary beyond which deep translation cannot be achieved even with heuristics is somewhat arbitrary, the big advantage of deep processing lies in the fact that the system ‘knows’ its boundaries and actually fails when a certain level of quality cannot be guaranteed. As discussed in section 1, in a dialogue system, a bad translation might still be better than none at all, so one of the shallow modules can be selected when deep processing fails.

5 Related Work and Conclusions

VERBMOBIL also contains a component that automatically generates dialogue scripts and result summaries of the dialogues in all target languages (Alexandersson and Poller, 1998; Alexandersson and Poller, 2000). This component uses the generation modules of VERBMOBIL for sentence generation as well as the translation system itself to achieve multilinguality. To some extent, this task also benefits

³DOFM: day of the month. Note that in this paper, the presentation of the semantic representation language is highly abstracted from the actual interface language.

from the task inherent robustness features of the overall system and its modules which we described in this paper.

Our problem classification also shows up in other generation systems. There is a multi-lingual generation project (Uchida et al., 1999) that utilizes an interlingua-based semantic representation to generate web-documents in different output languages from one common representation. Although technical problems are less prominent, the task-inherent problems are almost the same. Again, the generator has to be able to deal with, e.g., disconnected or contradicting input graphs.

References

- Jan Alexandersson and Peter Poller. 1998. Toward multilingual protocol generation for spontaneous speech dialogues. In *Proceedings of the Ninth International Workshop on Natural Language Generation*, Niagara-on-the-Lake, Ontario, Canada, August.
- Jan Alexandersson and Peter Poller. 2000. Multilingual summary generation in a speech-to-speech translation system for multilingual negotiation dialogues. In *Proceedings of INLG 2000*, Mitzpe Ramon, Israel, June.
- T. Becker, W. Finkler, A. Kilger, and P. Poller. 1998. An efficient kernel for multilingual generation in speech-to-speech dialogue translation. In *Proceedings of COLING/ACL-98*, Montreal, Quebec, Canada.
- Tilman Becker, Anne Kilger, Patrice Lopez, and Peter Poller. 2000. Multilingual generation for translation in speech-to-speech dialogues and its realization in verbmobil. In *Proceedings of ECAI 2000*, Berlin, Germany, August.
- Tilman Becker. 1998. Fully lexicalized head-driven syntactic generation. In *Proceedings of the Ninth International Workshop on Natural Language Generation*, Niagara-on-the-Lake, Ontario, Canada, August.
- Johan Bos, Björn Gambäck, Christian Lieske, Yoshiki Mori, Manfred Pinkal, and Karsten Worm. 1996. Compositional semantics in verbmobil. In *Proceedings of Coling '96*, Copenhagen, Denmark.
- Lynne Cahill, Christy Doran, Roger Evans, Chris Mellish, Daniel Paiva, Mike Reape, Donia Scott, and Neil Tipper. 1999. Towards a Reference Architecture for Natural Language Generation Systems. Technical Report ITRI-99-14, Information Technology Research Institute (ITRI), University of Brighton.
- Michael Dorna. 1996. The adt package for the verbmobil interface term. Verbmobil-Report 104, University Stuttgart, April.
- R. Kasper, B. Kiefer, K. Netter, and K. Vijay-Shanker. 1995. Compilation of hpsg to tag. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 92–99, Cambridge, Mass.
- M.W. Meteer. 1990. *The “Generation Gap” – The Problem of Expressibility in Text Planning*. Ph.D. thesis, Amherst, MA. BBN Report No. 7347.
- Norbert Reithinger. 1999. Robust information extraction in a speech translation system. In *Proceedings of EuroSpeech-99*, pages 2427–2430.
- Hiroshi Uchida, Meiyong Zhu, and Tarcisio Della Senta. 1999. UNL. IAS, United Nations University, Tokyo, Japan, November.
- Karsten Worm. 1998. A model for robust processing of spontaneous speech by integrating viable fragments. In *Proceedings of COLING-ACL '98*, Montreal, Canada.