Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

Document
D-97-06

DFKI Workshop on Natural Language Generation

Tilman Becker, Stephan Busemann, Wolfgang Finkler (eds.)

September 1997

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH

Postfach 20 80
67608 Kaiserslautern, FRG
Tel.: + 49 (631) 205-3211
Fax: + 49 (631) 205-3210

Stuhlsatzenhausweg 3
66123 Saarbrücken, FRG
Tel.: + 49 (681) 302-5252
Fax: + 49 (681) 302-5341

Deutsches Forschungszentrum für Künstliche Intelligenz

The German Research Center for Artificial Intelligence (Deutsches Forschungszentrum für Künstliche Intelligenz, DFKI) with sites in Kaiserslautern and Saarbrücken is a non-profit organization which was founded in 1988. The shareholder companies are Atlas Elektronik, Daimler-Benz, Fraunhofer Gesellschaft, GMD, IBM, Insiders, Mannesmann-Kienzle, Sema Group, Siemens and Siemens-Nixdorf. Research projects conducted at the DFKI are funded by the German Ministry of Education, Science, Research and Technology, by the shareholder companies, or by other industrial contracts.

The DFKI conducts application-oriented basic research in the field of artificial intelligence and other related subfields of computer science. The overall goal is to construct systems with technical knowledge and common sense which - by using AI methods - implement a problem solution for a selected application area. Currently, there are the following research areas at the DFKI:

- ☐ Intelligent Engineering Systems
- ☐ Intelligent User Interfaces
- ☐ Computer Linguistics
- ☐ Programming Systems
- ☐ Deduction and Multiagent Systems
- ☐ Document Analysis and Office Automation.

The DFKI strives at making its research results available to the scientific community. There exist many contacts to domestic and foreign research institutions, both in academy and industry. The DFKI hosts technology transfer workshops for shareholders and other interested groups in order to inform about the current state of research.

From its beginning, the DFKI has provided an attractive working environment for AI researchers from Germany and from all over the world. The goal is to have a staff of about 100 researchers at the end of the building-up phase.

Dr. Dr. D. Ruland
Director

DFKI Workshop
on
Natural Language Generation

Tilman Becker, Stephan Busemann, Wolfgang Finkler (eds.)

DFKI-D-97-06

This work has been supported by a grant from The Federal Ministry of Education, Science, Research and Technology (FKZ ITWM-01 IV 701 V0).

© Deutsches Forschungszentrum für Künstliche Intelligenz 1997
This work may not be copied or reproduced in whole of part for any commercial purpose. Permission to copy in whole or part without payment of fee is granted for nonprofit educational and research purposes provided that all such whole or partial copies include the following: a notice that such copying is by permission of the Deutsche Forschungszentrum für Künstliche Intelligenz, Kaiserslautern, Federal Republic of Germany; an acknowledgement of the authors and individual contributors to the work; all applicable portions of this copyright notice. Copying, reproducing, or republishing for any other purpose shall require a licence with payment of fee to Deutsches Forschungszentrum für Künstliche Intelligenz.
ISSN 0946-008X

DFKI Workshop on Natural Language Generation

Tilman Becker, Stephan Busemann, Wolfgang Finkler (eds.)

April 23, 1997

Abstract

On the Saarbrücken campus sites as well as at DFKI, many research activities are pursued in the field of Natural Language Generation (NLG). We felt that too little is known about the total of these activities and decided to organize a workshop in order to share ideas and promote the results.

This DFKI workshop brought together local researchers working on NLG. Several papers are co-authored by international researchers. Although not all NLG activities are covered in the present document, the papers reviewed for this workshop clearly demonstrate that Saarbrücken counts among the important NLG sites in the world.

Contents

Preface

On the Saarbrücken campus sites as well as at DFKI, many research activities are pursued in the field of Natural Language Generation (NLG). We felt that too little is known about the total of these activities and decided to organize a workshop in order to share ideas and promote the results.

This DFKI workshop took place on April 23, 1997. It brought together local researchers working at different DFKI labs, the Computer Science Department, and the "Institut für Angewandte Sprachwissenschaft sowie Übersetzen und Dolmetschen" at the University of the Saarland. Several papers are co-authored by international researchers. The Call for Papers included NLG and related fields, but eventually all contributions concentrated on NLG proper. The workshop produced a valuable exchange of results on completed and ongoing work. Although not all NLG activities are covered in the present document, the papers reviewed for this workshop clearly demonstrate that Saarbrücken counts among the important NLG sites in the world.

During the final discussion it was agreed to strive for a joint public presentation of the Saarbrücken NLG activities by providing information on a central WWW page. This page links up to the different individual system or project pages. It is maintained by DFKI, while the maintainance of the links remains with their respective owners.

We hope that the NLG page, being updated on a regular basis, will provide a complete and useful picture of current work on NLG in Saarbrücken.

The URL is `http://www.dfki.de/services/NLG/`.

Saarbrücken, September 1997

Tilman Becker
Stephan Busemann
Wolfgang Finkler

Workshop Programme

9.00	–	9.05	Introduction
9.05	–	9.40	**Anne Kilger:** Microplanning in Verbmobil as a Constraint-Satisfaction Problem
9.40	–	10.05	**Tilman Becker:** Syntactic Generation with a Preprocessed HPSG Grammar
10.05	–	10.40	**Stephan Busemann:** Putting Semantic-Head-Driven Generation to the Limits: Experiments with Multi-Purpose Semantic Representations
10.40	–	11.05	*Break*
11.05	–	11.30	**Stephan Busemann, Helmut Horacek:** Generating Air Quality Reports from Environmental Data
11.30	–	12.05	**Xiaorong Huang, Armin Fiedler:** PROVERB: Verbalizing Proofs
12.05	–	12.30	**Xiaorong Huang, Tianfang Yao, Huanye Sheng;** presented by **Yufang Wang:** The Project ACNLG: Applying Natural Language Generation in China
12.30	–	14.00	*Lunch*
14.00	–	14.35	**Wolfgang Finkler:** Nonmonotonic Aspects of Incremental Natural Language Producti Performing Self-Corrections in a Situated Generator
14.35	–	15.10	**Peter Poller:** EFFENDI–EFfizientes FormulierEN von DIalogbeiträgen
15.10	–	15.35	*Break*
15.35	–	16.00	**Thomas Weis:** Resource-Adaptive Dialog Planning
16.00	–	16.25	**Helmut Horacek:** Structural Changes in Natural Language Generation
16.25	–	16.50	**Elke Teich, Erich Steiner, Renate Henschel,** **John Bateman:** AGILE: Automatic Drafting of Technical Documents in Czech, Russian, and Bulgarian
16.50	–	17.00	*Break*
17.00	–	18.00	Final Discussion

Syntactic Generation with a Preprocessed HPSG Grammar

Tilman Becker

German Research Center for Artificial Intelligence (DFKI GmbH)
Stuhlsatzenhausweg 3, 66123 Saarbrücken, Germany
becker@dfki.uni-sb.de

Abstract

The syntactic generator in the dialog translation system Verbmobil is fed by a microplanning component which – after a lexical choice step – generates an annotated dependency structure for the selected words. In order to make maximal use of this input, the Head-driven Phrase-Structure Grammar (HPSG) which is the basis for the syntactic generator is preprocessed to create the complete set of maximal projections from all lexical types in the grammar. With these projections, the generation task consists of finding a suitable combination of such projections. Although there remains a certain trade-off, this setup eliminates the need to apply the HPSG schemata online and allows the use of simpler and cheaper unification steps. The preprocessing we employ is also known as a 'compilation' of HPSG to a Tree Adjoining Grammar (TAG) since the resulting projections are the elementary trees of a TAG grammar.

1 Natural Language Generation in Verbmobil

Verbmobil (see [Wah93, BWW97]), is a system for speech–to–speech dialog translation. The input for the Generation VM–GECO[1] module is generated by a semantic–based transfer component (see [DE96]). The interface language chosen comprises the encoding of target language–specific semantic information following a combination of the Discourse Representation Theory and Minimal Recursion Semantics.
The internal architecture of the generation module is *modularized*: the generation process is separated into two phases, realized by a **microplanner** and a **syntactic generator**. Throughout the system, we emphasize *declarativity*, which is also a necessary precondition for a comprehensive *off–line preprocessing* of external knowledge bases (in particular a preprocessing of the grammar which has been developed at CSLI in the HPSG framework) to perform regularly repeated computations in advance.

2 Modularization: Microplanning and Syntactic Generation

The microplanning component also carries out word–choice. It generates an annotated *dependency structure* which is used by the syntactic generation component to realize a surface string.
One goal of this modularization is a stepwise constraining of the search–space of alternative linguistic realizations, using different views in the different modules. In each step, only an abstraction of the multitude of information contained in an alternative needs to be considered. These abstractions also allow for the expression of generalizations over classes of alternatives, resulting in more efficient processing as well as more compact knowledge bases.

[1] VerbMobil GEneration COmponents

3 Declarativity in the Syntactic Generator

All modules of the generator utilize external, declarative knowledge bases. For the syntactic generator, extensive off-line preprocessing of the highly declarative HPSG grammar for English[2] is applied. The grammar very closely reflects the latest developments of the underlying linguistic theory (see [PS94]) and has not been written exclusively as a generation grammar[3]. It is specialized, however, in that it covers phenomena of spoken language. The high level of abstraction which is achieved in the hierarchically organized grammar description (see [Fli87]) allows for easy maintenance as well as off-line preprocessing. The off-line preprocessing steps described in the next section keep the declarative nature of the grammar intact, i.e. they retain explicitly the phrase structures and syntactic features as defined by the HPSG grammar.
In general, declarative knowledge bases allow for an easier adaptation of the system to other domains and languages. This is a huge benefit in the current second phase of the Verbmobil project [BFKW96] where the generator is extended to cover German, English and Japanese as well as additional and extended domains with a considerably larger vocabulary.

4 Off–Line Preprocessing: HPSG to TAG Compilation

The subtasks in a syntactic generation module based on an HPSG grammar will always include the application of schemata such that all *syntactic* constraints introduced by a lexical item (especially the SUBCAT list are fulfilled. This results in a constant repetition of e.g. building up the projection of a verb in a declarative sentence. In preprocessing the HPSG grammar we aim at computing all possible partial phrase structures which can be derived from the information in a lexicon entry. Given such sets of possible syntactic realization together with a set of selected lexicon entries for an utterance and finally their dependencies, the task of a syntactic generator is simplified considerably. It now does not need to explore all possible, costly applications of HPSG schemata but merely has to find suitable precomputed syntactic structures for each lexical item and combine them appropriately.
For this preprocessing of the HPSG grammar, we adapted the process described in [KKNVS95]. The basis for the compilation is an identification of syntactically relevant *selector features* which express subcategorization requirements of a lexical item, e.g. the VALENCE features. In general, a phrase structure is complete when these selector features are empty.
Starting from the feature structure for a lexical item, HPSG schemata are applied such that the current structure is unified with a daughter feature of the schema. The resulting structure is again subject to this process. This compilation process stops when certain termination criteria are met, e.g. when all selector features are empty. Thus, all projections from the lexical item are collected as a set of minimally complete phrase structures which can also be interpreted as *elementary trees* of a Tree–Adjoining Grammar (TAG). Instead of applying this compilation process to all lexical items, certain abstractions over the lexical entries are specified in the HPSG grammar. In fact, the needs of the compilation process have led to a clear–cut separation of lexical types and lexical entries as shown in Figure 1. A typical lexical entry is shown in Figure 2 and demonstrates that only three kinds of information are stored: the lexical type (MV_NP_TRANS_LE[4]), the semantic contribution (the relation _SUIT_REL) and morphological information (the stem and potentially irregular forms). By expanding the lexical type, the full feature structure can be obtained.
Some of the trees which result from the preprocessing of the lexical type MV_NP_TRANS_LE are shown in Figure 2. The figure shows only the phrase structure and an abstraction of the node's categories. All nodes still represent the full HPSG feature structures. E.g., the tree MV_NP_TRANS_LE.2 of Figure 2 represents an imperative clause. As a consequence PERSON has the value SECOND and CL-MODE is set to IMPERATIVE. Note that the compilation process stopped at this node since the selector features are empty.
From these trees, two kinds of knowledge bases are built. For the microplanner, the relation between

[2]The HPSG grammar is being developed at CSLI, Stanford University. Development is carried out on a grammar development platform which is based on TDL [KS94].

[3]In fact, most of the testing during grammar development depends on the use of a parser.

[4]MV_NP_TRANS_LE represents 'transitive main verb with NP object.'

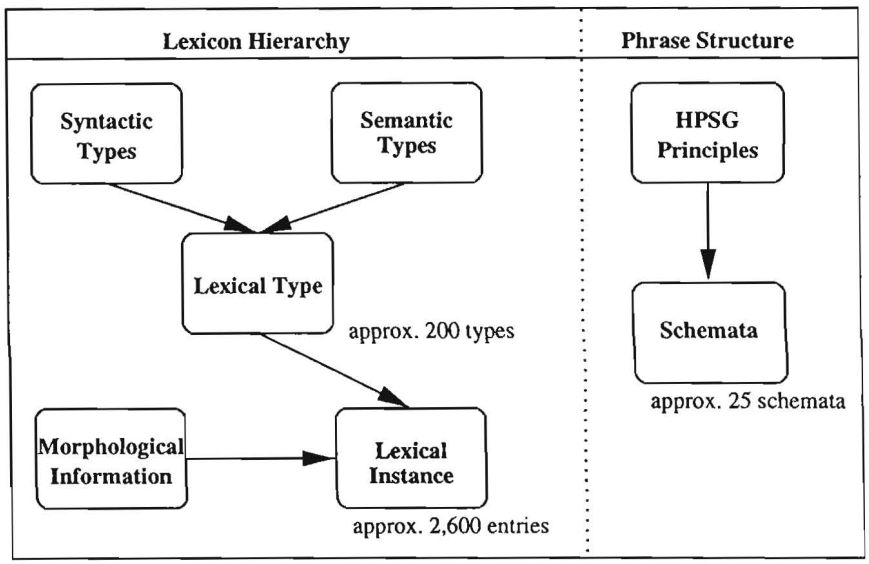


Figure 1: Organization of the HPSG grammar.

```
suit_v1 := mv_np_trans_le &
  [ STEM < "suit" >,
    SYNSEM.LOCAL.CONT.STEMLISZT <! [ PRED _suit_rel ] !> ].
```

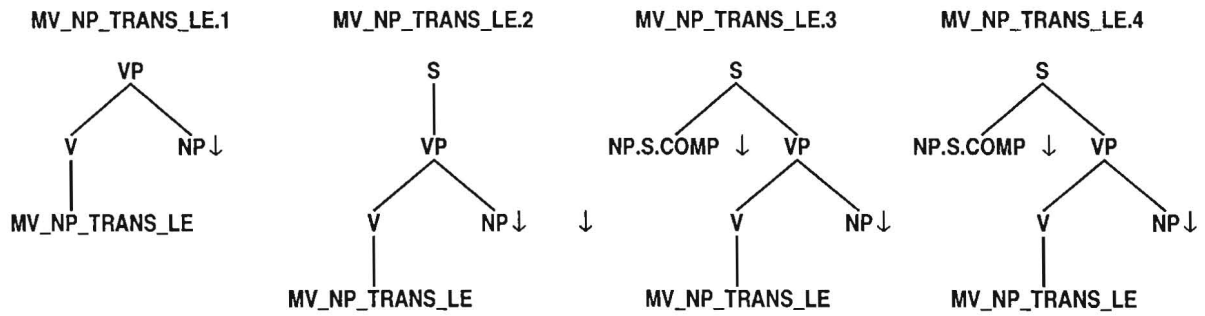


Figure 2: Specification of a lexical instance for the verb "suit" and some of the trees for transitive verbs. They are compiled from the corresponding lexical type MV_NP_TRANS_LE as defined in the HPSG grammar. Trees 3 and 4 differ only with respect to their feature structures which are not shown in this figure.

the lexical and syntactic realization and the semantic representation (encoded in the SYNSEM LOCAL CONT feature) is extracted as a constraint. For the syntactic generator, the relevant syntactic information is extracted in the form of a Feature–Based Lexicalized TAG (FB-LTAG) grammar, see [JLT75, VSJ91, SAJ88]. This includes the phrase structure and a selected part of the feature structure (mainly the SYNSEM LOCAL CAT and SYNSEM NON-LOCAL features). Figure 3 shows the bottom fea-

ture structure extracted from the root node of MV_NP_TRANS_LE.2. Note that some of the feature paths are abbreviated, e.g. SLCI stands for SYNSEM LOCAL CONT INDEX. The elementary TAG trees which are built from the compilation result have so–called *restricted feature structures* which can be exploited for an efficient, specialized unification algorithm.

Bottom Dag at selected node:

```
[:ROOT: [SLC: [HEAD: [PRD: (- +)]
              [MOOD: (SUBJUNCTIVE MODAL_SUBJ INDICATIVE)]
              [VOICE: (PASSIVE ACTIVE)]
              [TENSE: (FUTURE PAST PRESENT)]
              [VFORM: BSE]
              [INV: -]
              [AUX: -]
         [ROOT: +]
         [CL-MODE: IMPERATIVE]
    [RULE: IMPERATIVE_RULE]
    [SLCI: NIL]
    [SYNSEM: [NON-LOCAL: [QUE: -]
```

Figure 3: The bottom feature structure of the S node of tree MV_NP_TRANS_LE.2.

The node names shown in the figures represent a disjunction of possible categories, e.g. NP.S.COMP in tree MV_NP_TRANS_LE.3 implies that the subject of a transitive verb may be a nominal or sentential phrase.
Finally, the leave nodes of the trees (except for the lexical item itself) are marked either as substitution nodes or as a foot node, thus creating an auxiliary tree. In a TAG derivation, substitution nodes are replaced with trees bearing the correct category and a unifyable feature structure at their root node. Auxiliary trees can be inserted into other trees by the adjunction operation.

5 The Syntactic Generator VM-GIFT

The task of the syntactic generator is the construction of a sentence (or phrase, given the often incomplete utterances in spoken dialogs) from the microplanning result which is then sent to a speech-synthesis component. It proceeds in four major steps which are also depicted in Figure 4.

- A **preprocessing** phase computes the necessary auxiliary verbs from the tense, aspect, and sentence mood information. It also rearranges the dependency tree accordingly (e.g. subject arguments are moved from the main verb to become dependents of the inflected auxiliary verb).
- A **tree selection** phase determines the set of relevant TAG trees. A first tree **retrieval** step maps every object of the dependency tree into a set of applicable elementary TAG trees. The main tree **selection** phase uses information from the microplanner output to further refine the set of retrieved trees.
- A **combination** phase finds a successful combination of trees to build a (derived) phrase structure tree.
- An **inflection** phase uses the information in the feature structures of the leaves (i.e. the words) to apply appropriate morphological functions, including the use of *irregular forms* as provided by the HPSG lexicon and *regular inflection* function as supplied (as LISP code) by the HPSG grammar.

The two main phases are the tree selection and the combination phase. The tree selection phase consists of two steps. First, a set of possible trees is retrieved and then appropriate trees are selected from this set. The retrieval is driven by the HPSG instance or word class that is supplied by the microplanner. It is mapped to a lexical type by a lexicon that is automatically compiled from the HPSG grammar. The lexical types are then mapped to a tree family, i.e., a set of elementary TAG trees representing all possible minimally complete phrase structures that can be build from the instance. The additional information in the dependency tree is then used to add further feature values to the trees. This additional information acts as a filter for selecting appropriate trees in two stages:

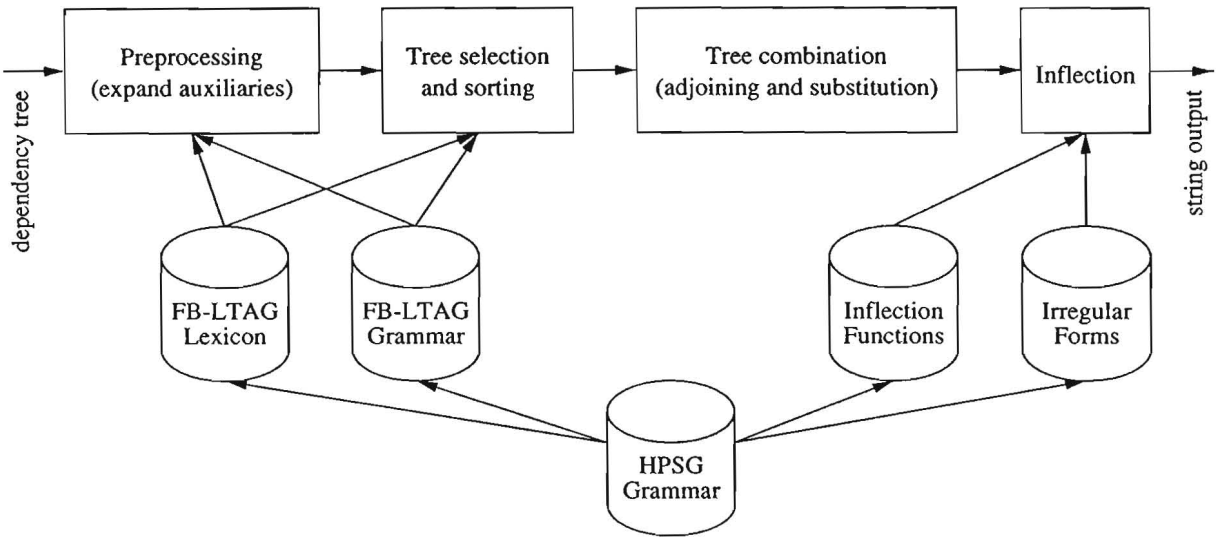


Figure 4: Steps of the syntactic generator.

- Some values are incompatible with values already present in the trees. These trees can therefore be filtered immediately from the set. E.g., a syntactic structure for an imperative clause is marked as such by a feature and can be discarded if a declarative sentence is to be generated.
- Additional features can prevent the combination with other trees during the combination phase. This is the case, for example with agreement features.

The combination phase explores the search space of all possible combinations of trees from the candidate sets for each lexical item (instance). An inefficient combination phase is a possible drawback of using the precomputed TAG trees. Fortunately, there is sufficient information available from the microplanner result and from the trees such that a well-guided best-first search strategy can be employed in the current system. The difference in run-time can be as dramatic as 24 seconds (comprehensive breadth-first) versus 1.5 seconds (best-first).

As part of the tree selection phase, based on the rich annotation of the input structure, the tree sets are sorted locally. Then a backtracking algorithm traverses the dependency tree in a bottom-up fashion[5]. At each node, and for each subtree in the dependency tree, a candidate for the phrase structures of the subtree is constructed. Then all possible adjunction or substitution sites are computed, possibly sorted (e.g. allowing for preferences in word order) and the best candidate for a combined phrase structure is returned. Since the combination of two partial phrase structures by adjunction or substitution might fail due to incompatible feature structures, a backtracking algorithm must be used. A partial phrase structure for a subtree of the dependency is finally checked for completeness. These tests include the unifiability of all top and bottom feature structures and the satisfaction of all other constraints (e.g. obligatory adjunctions or open substitution nodes) since no further adjunctions or substitutions will occur in this subtree.

The necessity of a spoken dialog translation system to produce output robustly calls for some relaxations in these tests. E.g. 'obligatory' arguments may be missing in the utterance and the tests in the syntactic generator must accept a sentence with a missing subject if no other complete phrase can be generated.

Figure 5 shows an example for the input of from the microplanner after the preprocessing phase has inserted the entity LGV1 for the auxiliary *will*.

In the tree retrieval phase for L5-WORK_ACCEPTABLE, first the HEAD information is used to determine the lexical types of the possible realizations SUIT_V1 and SUIT_V2, namely MV_NP_TRANS_LE and MV_EXPL_PREP_TRANS_LE respectively. These types are then mapped to their respective sets of elementary trees, a total of 25 trees.

[5]The algorithm stores intermediate results with a memoization technique.

```
((ENTITY LGV1
  ((CAT V) (HEAD WILL_AUX_POS) (INTENTION WH-QUESTION) (FUNC AUX)
   (TENSE FUTURE) (MOOD INDICATIVE) (VOICE ACTIVE) (FORM ORDINARY)
   (VFORM FIN)))
 (ENTITY L5-WORK_ACCEPTABLE
  ((FORM ORDINARY) (VFORM BSE) (CAT V) (GOVERNED-BY WH-SENTENCE)
   (OPTIONAL-AGENT NO) (HEAD (OR SUIT_V1 SUIT_V2)) (REALIZED LOCAL)
   (REG LGV1)))
 (ENTITY L13-PRON
  ((REALIZED LOCAL) (CAT PPRON) (PERS 3) (NUM SG) (GENDER NTR)
   (TYPE NORMAL) (GOVERNED-BY V) (IS-COMPLEMENT T) (FORM CONTINUOUS)
   (REG LGV1) (FUNC AGENT)))
 (ENTITY L10-PRON
  ((REALIZED LOCAL) (CAT PPRON) (PERS 2A) (NUM SG) (GENDER FEM) (TYPE NORMAL)
   (GOVERNED-BY (OR V PREP SENTENCE)) (FORM CONTINUOUS) (REG L5-WORK_ACCEPTABLE)
   (FUNC PATIENT)))
 (ENTITY L6-TEMP_LOC
  ((CAT ADV) (REAL WH_QUEST) (SORT TIME) (POINTED-BY TEMP_LOC)
   (GOVERNED-BY (OR V N ADV SENTENCE)) (PRED TIME) (HEAD WHEN1)
   (REALIZED LOCAL) (WH-FOCUS T) (REG L5-WORK_ACCEPTABLE) (FUNC TEMP-SPEC)))
 (ENTITY L15-TEMP_LOC
  ((CAT ADV) (HEAD THEN_ADV) (REALIZED GROUP-TIME-DEMONSTRATIVE)
   (REAL (OR ADV WH_QUEST YOFC)) (SORT (SUBSORT TIME)) (POINTED-BY TEMP_LOC)
   (GOVERNED-BY (OR V N ADV SENTENCE)) (REG L5-WORK_ACCEPTABLE) (FUNC TEMP-SPEC))))
```

Figure 5: Example for input from microplanning after preprocessing for auxiliaries

In the tree selection phase, this number is reduced to six. For example, the tree MV_NP_TRANS_LE.2 in Figure 2 has a feature CL-MODE with the value IMPERATIVE. Now, the microplanner output for the root entity LGV1 contains the information (INTENTION WH-QUESTION). The INTENTION information is unified with all appropriate CL-MODE features, which in this case fails. Therefore the tree MV_NP_TRANS_LE.2 is discarded in the tree selection phase.
The combination phase uses the best-first bottom-up algorithm described above to determine one suitable tree for every entity and also a target node in the tree that is selected for the governing entity. For the above example, the selected trees and their combination nodes are shown in Figure 6[6].

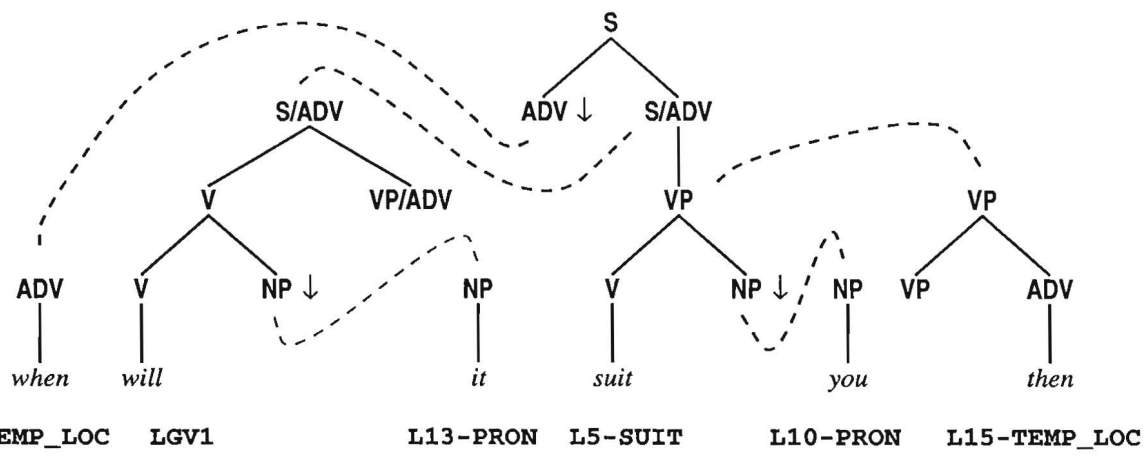


Figure 6: The trees finally selected for the entities of the example sentence. The dashed lines connect to suitable substitution or adjunction nodes. They correspond to the dependency tree.

[6]Note that the node labels shown in Figures 6 are only a concession to readability. The TAG requirement that in an auxiliary tree the footnode must have the same category label as the root node is formally fulfilled in our system.

The inflection function finally uses attribute values like verb-form, number and person from the final tree to derive the correct inflections. Information about the sentence mode WH-QUESTION can be used to annotate the resulting string for the speech-synthesis module.

6 Conclusion

We have shown how preprocessing an HPSG grammar can be used to avoid the costly on–line application (unification) of HPSG schemata in a modularized generation system with a microplanner and a separate syntactic generator. The compilation of an HPSG grammar to TAG grammar allows the use of an efficient syntactic generator without sacrificing the declarative nature of the HPSG grammar.

References

[BFKW96] T. Becker, W. Finkler, A. Kilger, and W. Wahlster. Vorhabensbeschreibung zur Sprachgenerierung innerhalb des Teilprojektes 5 (Sprachgenerierung und –synthese) in Verbmobil, Phase 2. Document, German Research Center for Artificial Intelligence (DFKI GmbH), Saarbrücken, Germany, August 1996.

[BWW97] Th. Bub, W. Wahlster, and A. Waibel. Verbmobil: The combination of deep and shallow processing for spontaneous speech translation. In *Proceedings of ICASSP '97*, 1997. (forthcoming).

[DE96] M. Dorna and M. Emele. Semantic–based transfer. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING '96)*, 1996.

[Fli87] Daniel P. Flickinger. *Lexical Rules in the Hierarchical Lexicon.* PhD thesis, Stanford University, 1987.

[JLT75] A.K. Joshi, L. S. Levy, and M. Takahashi. Tree Adjunct Grammars. *J. Comput. Syst. Sci.*, 10(1), 1975.

[KKNVS95] R. Kasper, B. Kiefer, K. Netter, and K. Vijay-Shanker. Compilation of hpsg to tag. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 92–99, Cambridge, Mass., 1995.

[KS94] Hans-Ulrich Krieger and Ulrich Schäfer. *TDL*—a type description language for constraint-based grammars. In *Proceedings of the 15th International Conference on Computational Linguistics, COLING-94*, pages 893–899, 1994.

[PS94] Carl Pollard and Ivan A. Sag. *Head-Driven Phrase Structure Grammar.* Studies in Contemporary Linguistics. University of Chicago Press, Chicago, 1994.

[SAJ88] Y. Schabes, A. Abeillé, and A.K. Joshi. Parsing strategies with 'lexicalized' grammars: Application to Tree Adjoining Grammars. In *Proc. 12th International Conference on Computational Linguistics (COLING-88)*, pages 578–583, Budapest, August 1988.

[VSJ91] K. Vijay-Shanker and Aravind K. Joshi. Unification Based Tree Adjoining Grammars. In J. Wedekind, editor, *Unification-based Grammars.* MIT Press, Cambridge, Massachusetts, 1991.

[Wah93] W. Wahlster. Verbmobil: Translation of face–to–face dialogs. Research Report RR-93-34, German Research Center for Artificial Intelligence (DFKI GmbH), Saarbrücken, Germany, 1993.

Putting Semantic-Head-Driven Generation to the Limits: Experiments with multi-purpose semantic representations

Stephan Busemann
German Research Center for Artificial Intelligence (DFKI GmbH)
Stuhlsatzenhausweg 3, 66123 Saarbrücken (Germany)
busemann@dfki.uni-sb.de

Abstract

Constraint-based grammars can, in principle, serve as the major linguistic knowledge source for both parsing and generation. Surface generation starts from input semantics representations that may vary across grammars. For many declarative grammars, the concept of derivation implicitly built in is that of parsing. They may thus not be interpretable by a generation algorithm. We show that linguistically plausible semantic analyses can cause severe problems for semantic-head-driven approaches for generation (SHDG). We use SEREAL, a variant of SHDG and the DISCO grammar of German, both developed at DFKI, as our source of examples. We propose a new approach that explicitly accounts for the interface between the grammar and the generation algorithm by adding a control-oriented layer to the linguistic knowledge base that reorganizes the semantics in a way suitable for generation.

1 Introduction

Semantic-Head-Driven Generation (SHDG) (Shieber et al., 1990) is one of the most widespread algorithms for sentence realization with constraint-based grammars. It is largely theory-independent and has been used for Head-Driven Phrase Structure Grammars (HPSG), Definite Clause Grammars, and Categorial Unification Grammars. Since its publication, SHDG had to compete with other algorithms (e.g. (Russell et al., 1990), (Strzalkowski, 1994), (Martinovic and Strzalkowski, 1992)) which led to numerous ways of improving the basic procedure.

A major question remained unsolved (and it is unsolved for other algorithms as well), namely that of the algorithm's requirements on the properties of the grammar used. In previous work, Shieber imposed a condition on "semantic monotonicity" that holds for a grammar if for every phrase the semantic structure of each immediate subphrase subsumes some portion of the semantic structure of the entire phrase (Shieber, 1988, p. 617). Semantic monotonicity is very strict and could be relaxed in SHDG: It was shown that semantically non-monotonic grammars can be processed by SHDG. It is a yet open question whether all semantically monotonic grammars can be processed by SHDG and what the class of SHDG-processable grammars is.

In this paper we show that additional problems may occur with semantic representations that are linguistically well motivated. Using the semantics of the DISCO system (Dialogue System for Cooperating agents) developed at DFKI (Uszkoreit et al., 1994) as an example, we show that there are semantically monotonic grammars that cannot be processed directly by SHDG. We discuss possible methods to solve the problem and propose a new approach that explicitly accounts for the interface between the grammar and the generation algorithm by adding a control-oriented layer to the linguistic knowledge base that reorganizes the semantics in a way suitable for generation.

The kind of problem investigated in this paper relates to the fundamental question of how to organize a modular system consisting of linguistic knowledge (a grammar) and control knowledge (parser or generator). It turns out that declarative grammars contain hidden assumptions about processing issues.

2 SHDG and the Grammar Interface

We briefly review some essential points of SHDG.[1] The algorithm is centered around the notion of a *pivot* node, which provides an essential feature specification from which it first generates all descendants

[1] We assume the reader to be familiar with SHDG as described by (Shieber et al., 1990).

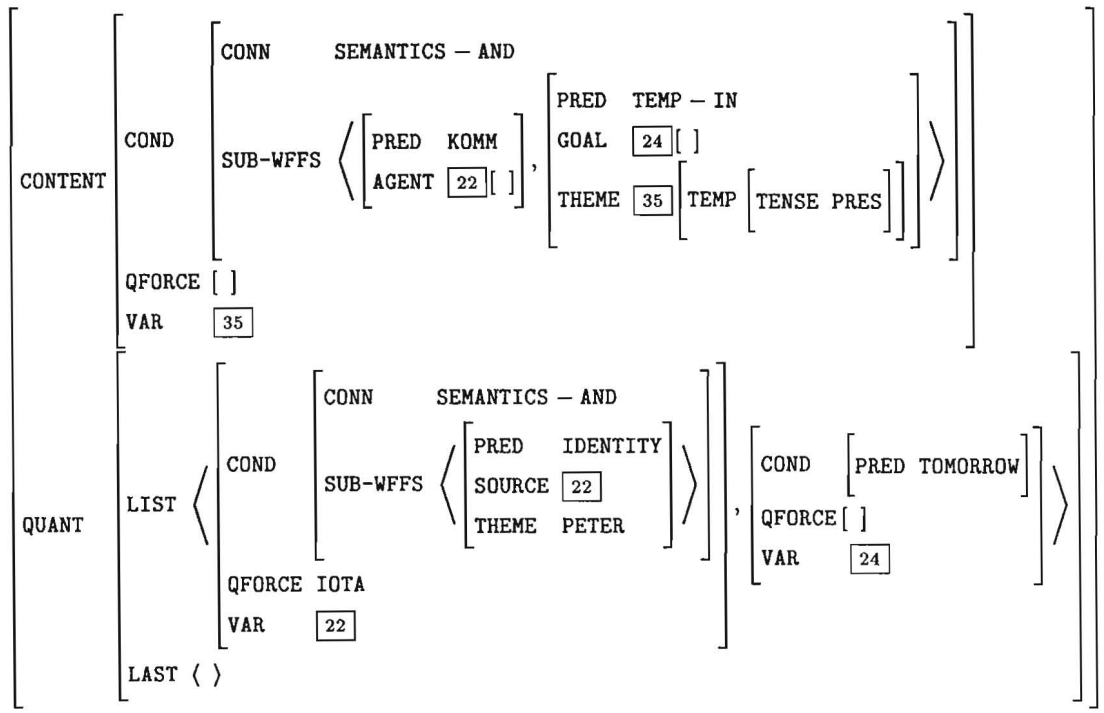


Figure 1: Semantic Feature Structure for *Peter kommt morgen* [Peter arrives tomorrow].

in a top-down manner, and then tries to connect the newly generated subtree to a higher node (or the root node) in bottom-up fashion. Both generating descendants and connecting to higher nodes involves the application of grammar rules. Correspondingly, rules are subdivided into two classes: *chain rules* are used for bottom-up connection while *non-chain rules* are applied for top-down expansion. Chain rules differ from non-chain rules in that their left-hand side essential feature is identical to the essential feature of one of their right-hand side elements. This element is called the "semantic head" of the chain rule. Lexical entries are non-chain rules in a trivial way since they have no categorial right-hand side elements.

The only specific assumption SHDG makes about a grammar is that chain rules and their semantic heads can be identified. However, the property of being a chain rule (or non-chain rule) is often assigned by the grammar writer on purely linguistic grounds although it determines the processing strategy: If the set of chain rules happens to be empty, SHDG operates strictly top-down. If the set of non-chain rules consists of lexicon entries only, SHDG behaves like a bottom-up generator. Having the linguist unconsciously influence the processing strategy of SHDG can lead to uninterpretable grammars, as we will show below.

We now introduce some basic assumptions about grammars. A grammar induces a context-free backbone and has separate layers to represent morphological, syntactic, and semantic properties of categories. We assume furthermore that the generator can be told how to identify mother and daughter categories of grammar rules. The generator is guided by its input layer, the semantics. Thus we refer to the input layer as the *essential feature*.

The under-specification of the essential feature at execution time is a well-known phenomenon (Russell et al., 1990). It can show up during top-down expansion of a grammar rule that does not share the essential features of the daughters with parts of the mother. Non-termination or failure to find a derivation will result. However, a generator must *terminate* on all allowable input. We thus formulate a condition on generator/grammar pairs that ensures successful recursive applicability of the generation procedure:

> **Essential Feature Specification Condition (EFSC):** The essential feature must specify exactly the constituent to be generated *at the time the generation procedure is executed* on it.

Obviously, this requirement needs to be concretized in terms of specific algorithms since the or-

der in which a generator processes right-hand side elements of rules is crucial. EFSC for SHDG depends on the order in which nodes of a local tree are recursively expanded. (Shieber et al., 1990) quite arbitrarily assume a strict left-to-right processing of non-semantic-head daughter nodes. EFSC is easily violated by a daughter of a non-chain rule that influences the essential feature of a preceding daughter.

3 The System Setup

This section introduces the generator/grammar pair used for the present study. After a sketch of our variant of SHDG we discuss the semantics layer of the constraint-based grammar of German to the extent necessary to demonstrate violation of EFSC and to describe a solution.

3.1 The SEREAL system

The SEREAL (Sentence Realizer) is a Common Lisp SHDG implementation that uses kernel components of the DISCO NL understanding system (Uszkoreit et al., 1994).

DISCO is a linguistic core engine capable of analyzing NL sentences as quasi-logical form representations that can subsequently be submitted to further semantic analysis. The DISCO grammar is encoded in TDL (Krieger and Schäfer, 1994), a powerful type definition language and type inference mechanism for feature structures. The basic processing engine is the feature constraint solver UDINE, which is used to perform (destructive) unification during parsing and generation. A mapping between word forms and morpho-syntactically annotated word stems is achieved by the MORPHIX-3 system (Finkler and Neumann, 1988).

SEREAL is integrated into the DISCO system to the extent that it uses the same grammar, UDINE, TDL, and MORPHIX-3. It can be fed with the parser's semantics output and thus serve as a useful grammar development tool.

A special mechanism had to be developed for efficient lexicon access. The SHDG algorithm simply assumes all lexicon entries to be available as non-chain rules. This is, however, not advisable for large lexicons. Rather, only the relevant entries should be accessed. Therefore, SEREAL indexes the lexicon according to semantic information. Consider, for instance, the semantic representation in Figure 1.[2]

[2]This is a simplified version of a semantic representation taken from a parse with the DISCO grammar. For presentation purposes we adopt the familiar matrix notation for feature structures. < and > are print macros for lists that expand into the common feature structure notation for lists (cf. (Shieber, 1986, page 29)). Although TDL defines *typed* feature structures, we omit type information here as it is not relevant.

Lexical indices usually are semantic predicates denoted by the PRED feature, e.g. KOMM is the index for the main verb (*arrive*). Exceptions include determiners, which are indexed according to the value of QFORCE and proper names, which are indexed according to the value of THEME. A priority system on indices (THEME > QFORCE > PRED) reduces the number of accessible indices. This way an index points to very few lexicon entries.[3] Indices are retrieved as values of some path in the essential feature specification. Insertion of an entry into a derivation requires its essential feature to subsume the input structure in order to prevent the violation of the coherence condition.

Clearly both indices and path descriptions are grammar dependent and form a part of the interface between SEREAL and the DISCO grammar. In Figure 1, the following indices are used to access lexicon entries: KOMM, PETER, TEMP-IN.

The algorithm has been criticized for not terminating on left-recursive rules (Strzalkowski, 1994). Under the assumption of semantic monotonicity, the determination of a pivot can be conditioned by a check for semantic content. If the semantics is "empty" (i.e., it corresponds to the top feature structure), processing fails and alternative possibilities have to be explored. Since left recursion occurs only in top-down direction, we are dealing with non-chain rules, which ensures that the semantics of a right-hand side element differs from that of the left-hand side. Semantic monotonicity ensures that it is "smaller" in some sense, thus guaranteeing termination.

(Martinovic and Strzalkowski, 1992) criticized the possible failure of top-down expansion due to the strict left-to-right processing of the list of right-hand side elements. Since the instantiation of the semantics of some right-hand elements can depend on the previous successful expansion of others, a strict order that does not consider such relations is inadequate. In SEREAL, the left-most right-hand side element of a rule is expanded first that has a non-empty semantics instantiated.

3.2 The DISCO semantics layer

The DISCO grammar is a semantically monotonic lexicalized, HPSG-style grammar of German with about 20 rules, 13 of them binary. The remaining ones are unary (lexical) rules that serve to introduce

[3]This depends on how many lexemes carry the same index. Usually we have one to three, in rare cases up to fifteen, entries per index.

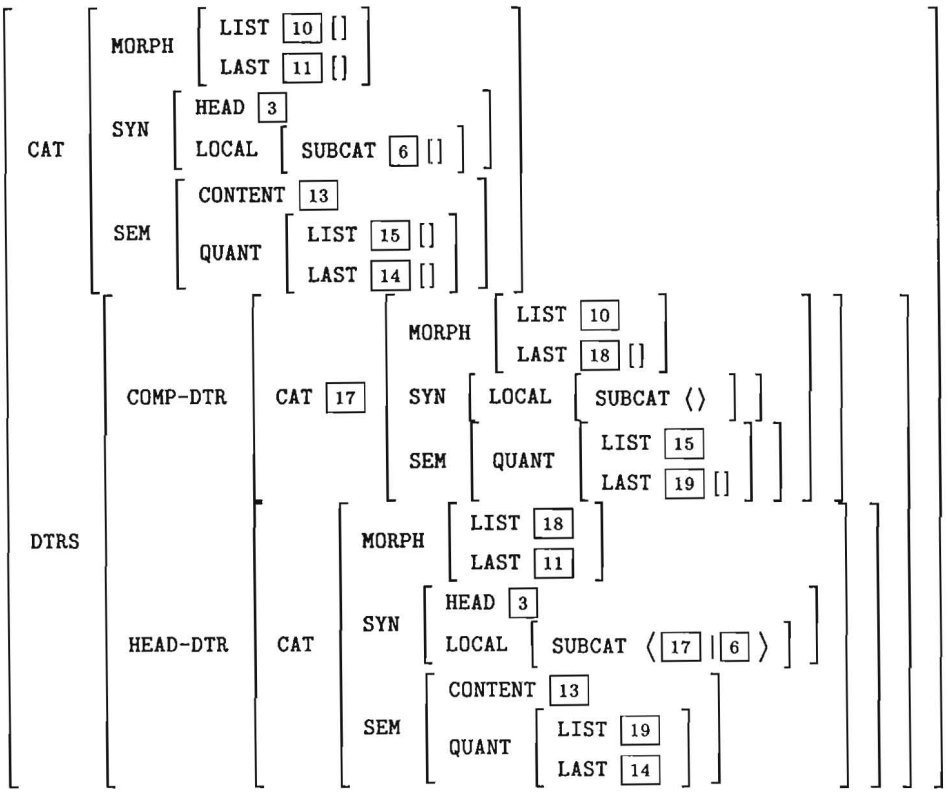


Figure 2: A Head-Complement Rule (simplified for expository purposes).

syntactic features for lexemes in particular environments. For instance, verb lexemes can be made finite or infinite, adjectives can be made attributive or predicative. The binary rules account for complement and adjunct realization.

The development of the DISCO grammar was, as many others, based on purely linguistic motivations. Although a declarative representation is used, the concept of derivation implicitly built in is that of (bottom-up) parsing. Again, this is common. The parsing view of the grammar developer influences the goals that a semantic representation should fulfill. The DISCO semantics layer should

- represent a linguistically well motivated (surface) propositional semantics of NL sentences,
- provide the interface to subsequent non-compositional, extra-grammatical semantic interpretation (e.g. anaphora resolution, scope disambiguation), and
- represent the essential feature for grammar-based sentence realization.

The semantics layer corresponds to quasi-logical forms (Alshawi, 1992) that are defined through the grammar and represented with help of feature structures (Nerbonne, 1992). The relevance of the surface ordering of complements and adjuncts during later semantic processing made it necessary to encode ordering information at the semantics layer. This is reflected by the QUANT feature, which contains a list of the semantics of the complements and adjuncts in the order they occur at the surface. The relations between them are expressed by the CONTENT feature with help of the VAR feature.

Consider as an example the semantics structure in Figure 1. QUANT has two elements, the first one representing the proper name and the second one the temporal adverb *tomorrow*. CONTENT represents a CONDition on the meaning consisting of a conjunction of sub-formulae. The first formula represents a one-place predicate KOMM, the argument of which points, via VAR, into the first element of the QUANT list. The second sub-formula represents a two-place predicate TEMP-IN. Its first argument points into the second element of QUANT, and its second argument relates to the whole CONTENT feature. Thus the predicate is to be interpreted as a temporal sentential modifier.

Semantic information mainly originates from lexical entries. A few general principles of feature distribution are represented with the grammar rules. Figure 2 shows a head-complement rule with the complement being the first element of the head's subcategorization list. The complement is preceding the head (not shown). CONTENT is shared between the mother (CAT) and the head daughter. In a rule's left-hand side constituent, QUANT denotes the concatenation of the QUANT values of the sequence of right-hand side elements.

List concatenation is encoded using difference lists. Thus it is not necessary to use functional feature values such as append. The difference list type built into in TDL denotes a list L by defining a list L1 under the feature LIST and another list L2 under the feature LAST such that L2 is a tail of L1 and the concatenation of L and L2 yields L1. This can easily be achieved by choosing appropriate coreferences.

In the case of bottom-up processing, this mechanism is used like a stack: at the mother node, the QUANT feature of the complement semantics has been pushed onto the list of elements collected so far (at the head daughter).

4 A Violation of EFSC

Investigation of the grammar rules shows that there are no binary chain rules since the QUANT feature within SEM differs at all nodes of a rule (cf. Figure 2). With the resulting top-down strategy the QUANT list at the mother node must be split into two sublists in order to instantiate the QUANT lists of the daughter nodes. This is a nondeterministic problem that, given the present implementation of difference lists, leads to under-specification.

Unification of some input semantics with the mother node (in Figure 2 under CAT.SEM[4]) does not specify how the QUANT list should be split, i.e. the QUANT.LAST feature of the COMP-DTR semantics, which is shared with the QUANT.LIST feature of the HEAD-DTR semantics, is not affected at all by this unification operation. Any further expansion steps using similar rules will not specify the semantics any further, and hence non-termination results.[5]

This problem is not specific to the DISCO grammar. Difference lists are a common descriptive device used in many constraint-based grammars. For instance, the same problem arises with the minimal recursion semantics, a framework for semantics within HPSG, which was developed to simplify transfer and generation for machine translation (Copestake et al., 1995).

Neither is the problem specific to SEREAL or SHDG. It is specific to top-down processing of difference lists in general.

5 Reorganizing Semantic Information

Whenever a grammar/generator pair violates EFSC, two basic directions offer themselves as remedies: Either the generator is modified to account for the grammatical analysis, or the grammar is adapted to the needs of the generator.

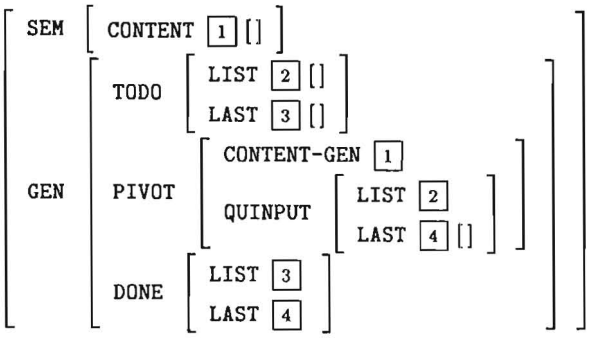


Figure 3: The Organization of the GEN Layer.

Grammar writing should be guided by linguistic adequacy considerations rather than by algorithmic issues. Linguistically plausible analyses should not be rejected because they are not processed by the generator used. On the other hand, designers of generation (or parsing) algorithms want to create generic tools that can be used for large classes of grammars. Such algorithms, including those of the SHDG type, should not be geared towards a particular grammar. Moreover, in a large grammar, processing problems may occur with several phenomena, and solving them either way would eventually sacrifice the modularity of the grammar and the generator.

In conclusion, neither of the two ways is satisfactory. In this contribution we present a novel approach that complements a single grammar by an explicit and modular interface layer that restructures the semantic information in such a way that it supports bottom-up processing within SEREAL.

[4]We use the period between feature names to denote feature path descriptions.

[5]It may be argued that the CONTENT feature could serve as a pivot. It is indeed shared between mother and head in most rules, which would then be chain rule candidates. However, semantic information necessary to guide the generation of many phrasal constituents may be represented only by QUANT.

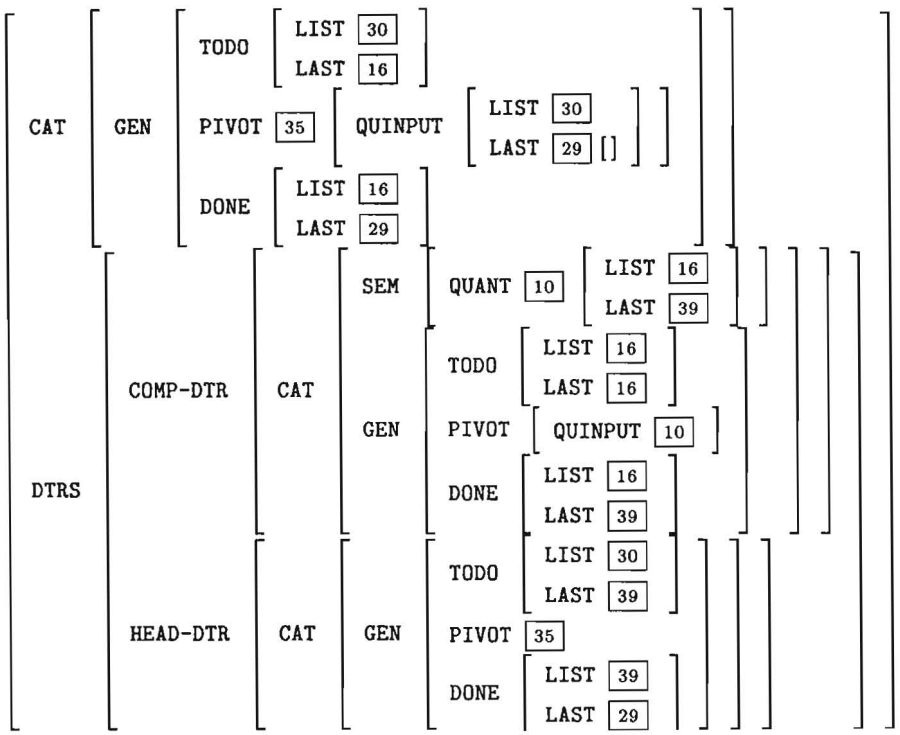


Figure 4: The GEN Feature in a Head-Complement Rule.

This method improves over previous approaches in various ways:

- The interface is defined declaratively;
- Reversibility properties of the grammar are preserved;
- The modularity of the grammar and the generator are preserved.

This layer, GEN, is assigned to every category of the grammar (cf. Figure 3). Its definition does not modify the grammar, rather a new module is added to it. Since semantic information is not constrained, but just restructured in GEN, reversibility properties of the grammar are not touched. Parsing results are completely independent from the presence of GEN. Since the restructuring is achieved by using coreferences with the parts of the semantic layer, generation uses the same kind of semantic information as parsing. Hence, SEREAL will deliver all sentences for a semantic representation restructured in GEN that yield that semantic representation when they are parsed.

Within GEN we define a new essential feature, PIVOT, that shares the semantic content (under CONTENT-GEN) and contains the QUANT list of the input (under QUINPUT). We specify explicitly the sublist of QUINPUT covered by the subtree represented by the category at hand using the list DONE, and we also note the list of remaining elements that still need to be processed (TODO). This is encoded using difference lists.

The binary grammar rules are extended as follows (Figure 4 shows the GEN feature added to the rule in Figure 2). Mother and head daughter share their PIVOT features, which yields us chain rules (and the desired bottom-up processing strategy). Obviously the mother's DONE list must be the concatenation of all daughters' DONE lists. Moreover, the complement daughter's TODO list must be empty, which is why QUINPUT and DONE coincide. QUINPUT of the complement daughter is shared with SEM.QUANT. It is completely specified after the subtree represented by the head daughter has been completed.

6 Conclusion

Interfaces between constraint-based grammars and generation systems must be defined in a very specialized way. In this paper we have introduced a general condition on grammars, EFSC, which offers the pos-

sibility to identify different sources of failure. In view of the disadvantages of current approaches dealing with EFSC violations, we have introduced into the descriptive framework a new, control-oriented layer of representation, GEN, that reorganizes semantic information in such a way that it does not violate EFSC for the generation algorithm used.

GEN is the essential feature of a generation procedure and serves to define the interface between a grammar and a generator. This way, the interface is explicitly and declaratively defined. Besides architectural advantages, this approach has considerable practical benefits compared to compilation methods. It uses the same representational means that serve for the implementation of the grammar. If a grammar writer chooses to modify the encoding of certain linguistic phenomena, potential clashes with the interface definitions can be detected and removed more easily.

Although the method is generally applicable, the GEN layer must be defined explicitly for every grammar/generator pair. Depending on whether and where EFSC is violated, GEN may just co-specify the semantics (the trivial case), or reconstruct the semantics in an EFSC-compatible fashion. An instance of the latter was described above for the DISCO grammar and SEREAL. If a different generator is chosen for the DISCO grammar, neither the algorithm nor the grammar needs to be modified. The same holds true, if SEREAL was to interpret a different grammar. In both cases, it is the definition of GEN that would have to be replaced.

The techniques presented are implemented in TDL and CommonLisp within the SEREAL system.

Acknowledgments

This work has been supported by a grant from The German Federal Ministry for Research and Technology (FKZ ITW 9402). I am grateful to Feiyu Xu for implementing a first version of the interface, and to Jan Alexandersson, Edmund Grimley Evans, and Harald Lochert for implementing parts of the SEREAL system.

References

Hiyan Alshawi, editor. 1992. *The Core Language Engine.* ACL-MIT Press Series in Natural Language Processing. MIT Press, Cambridge MA.

Ann Copestake, Dan Flickinger, Robert Malouf, Susanne Riehemann, and Ivan A. Sag. 1995. Translation with minimal recursion semantics. In *Proc. 6th International Conference on Theoretical and Methodological Issues in Machine Translation*, Leuven.

Wolfgang Finkler and Günter Neumann. 1988. Morphix: A fast realization of a classification-based approach to morphology. In H. Trost, editor, *Proceedings der 4. Österreichischen Artificial-Intelligence Tagung, Wiener Workshop Wissensbasierte Sprachverarbeitung*, pages 11–19, Berlin, August. Springer.

Hans-Ulrich Krieger and Ulrich Schäfer. 1994. *TDL*–a type description language for constraint-based grammars. In *Proceedings of the 15th International Conference on Computational Linguistics, COLING-94*, Kyoto, Japan.

Miroslav Martinovic and Tomek Strzalkowski. 1992. Comparing two grammar-based generation-algorithms: A case study. pages 81–88, Newark, Delaware.

John Nerbonne. 1992. Constraint-based semantics. In Paul Dekker and Martin Stokhof, editors, *Proceedings of the 8 th Amsterdam Colloquium*, pages 425–444. Institute for Logic, Language and Computation. Also available as Research Report RR-92-18, Deutsches Forschungszentrum für Künstliche Intelligenz, Saarbrücken, Germany.

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*, pages 205–211., Pittsburgh.

Stuart M. Shieber, Gertjan van Noord, Robert C. Moore, and Fernando C. N. Pereira. 1990. A semantic-head-driven generation algorithm for unification-based formalism. *Computational Linguistics*, 16(1):30–42.

Stuart M. Shieber. 1986. *An Introduction to Unification-Based Approaches to Grammar*, volume 4 of *CSLI Lecture Notes.* Stanford University, Stanford (CA).

Stuart M. Shieber. 1988. A uniform architecture for parsing and generation. In *Proceedings of the 12th International Conference on Computational Linguistics and the 24th Annual Meeting of the Association for Computational Linguistics*, pages 614–619, Budapest, Hungary, August 22-27.

Tomek Strzalkowski. 1994. A general computational method for grammar inversion. In Tomek Strzalkowski, editor, *Reversible Grammars in Natural Language Processing*, pages 175–200. Kluwer, Boston, Dordrecht, London.

Hans Uszkoreit, Rolf Backofen, Stephan Busemann, Abdel Kader Diagne, Elizabeth A. Hinkelman, Walter Kasper, Bernd Kiefer, Hans-Ulrich Krieger, Klaus Netter, Günter Neumann, Stephan Oepen, and Stephen P. Spackman. 1994. DISCO–An HPSG-based NLP System and its Application for Appointment Scheduling. In *Proceedings of the 15th International Conference on Computational Linguistics, COLING-94*, Kyoto, Japan.

Generating Air Quality Reports From Environmental Data

Stephan Busemann, Helmut Horacek*
German Research Center for Artificial Intelligence (DFKI GmbH)
Stuhlsatzenhausweg 3, 66123 Saarbrücken (Germany)
{busemann, horacek}@dfki.de

Abstract

This paper describes ongoing work on the generation of German and French air quality reports on the basis of up-to-date environmental measurements. This real-world application is characterized by a simple and small sublanguage. The system is called with a bundle of user requests entered through a hyper-link navigator. For text planning, a schema-based component produces domain-specific semantic content representations that are fed to the TG/2 production system [Busemann, 1996] for linguistic realization. The semantics interface between the two components is tailored to the task and domain at hand. It is independent from the particular language chosen. It is argued that these design decisions have important practical benefits over more general, linguistic approaches. The texts produced are designed for administrative use. A version for the general public is foreseen as well.

1 The application scenario

This paper describes ongoing work on the generation of German and French air quality reports on the basis of regularly updated environmental measurements. Such data is made available on a server under development for TEMSIS (Transnational Environmental Management Support and Information System). It includes the pollutant, the measurements, the location and the time the measurements were taken, and thresholds that may cause some activity if overstepped. Besides such data, the server provides meta data that allow for descriptions of the measuring locations, of the pollutants measured and of regulations or laws according to which a comparison between measurements and thresholds can be performed.

With TEMSIS, an environmental information system is designed and implemented as part of a transnational cooperation between the communities in the German-French urban agglomeration, Moselle Est and Stadtverband Saarbrücken. Networked information kiosks will be installed in a number of communities to provide public and expert environmental information.

The timely availability of relevant information about the current environmental situation improves the planning and reactive capabilities of the administration considerably. The summarization of information in natural languages saves time by reducing the need of looking up heterogeneous data on the server. The domain of air quality reports is especially promising in this respect since the underlying data are relatively complex and highly structured. The generated texts can be complemented with graphical presentations of the development of measurements over time or comparisons with earlier periods. The generated texts can be edited by the administration to fit additional needs.

The generation system has interfaces to the server (access to data and meta data) and to the navigator, through which the user selects his request from a hierarchy of options.

*This work was funded by the European Union within the TEMSIS project (Telematics Applications C9, no. 2945).

Moreover, an interface to a diagram graphics generator is foreseen. The results are presented on the Web as an HTML document. The generation system must fulfill two tasks:

Text structuring: A schema-based component generates the text structure on the basis of the user's request. It combines fixed text blocks with dynamic text in an as language-neutral way as possible.

Surface realization: The production system TG/2 [Busemann, 1996] is reused for language-specific processing. It will operate with grammars partly tuned towards the domain and task requirements.

In addition, in order to be cooperative and helpful, the system must exhibit robustness in all its parts. Any of the interfaces in the generator may provide unexpected input. For instance, the number of available measurements may be insufficient to fit a schema, or the user request may be ill-formed. Nevertheless the generator has to capture such failures and produce adequate meta-level comments.

2 Text structuring using real world data

The overall task of the text planning component in TEMSIS is the production of an intermediate structure of an air pollution report suiting a small set of user specifications. These specifications determine a report structure and access paths to the concrete data to be included in the report. The report structure is taken out of a small set of pre-defined structures, which were defined on the basis of analyses carried out by domain experts in Germany and France. Each report consists of a set of assertions whose composition is obtained in varying degrees of cannedness (numbers refer to the sample report in Figure 1):

- Canned texts taken from the database (2), (6); these assertions constitute descriptions of the major domain concepts involved. Their inclusion into the report is optional.
- Freely generated, but data-independent assertions (1), (3); these assertions represent confirmations of user parameters. Their inclusion into the report is optional, too.
- Freely generated, data-dependent assertions (4), (5), (7); these assertions constitute presentations of stored or derived data, selected from the database in accordance with the user's specifications. Moreover, even the structure of these assertions depends to some extent on the database content; in case the data stored are considered insufficient for a reliable statement about the requested data, a suitable qualifying statement about the requested information is added. This is one of the places where cooperativeness meets robustness.

In addition, vectors of data are depicted as diagrams in some report types. Diagrams will not be produced by our system, but inserted into the text if appropriate at locations defined through the text structure. If a graphics component is not available, the information to be diagrammed can alternatively be presented as a formatted table.

In more technical terms, major tasks in the text organization part of the TEMSIS system are the following:

(1) Zur Betrachtung der Luftbelastung im Winter 1996/97 haben Sie die Meßstation Völklingen-City ausgewählt. *(optional)*

[In order to inform yourself about the air pollution during winter 1996/97, you have chosen the measurement station of Völklingen-City.]

(2) Die Lage der Meßstation Völklingen-City kann wie folgt charakterisiert werden: Die Station liegt mitten in der Völklinger Innenstadt auf 220 Meter Meereshöhe. Gemessen wird in 4 m Höhe über dem Boden. Die Station ist von Gebäuden umgeben und liegt an einer stark befahrenen Straße. *(optional)*

[The location of the measurement station of Völklingen-City can be described as follows: It is located ... at 220m of altitude...]

(3) Sie wollen sich über die Konzentration von Schwefeldioxyd in der Luft informieren. *(optional)*

[You want to know the concentration of sulfur dioxide.]

(4) Im Winter 1996/97 wurde der MIK-Wert nach VDI-Richtlinie 2310 von 1000 $\mu g/m^3$ an der Meßstation Völklingen-City nicht erreicht. Der MIK-Wert für eine 24-stündige Einwirkungsdauer (300 $\mu g/m^3$) wurde dreimal überschritten.

[During the winter 1996/97, the MIK value according to VDI directive 2310 of 1000 $\mu g/m^3$ was not reached at the measurement station of Völklingen-City. The MIK value for an exposition of 24 hours (300 $\mu g/m^3$) was exceeded three times.]

(5) Im Winter 1995 wurde der MIK-Wert nicht erreicht. Der MIK-Wert für eine 24-stündige Einwirkungsdauer wurde einmal überschritten.

[During winter 1995/96, the MIK value was not reached. The MIK value for an exposition of 24 hours was exceeded once.]

(6) Schwefeldioxyd ist ein gasförmiger Schadstoff, der im wesentlichen durch die Verbrennung von Kohle, Heizöl und Gas bei der Hausheizung, Stromerzeugung und ähnlichen Produktionsprozessen entsteht. Er verteilt sich im allgemeinen zu einer gleichmäßigen Luftbelastung. Er gefährdet die menschliche Gesundheit. *(optional)*

[Sulfur dioxide is a gaseous pollutant... It is dangerous to human health.]

(7) Der Grenzwert für den Schadstoff Schwefeldioxyd liegt in der Bundesrepublik bei 30 $\mu g/m^3$ Luft für die Langzeitbetrachtung von Durchschnittswerten. Die Kurzzeitbelastung darf nicht höher als 3000 $\mu g/m^3$ liegen (nachzulesen in der TA Luft). *(optional)*

[The threshold value for the pollutant sulfur dioxide is, in Germany, at 30 $\mu g/m^3$ for a long-term observation of average values. The short-term exposition must not be higher than 3000 $\mu g/m^3$ (according to the technical directive "TA Luft").]

Figure 1: A sample target text. The user has chosen from the navigator menus the pollutant SO_2, the location Völklingen, and the period "winter season 1996". In addition, a description of threshold passings was preferred to one of absolute values.

- Report structures must be defined in such a way that their parameters (corresponding to user specifications), their ingredients in terms of assertion patterns, and the required database calls whose results should fill certain places in these patterns are associated to each other in a declarative, flexible, and easily maintainable manner. In particular, common specifications are shared across report types, and data preparation procedures take care of filling specification parameters and data obtained from the database into appropriate places of assertion patterns.

- The instantiation of report structures must be organized in a systematic way, which comprises the selection and the refinement of assertion patterns, depending on relevant database values, lexical material, and context.

- The assertion specifications must be manipulated according to the textual context. Temporal, local, and subject circumstances are not repeated in the presentation.

The second task is the most interesting one in the above list. Assertion specifications originally available in terms of the condensed user parameters ultimately have to be related to lexical specifications in both French and German, the target languages in TEMSIS. In order to achieve these transitions in a systematic way, distinct predicates are defined on three ontological levels, corresponding to:

1. user parameters,

2. conceptual representations,

3. language-neutral representations

There is an increasing degree of explicitness from level 1 to 3. Assume a certain threshold value is included in a report.

1. It is implicitly associated with certain combinations of report specifications, according to a deep analysis of the underlying relations.

2. Its representation on a conceptual level comprises a semantically rich predicate and a value.

3. The concept is expanded into a description at the language-neutral level, distinguishing the threshold from its justification (that is, the law by which it is introduced, and the time period determining its validity).

The techniques of mapping structures across representation levels works on the basis of a small set of compositional schemata, as described in detail in [Horacek, 1996]. The system's functionality within the domain of air quality reporting is well-defined and sufficiently limited for using more condensed mapping schemata than in [Horacek, 1996]. A deep analysis of the underlying relations would be unnecessarily time-consuming in our application.

A fourth ontological level is that of language-specific representation. Here the resulting description may be realized by different word groups in the target language (for instance, 'valeur limite autorisé' and 'gesetzlich zulässiger Grenzwert', respectively, none of the French words corresponding to a single German word and vice versa). This level is implemented through the TG/2 realizer.

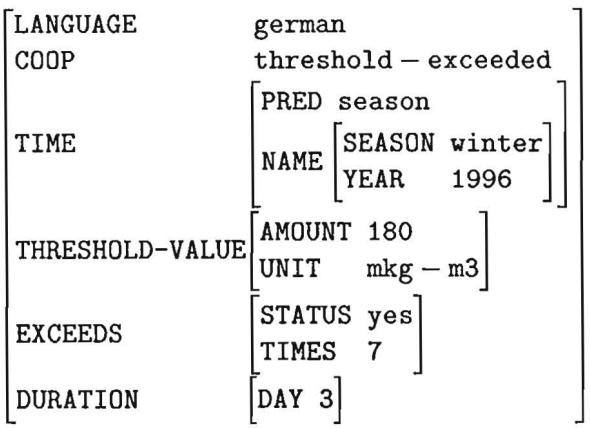


Figure 2: A sample TG/2 input representation for the German version of *During the winter 1996/97, the legally admissible threshold for a three-day assessment of 180 $\mu g/m^3$ was exceeded seven times.*

3 Language-specific realization with TG/2

The system TG/2 [Busemann, 1996, Wein, 1996] is a flexible and reusable, application-oriented text realization system that can be smoothly integrated with deep generation processes. It integrates canned text, templates, and context-free rules into a single production-rule formalism.

TG/2 is based on production system techniques [Davis and King, 1977] that preserve modularity of processing and linguistic knowledge, hence making the system transparent and reusable for various applications.

In the application at hand, the interface between the text planner and TG/2 consists of domain speechact representations. An example is shown in Figure 2. These representations are ignorant with respect to the differences between German and French. Basically, they express a speechact (COOP) combined with a set of domain roles. The roles express e.g. the pollutant, the type and the value of thresholds, actual measurements, information about the time and the place the measurements were taken etc.

The text planner uses this intermediate representation every time it feeds an assertion to TG/2. The planner knows which roles must be expressed for TG/2 to generate output. On the basis of contextual knowledge, it also knows which information should be left out since it is already known to the user. In utterance (4) of Figure 1, the pollutant need not be mentioned, since it was introduced in (3) already. Similarly, utterance (5) need not repeat many parts mentioned already in (4), e.g. details about the threshold, or the location. The text planner decides which types of informations are passed on for realization in TG/2.

Formally, the representations are encoded as feature structures. This allows for a very comfortable adaptation to a TG/2-internal format through the unification with a predefined structure that expresses the necessary mappings as coreferences. The internal representation is looked up by the grammar interpreter in order to determine which production rules can be applied, and what information must be realized into natural language.

Grammar rules are designed as productions (cf. Figure 3). They are encoded in the

```
(defproduction wertueberschreitung "WU06"
  (:PRECOND (:CAT DECL
             :TEST ((pred-eq 'threshold-exceeded)
                    (not (threshold-type-p))))
   :ACTIONS (:TEMPLATE  "Der gesetzlich zulaessige Grenzwert von "
                        (:RULE VAL (get-param 'threshold-value))
                        (:OPTRULE POLL (get-param 'pollutant))
                        "wurde "
                        (:OPTRULE PPtime (get-param 'time))
                        (:RULE DUR (get-param 'duration))
                        (:OPTRULE SITE (get-param 'site))
                        (:RULE EXCEEDS (get-param 'exceeds))
                        ".")))
```

Figure 3: A TGL rule for German encoding a template to be used for input as shown in Figure 2. Information about the threshold value, the duration and about threshold violation are mandatory; all other slots are optional. The function `get-param` extracts the relevant information from the translated input representation.

language TGL [Busemann, 1996]. A rule is applicable, if its preconditions are met. The rule in Figure 3 is applicable to input material as shown in Figure 2, because the `COOP` slot matches, and there is no information about the threshold type (such information would lead to a different sentence pattern). TGL rule development in previous applications showed that it is possible to separate general, linguistic rules from specific ones, thus allowing the general portions to be reused in other applications.[1] The use of different levels of abstraction from underlying message information (canned text, templates, context-free grammars) allows the grammar writer to model general, reusable linguistic knowledge as well as more specific task and domain-oriented wordings. In particular, standardized linguistic realizations of typical situations can be directly encoded into the grammar as canned text.

By associating canned text with domain speech acts, TG/2 behaves in a domain and task specific way. The loss of flexibility in the wording, which the text planner cannot influence, is hardly a problem in technical documents. However, repetition of known information can be avoided. The possibility of omission is reflected in the grammar through the notion of optional rule applications (`OPTRULE`, cf. Figure 3). Optional rules are ignored if the input structure does not contain relevant information. In the domain and for the task at hand, it was possible to design the text templates in such a way that this compositional approach leads to fluent text.

If alternative formulations for some message are encoded in the TGL grammar, they can be ordered according to a set of preference criteria that cause the system to prefer certain formulations over others. Grammar rules leading to preferred formulations are selected first from a conflict set of concurring rules. This way, different formulations can be generated as well as texts of different length. The preference mechanisms will be used to tailor texts for administrative and public uses, respectively.

[1] For instance, a sub-grammar describing dates in the domain of appointment scheduling [Busemann *et al.*, 1997] could be reused here with minor extensions.

4 Conclusion and Future Work

We described ongoing work on the generation of German and French air quality reports on the basis of up-to-date environmental measurements. A prototype implementation has been achieved and is being tested with real data. The texts are generated in either German or French. They are presented as HTML documents (alternatively, LaTeX or plain ASCII text is provided on demand).

We claimed that a simple, compositional approach to text structuring and realization be sufficient for this domain. Obviously there is a large class of relatively simple NLG applications that can be captured by the approach presented in this paper. However, other applications may require some interdependency between the text planner and the realizer, thus calling for a more elaborate and flexible interface including e.g. logical forms for the utterances. While such representations can be handled within TG/2, as was shown in [Busemann, 1996], they require much more effort for grammar development and extension than the flat ones adopted here.

Future work will place particular emphasis on the application-oriented design of the interface between the text structuring component and the realizer. We believe that gathering experience with real applications is a good starting point for determining relations between requirements imposed by the applications and the level of abstraction chosen for the intermediate representations. As a result we will be able to tailor systems better according to the complexity of NLG applications.

References

[Busemann *et al.*, 1997] Stephan Busemann, Thierry Declerck, Abdel Kader Diagne, Luca Dini, Judith Klein, and Sven Schmeier. Natural language dialogue service for appointment scheduling agents. In *Proc. 5th Conference on Applied Natural Language Processing*, pages 25–32, Washington, DC., 1997.

[Busemann, 1996] Stephan Busemann. Best-first surface realization. In Donia Scott, editor, *Eighth International Natural Language Generation Workshop. Proceedings*, Herstmonceux, Univ. of Brighton, England, 1996. Also available as Research Report RR-96-05, Deutsches Forschungszentrum für Künstliche Intelligenz, Saarbrücken, Germany.

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

[Horacek, 1996] Helmut Horacek. Lexical choice in expressing metonymic relations in multiple language. *Machine Translation*, (11):109–158, 1996.

[Wein, 1996] Michael Wein. Eine parametrisierbare Generierungskomponente mit generischem Backtracking. Master's thesis, Department for Computer Science, University of the Saarland, 1996.

Nonmonotonic Aspects of Incremental Natural Language Production: Performing Self–Corrections in a Situated Generator

Wolfgang Finkler
German Research Center for Artificial Intelligence, Saarbrücken

Abstract

In an incremental generation system the production of output starts before the complete input specification to the generation system is known. That runs the risk of being forced to perform a self–correction. We present an interdisciplinary approach to generation that deals with nonmonotonic aspects of incremental processing by utilizing a reason maintenance system. For the first time, a generator has been presented which can realize typical self–corrections, and is able to create declarative representations for the generated spontaneous utterances. The results of an extensive corpus analysis of self–corrections are a basis for the implemented system *PERFECTION*.

1 Motivation

The effective use of automatically produced spoken language output in a user interface requires the observation of a number of constraints on the design and realization of a natural language generator. Besides general aspects such as tailoring output to specific users, producing output that is well–formed and adequate with respect to the contents, there is a need to pay attention to temporal factors of the generation process. Research results in the field of Human–Computer interaction show that speech systems are subject to strong real–time constraints. Speech output of a system demands the human addressee to preserve steady attention in order to avoid missing some part of the transitory output. Therefore, the delay of the output shouldn't be too long. Furthermore, a dialogue partner might utilize a possibility for turn–taking, when there is an initial delay in a dialog contribution that differs too much from typical delays.

2 Incremental Processing during Natural Language Production

In order to fulfill these real–time constraints on Human–Computer interaction we suggest the utilization of the incremental processing mode in a user interface with spoken language output being highly relevant from a practical perspective. Hereby, the production of output starts before the complete input to the generation system is known. A prompt system reaction may result that provides immediate feedback to elements of users' input. A user may exert influence on a computation while perceiving fragments of the system output. Such tight coupling of applications and their usage has been envisaged for intelligent user interfaces [Marcus and van Dam, 1991]. In addition, applications such as simultaneous interpretation of natural language or reporting about ongoing events typically utilize the incremental processing mode. From a theoretical point of view, the incremental processing mode is part of psychologically plausible models of human language production (see, e.g., [Kempen, 1978], [Levelt, 1989]).

2.1 Incremental Output and the Need for Self–Corrections

Gains in the responsive nature of a system cannot be expected to be obtainable without running a risk. During incremental generation the computation as well as the articulation of the beginning of an utterance are performed in spite of temporarily incomplete input data. If further input increments specify unexpected additional data or even demand the modification or deletion of previously specified data, revisions in the computation have to be dealt with [De Smedt and Kempen, 1987], [De Smedt, 1991]. Eventually, parts of the already articulated system output are affected. They have to be identified and repaired in an adequate continuation of the utterance [Finkler, 1996].

We present aspects of an implemented approach to incremental syntactic generation that goes well beyond previously published incremental generators (see, e.g.,

MUMBLE by [Meteer *et al.*, 1987], IPG by [Kempen and Hoenkamp, 1987], POPEL–HOW by [Finkler and Neumann, 1989] and [Reithinger, 1991], IPF by [De Smedt, 1990], the generation component in ΦDM DIALOG by [Kitano, 1990], FIG by [Ward, 1994], SYNPHONICS–Formulator by [Abb *et al.*, 1993], and TAG–GEN by [Kilger and Finkler, 1995]). The additional capability of our system is related to a sophisticated realization of self–corrections during incremental output production. Special emphasis was laid on the treatment of nonmonotonic cases of the incremental processing mode and on situated aspects of language production. A natural language generation system modifies its environment by means of spoken language output. Speech output cannot be withdrawn in the same way as is possible for internal results of computations of a system. It is utilized as a situational influence on the further processing of the generator. These observations are of specific interest to tasks that necessitate modifications of the current output because of nonmonotonic input specifications. The generator is able to perform self–corrections of the already produced output by means of computing an adequate moment of interruption, editing terms, and the continuation of the utterance. Instead of merely realizing a technical repair such as a complete restart, it may simulate and represent certain types of self–corrections as they typically occur in human language production.[1]

2.2 A corpus analysis as the empirical basis for automatic production of self–corrections

In order to broaden the knowledge of the shape of typical self–corrections in dialog situations, an extensive corpus analysis has been realized for a selection of collected speech data of the *VERBMOBIL*–project. In that effort, 1251 self–corrections have been identified in a set of acoustic data of approximately 8 hours duration containing 4590 dialog turns. Thereby, hypotheses of psycholinguistic models about the shape and the course of self–corrections have been checked and improved by means of additional qualifications ([Levelt, 1983], [Kempen, 1991], [Finkler, 1996]). The results of the corpus analysis have been utilized to guide the production of self–corrections in the proposed model for generation. Utterance 1 illustrates a simple case of a self–correction.

Utterance 1:
... eine schöne Auswahl +/von verschiedenen Progra=/+ <"ah> von verschiedenen <!1 verschiedne> Speisen .
(... a good selection +/of diverse progra=/+ <uh> of diverse dishes .)

[VERBMOBIL CD 1, Dialog N004K.TRL:MM4010]

This utterance shows several of the properties that seem to be typical for self–corrections in spoken language use for languages such as Dutch, German, and English according to research results of Psycholinguistics and related disciplines (see, e.g., [Levelt, 1983], [Finkler, 1996], [Carletta *et al.*, 1993]). These properties are related to the moment of interruption, the location of a within–word interruption, the editing phase and the continuation of the flow of speech. Concerning specific phenomena that can be identified in Utterance 1 our analysis of self–corrections of the exchange type (N = 405) revealed the following distributions in our corpus:

- There is an immediate interruption of the original utterance at the reparandum element 'Progra'. In our corpus the original utterance typically is interrupted not later than three syllables after the reparandum (in 89.81 % of all cases).
- There is a within–word interruption at the reparandum. In our corpus within–word interruptions at the reparandum take place in 40.22 % of all cases of immediate interruptions of the original utterance at the reparandum position. Within–word interruptions inside non–reparandum words take place in 22.54 % of all cases of delayed interruptions.
- The hesitation element 'äh' has been used in Utterance 1 in the editing phase. In our corpus there is a surprisingly high frequency of unfilled pauses in the editing phase (73.09 %). However, if the editing phase contains a hesitation element the vocal articulation 'äh' is the most common one. It is used in 88 % of those cases.
- The utterance continues by restarting at a phrase boundary and by repeating elements that have been articulated before the reparandum, i.e., by means of an anticipatory retracing. In our corpus there is a restart at a phrase boundary in 90.72 % of all cases of immediate interruptions at the reparandum. When a noun is used as reparans, the utterance typically continues by means of an anticipatory retracing (66.23 %) in contrast to the usage of a direct replacement.

[1] For our applications it is not desirable to simulate phenomena such as stuttering and speech errors and their corrections. Instead, we have tried to mimic phenomena of human language performance that are compatible with problems caused by temporal and nonmonotonic aspects of incremental processing.

3 Some Aspects of Situated Generation

Instead of presenting more details of a descriptive view of self–corrections we illustrate how they may be constructed by our incremental generator. We conceive the process of incremental natural language generation as a situated activity. Thereby, we take into account the fact that speech output of a generation system changes the environment of the system. The process of speech production should pay attention to the situational influences of the already produced incremental output. That view of a generation system discloses a particular connection to planning systems within which planning and execution take place in an interleaved fashion and that allow for reactive behavior. There is an obvious correspondence between a robot that has to patch its partially executed plan after having obtained information about its unexpectedly modified environment and a generator that tries to repair its partially produced and articulated utterance after having identified the inappropriateness of parts of its output with respect to the current situation.

Our approach also resembles certain aspects of the reactive approach to explanation as introduced in [Moore, 1989]. A system's output is to be continued according to input that has been provided on the fly during the natural language generation process. In contrast to [Moore, 1989] who focuses on the what–to–say part of natural language generation, we have developed a syntactic generator for the how–to–say and when–to–say tasks of natural language generation that may process and output utterances of an increment size below the clause level.

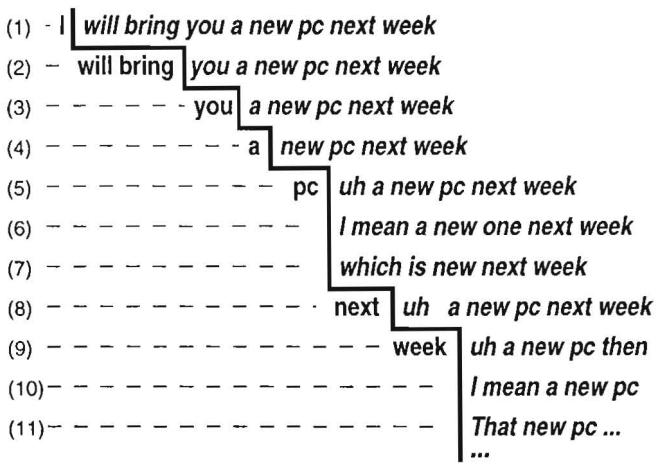


Figure 1: A variety of spoken language utterances for differing specification times of an optional input element

Figure 1 illustrates some of the results of alternative techniques that might be used for continuing an utterance in a situated generation process. The borderline that separates Figure 1 into a left and a right part indicates the current output position of the generator for each of the 11 rows when that component is forced to integrate the modifying element 'new' into its ongoing verbalization. The resulting utterance of the rows (1) to (4) doesn't contain a self-correction, since the optional element was known in due time. The rows numbered (5) to (10) illustrate several self–corrections that are caused by a late insertion of the modifying element. There are several options to avoid redundancy in the continuation of the utterance (see rows (6) and (9)). Row (7) shows a kind of a hidden repair by appending a relative clause [Finkler and Schauder, 1992]. Note, that this option is no longer available, when the next word has been articulated (see row (8)). There exist various possibilities of metacomments to integrate the verbalization of the modifier into a further utterance (e.g., see row (10)). Row numbered (11) illustrates a strategy that hides the self–correction in an elegant way.

We have developed a classification of structural properties of overt utterances in relation to the reparandum. The resulting situation classes are used in order to distinguish relevant situations and to guide the selection of an adequate continuation strategy. We give two examples: One situation class comprises cases where the already produced output is unrelated to the reparandum (rows (1) to (3) in Figure 1)). Another of the 12 defined situation classes specifies that the reparandum has been completely articulated and that the current output position is located inside the following constituent. That condition is fulfilled for row number 8 in Figure 1. The situation classes are used in the editor component of the generator as described in Section 5.1.

4 Design Principles for Incremental Syntactic Generation

We introduce several prominent design principles for incremental syntactic generation that have been obeyed in the realization of our system. Requirements on the generator that arise from dealing with incremental input are described as briefly as possible since they have been discussed for several incremental generators. First of all, exploiting parallelism might be useful during incremental processing since a suitable segmentation of the underlying representation structure is a prerequisite for the use of an incremental processing mode. The syntactic representation formalism should allow for flexible expansion operations because input increments may arrive in an arbitrary order. The generation process should be lexically guided and the representation as well as the processing of hierarchical and positional constraints of the grammar should be separated. These design principles have already been realized in [Finkler and Neumann, 1989], [De Smedt, 1990], and [Kilger and Finkler, 1995].

Some important design principles that are related to incremental output production are as follows:

1. The processes for syntactic generation, i.e., selecting, constructing and linearizing syntactic structures at one hand and output production, i.e., deliberating how to compose chunks of inner speech to be sent to the synthesis component at the other hand should be decoupled. That facilitates a natural timing of articulation since pauses may be utilized at utterance positions that are not restricted by the shape of atomic building blocks of the syntactic generation component. We realized an output manager that may buffer some already computed inner speech before feeding the speech synthesis component and that updates a representation of the current output position of overt speech.

2. There should be a controlling device for triggering self–corrections and for synchronizing their computation with the ongoing production process. Global information about the effects of decisions during incremental generation and about the current state of input consumption and output production provide the basis for guiding the generation of adequate self–corrections. We have conceived a production oriented approach to monitoring that differs from the perception–oriented approach as described by [Levelt, 1989]. By means of a direct access to internal representations of the syntactic generator the controlling device is able to identify affected structures when decisions are to be revised.

3. There should be a declarative representation for all generated utterances. In particular, that holds for utterances containing automatically produced self–corrections. Such an approach allows for a uniform processing in cases where repeated self–corrections have to be produced and integrated into one utterance. In our approach, syntactic generation makes use of a lexicalized unification–based grammar. The grammar encodes elementary syntactic structures that typically occur in self–corrections as well as elementary syntactic structures that are used in the normal case, i.e. in utterances without repairs. The grammar is organized in two layers in order to separate both types of grammar rules. Constructions of the second layer — for self–corrections — observe results of psycholinguistic studies of self–corrections and of our corpus analysis. The resulting utterances are similar to coordination constructions (see [Levelt, 1983], [Kempen, 1991]). Figure 2 shows a representation of a self–correction in our variant of Tree Adjoining Grammar for the utterance "I will bring you a pc uh a new pc".[2]

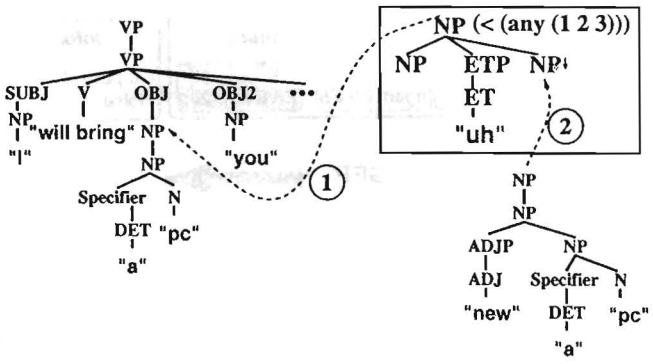


Figure 2: Representing constructions of spontaneous speech in a declarative grammar

The auxiliary structure enclosed in a rectangular box belongs to the second layer of the grammar. It connects the original utterance with the continuation of the self–correction. After having performed operations of adjunction and substitution (as indicated by the numbered arrows) the complete dominance structure is represented.

5 Performing Self–Corrections during Incremental Generation

Figure 3 illustrates both a functional view of the generation system and aspects of data–flow and control–flow between the submodules of our component.

Input data to be verbalized may be specified in a random order. They encode entities, i.e., content words and semantic relations between them. As exemplified by the rightmost input increment in Figure 3 there may be nonmonotonic input specifications (CE represents: exChange-Entity). The example specification forces the exchange of the filler of the agent role. When the system's output has gone beyond a point that is to be modified, a self–correction is performed as indicated in the lower part of Figure 3.

We present a sketch of the computation inside the syntactic generator. The component consists of four modules working in parallel: Input manager and output manager as the interface modules consume input data and provide incremental output of the system. Monitor and Editor are used as controlling devices during the production of self–corrections (see Section 5.1). The processing of input data is performed in a distributed parallel model by a set of cooperating objects which are dynamically created.[3] Each of these objects runs its own program, selects syntactic structures for verbalizing its input data and communicates with other objects

[2]Note, that we have separated both encoding and processing of synctactic constraints for dominance and linear precedence relations in our grammar. In particular, the illustrated syntactic structures are not tree structures. Their linear order is interpreted on the basis of context–dependent linearization rules.

[3]That aspect is similar to the use of parallelism in IPF [De Smedt, 1990] and POPEL [Finkler and Neumann, 1989].

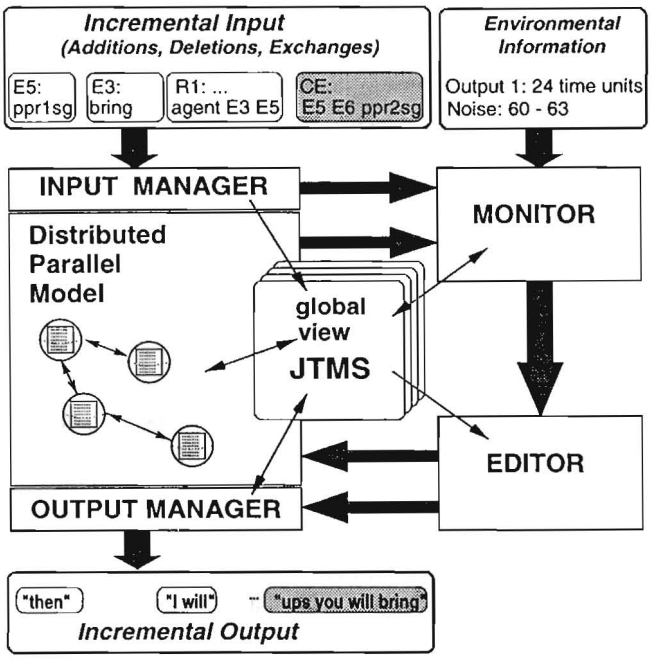


Figure 3: Architecture and functional view of the generation model

to combine syntactic structures. We utilize operations for distributed adjunction and substitution in a variant of tree adjoining grammars as described in [Kilger, 1994]. Linearization is performed in an interplay between the output manager and individual objects. The produced inner speech as well as overt speech are controlled by the output manager. The input manager as well as the objects deliver information about selected representations and partial results to a reason maintenance system.[4] That system serves as device for book-keeping of dependencies among choices. Monitor and Editor realize a production–oriented approach to control the syntactic generation.

5.1 Controlling Self–Corrections

The Monitor repeatedly checks for fulfillment of conditions to trigger self–corrections. Possible causes of self–corrections are unexpected input specifications such as optional elements that are given too late, nonmonotonic input specifications, or local problems of objects during combination or linearization of syntactic structures. Furthermore, information about superposition of environmental noise might be used to trigger a self–correction of the 'repetition' type. The Editor performs a temporary global management of the generator after having received an alert of the Monitor. It is responsible for interrupting the original utterance and evaluates the internal state and the situation class of the current output in order to determine the strategy for continuing the utterance. The editor interrupts the computation of all objects for a moment in order to prevent them from modifying the system state before the editing phase and the continuation of the self–correction have been initialized properly. After that, the processing of the self–correction is done by a set of new objects that have been created by the Editor.

[4] In our system we use the original implementation of the monotonic JTMS of [Forbus and De Kleer, 1993].

5.2 Dealing with Nonmonotonic Aspects of Incremental Generation

In general, an incremental system has to deal with nonmonotonic aspects of processing. It is not uncommon that there is a need for withdrawal of decisions and for retraction of assumptions. Reason maintenance systems are useful in such applications. We have utilized a JTMS in our incremental generator and solve the encoding problem in a way that differs from previous usage of RMS in systems for NL analysis (e.g. [Wirén, 1992], [Zernik and Brown, 1988]) and generation (e.g. [Inui *et al.*, 1992]). None of these systems explicitly mentions the case of having forwarded output to a next component that cannot be kept as turns out later. Instead, all revisions to be handled by utilizing an RMS are local tasks inside the NL component.

We have dealt with a stronger constraint. More than revising internal decisions by utilizing dependency-directed backtracking there is a need for a kind of 'unspeaking', i.e., repairing by means of continuing. Therefore, whenever there are nodes in the dependency network which represent parts of overt speech and which obtain a labelling of 'OUT', the problem solver coupled to the RMS, i.e. the Editor in our generator interprets those data as being related to the reparandum in a self–correction. That information is used to determine the situation class as indicated in the previous section.

6 Conclusions

We presented an advanced model for incremental syntactic generation of natural language. The current environment has to be considered when producing spoken language output. The implemented system *PERFECTION* is able to cope with nonmonotonic aspects of the incremental processing mode and thereby may simulate self–corrections of the already produced output as in human language production. There is a declarative representation for all generated utterances.

References

[Abb *et al.*, 1993] B. Abb, M. Herweg, and K. Lebeth. The incremental generation of passive sentences. In

Proc. 6th EACL, pages 3–11, Utrecht, The Netherlands, 1993.

[Carletta *et al.*, 1993] J. Carletta, R. Caley, and S. Isard. A collection of self-repairs from the map task corpus. Technical Report HCRC/TR-47, Human Communication Research Centre, University of Edinburgh, Edinburgh, Scotland, 1993.

[De Smedt and Kempen, 1987] K. De Smedt and G. Kempen. Incremental sentence production, self–correction and coordination. In G. Kempen, editor, *Natural Language Generation*, pages 365–376. Martinus Nijhoff, Dordrecht, 1987.

[De Smedt, 1990] K. De Smedt. *Incremental Sentence Generation: a Computer Model of Grammatical Encoding.* PhD thesis, Nijmegen Institute for Cognition Research and Information Technology, Nijmegen, the Netherlands, 1990. NICI TR No 90-01.

[De Smedt, 1991] K. De Smedt. Revisions during generation using non–destructive unification. In *Proc. Third European Workshop on Natural Language Generation*, pages 63–70, Judenstein, Austria, 1991.

[Finkler and Neumann, 1989] W. Finkler and G. Neumann. Popel-how – a distributed parallel model for incremental natural language production with feedback. In *Proc. 11th IJCAI*, pages 1518–1523, Detroit, MI, 1989.

[Finkler and Schauder, 1992] W. Finkler and A. Schauder. Effects of incremental output on incremental natural language generation. In *Proc. 10th ECAI*, pages 505–507, Vienna, Austria, 1992.

[Finkler, 1996] W. Finkler. *Automatische Selbstkorrektur bei der inkrementellen Generierung gesprochener Sprache unter Realzeitbedingungen: Ein empirisch–simulativer Ansatz unter Verwendung eines Begründungsverwaltungssystems.* PhD thesis, Technische Fakultät, Universität Saarbrücken, Saarbrücken, Germany, november 1996.

[Forbus and De Kleer, 1993] K.D. Forbus and J. De Kleer. *Building Problem Solvers.* The MIT Press, Cambridge, MA, 1993.

[Inui *et al.*, 1992] K. Inui, T. Tokunaga, and H. Tanaka. Text revision: A model and its implementation. In R. Dale, E.H. Hovy, D. Rösner, and O. Stock, editors, *Aspects of Automated Natural Language Generation*, pages 215–230. Springer, 1992.

[Kempen and Hoenkamp, 1987] G. Kempen and E. Hoenkamp. An incremental procedural grammar for sentence formulation. *Cognitive Science*, 11(2):201–258, 1987.

[Kempen, 1978] G. Kempen. Sentence construction by a psychologically plausible formulator. In R. Campbell and P. Smith, editors, *Recent Advances in the Psychology of Language, Volume 2: Formal and Experimental Approaches.* Plenum Press, New York, NY, 1978.

[Kempen, 1991] G. Kempen. Conjunction reduction and gapping in clause–level coordination: An inheritance–based approach. *Computational Intelligence*, 7:357–360, 1991.

[Kilger and Finkler, 1995] A. Kilger and W. Finkler. Incremental generation for real–time applications. Research Report RR-95-11, German Research Center for AI, Saarbrücken, Germany, 1995.

[Kilger, 1994] A. Kilger. Using utags for incremental and parallel generation. *Computational Intelligence*, 10(4):591–603, 1994.

[Kitano, 1990] H. Kitano. Incremental sentence production with a parallel marker–passing algorithm. In *Proc. 13th COLING*, volume 2, pages 217–221, Helsinki, Finland, 1990.

[Levelt, 1983] W.J.M. Levelt. Monitoring and self–repair in speech. *Cognition*, 14:41–104, 1983.

[Levelt, 1989] W.J.M. Levelt. *Speaking: From Intention to Articulation.* The MIT Press, Cambridge, MA, 1989.

[Marcus and van Dam, 1991] A. Marcus and A. van Dam. User–interface developments for the nineties. *Computer*, 24(9):49–57, 1991.

[Meteer *et al.*, 1987] M.W. Meteer, D.D. McDonald, S.D. Anderson, D. Forster, L.S. Gay, A.K. Huettner, and P. Sibun. MUMBLE–86: Design and Implementation. COINS Technical Report 87-87, University of Massachusetts, 1987.

[Moore, 1989] J.D. Moore. *A Reactive Approach to Explanation in Expert and Advice–Giving Systems.* PhD thesis, University of California, Los Angeles, CA, 1989.

[Reithinger, 1991] N. Reithinger. Popel– an incremental and parallel natural language generation system. In *Natural Language Generation in Artificial Intelligence and Computational Linguistics*, pages 179–199. Kluwer, Norwell, MA, 1991.

[Ward, 1994] N. Ward. *A Connectionist Language Generator.* Ablex, Norwood, NJ, 1994.

[Wirén, 1992] M. Wirén. *Studies in Incremental Natural–Language Analysis.* PhD thesis, Department of Computer and Information Science, Linköping, Sweden, 1992. Dissertation No. 292.

[Zernik and Brown, 1988] U. Zernik and A. Brown. Default reasoning in natural language processing. In *Proc. 12th COLING*, pages 801–805, Budapest, Hungary, 1988.

Structural Changes in Natural Language Generation

Helmut Horacek
Universität des Saarlandes
FB 14 Informatik
D-66041 Saarbrücken, Deutschland
horacek@cs.uni-sb.de

Abstract

In many domains and applications, an adequate and systematically developed natural language presentation of formally represented data is hindered by crucial discrepancies in the associated representations: formal specifications respectively raw data from which 'natural' texts are intended to be produced may differ fundamentally from dedicated natural language representations in both ontological and structural terms. Existing techniques in natural language generation and, even more, application-oriented systems take this fact into account in an insufficient manner only, and systematic approaches that address this issue in a sufficiently broad way are rare. Motivated by these shortcomings, we describe three methods that enable bridging structural as well as ontological differences between the underlying representations in one or another way. These methods pursue widely complementary goals to bridge differences in degrees of explicitness, detail, and perspective, and they apply distinct mechanisms, including inference rules, terminological logic, and pattern-based equivalence definitions. A suitable selection and combination of these methods that is oriented on the demands of a specific application should enable one to build a system with improved presentation capabilities.

1 Introduction

In many domains and applications, an adequate and systematically developed natural language presentation of formally represented data is hindered by crucial discrepancies in the associated representations: formal specifications respectively raw data from which 'natural' texts are intended to be produced may differ fundamentally from dedicated natural language representations in both ontological and structural terms. The demands of information storage, such as in databases, and the demands of problem solving, such as in knowledge-based systems significantly differ from presentation demands in natural language, and these differences manifest themselves in the underlying representations, too. The major differences are:

- *Varying degrees of explicitness* – formal representations typically are as detailed and explicit as needed for application purposes, while natural language representations must take into account that texts frequently leave a number of things implicit, thereby relying on the inferential capabilities of the audience.
- *Varying degrees in granularity* – formal representations typically rely on a particular level of granularity in terminological representations, according to application needs, while natural language representations must support the production of texts on a variety of granularity levels, meeting the demands of the audience.

- *Varying perspectives* – formal representations typically appear in a neutral or some application-oriented perspective, while natural language representations must enable textual presentations from largely varying perspectives, thereby meeting the demands of a given situation.

Existing techniques in natural language generation and, even more, application-oriented systems, take into account these discrepancies in an insufficient manner only, and systematic approaches that address this issue in a sufficiently broad way are rare. They do not abstract far enough from the original specifications, which may cause a variety of deficits in the texts produced. Motivated by these deficits, we intend to support the production of a communicatively adequate presentation by three methods that enable bridging structural as well as ontological differences between the underlying representations in one or another way. These methods pursue complementary goals, and they apply the following distinct mechanisms:

- the selection of a suitable ontological granularity in technical descriptions, which is realized by terminological transformations,
- the implicit conveyance of contextually inferable information, which is realized by incorporating inference rules into an RST-based text planner, and
- a flexible and rich lexicalization method, which is realized by elementary and compositional mapping schemata.

In the following, we briefly summarize each of these methods, which are described in detail in two chapters in our 'Habilitationsschrift', and in two long and recent journal papers, respectively. We conclude this paper by some considerations about how these methods can be meaningfully combined and applied in view of concrete demands and dedicated, that is, restricted coverage of a particular application.

2 Terminological transformations

Adequately presenting the results obtained by a database access or by the inference component of a knowledge-based system to the user of that system requires, among others, adapting the terminology used to the particularities indicated by a user profile. A special task in this presentation issue lies in explaining specific terms which might be unknown to the user, or whose precise meaning in the context of the domain and in view of the system's command of domain knowledge the user might be unaware of.

In order to support this task, an intermediate representation level mediating between representations oriented on storage or problem-solving purposes and those oriented on natural language purposes is built from which conceptual and lexical representations can be built more systematically, and to which these representations can be transformed easier (following [Horacek, 1996b]). Guidelines for the adequate design of this intermediate representation level include principles of conciseness, explicitness, and uniformity. The intermediate representation level, in fact, comprises special forms of conceptual representations.

Terminological transformations being applied on that representation level serve the purpose of reexpressing the meaning associated with individual representation elements (*Concepts* and *Roles*) in more explicit terms. By building hierarchically organized *Concept* definitions, such alternatives are available for *Concepts*. For *Roles*, the special definitions introduced in

the course of a refinement task in domain modeling provide for these alternatives (see below for more details).

Terminologically equivalent expressions (more compact or more detailed ones) can be obtained by applying some kind of elementary operations to individual representation elements, which can be repeated in a recursive way. These operations comprise:

- *Expanding* a conceptual description by replacing subexpressions according to terminological definitions, which express explicit definitions for certain aspects of the terms to be replaced.
- *Contracting* a conceptual description by replacing subexpressions according to terminological definitions, which express explicitly how a subexpression to be replaced is subsumed by a particular term.

These operations can be applied to *Concepts* and to *Roles*. Hence, there are four terminological transformation procedures: *expanding* a *Concept* definition, *contracting* a *Concept* definition, *expanding* a *Role* definition, and *contracting* a *Role* definition. Unlike for *Concepts*, terminological transformations are essentially a new task when being applied to *Roles*, especially in the direction of contraction.

Expanding *Concept* definitions is done by replacing selected *Concept* predications in a conceptual description by other *Concept* predications that constitute generalizations of the *Concepts* appearing in the original description. In order to maintain terminological equivalence, the more specific meaning attributed to the *Concepts* to be replaced must be reexpressed explicitly by adding appropriate descriptions to the newly introduced *Concept* predications. These descriptions comprise *Role* definitions attached to the specialized *Concepts*, but not to those *Concepts* replacing them, and more specific *Role* fillers which express value or cardinality restrictions that contribute only to the meaning of the *Concepts* to be replaced. Contracting *Concept* definitions reverses this operation. By applying these operations, the conceptual expression 'a female student' can be transformed into a terminologically equivalent expression 'a woman who goes to the university' which expresses the same information content in a structurally different way.

As for *Concepts*, it is also possible to replace the appearances of certain *Roles* in logical formulas. This is the case for *Roles* which are associated with a complex meaning and can be defined more explicitly in terms of possibly rather complex conceptual structures consisting of *Concepts* and *Roles* associated with less complex meanings. The terminological equivalence is expressed by transformation rules, which have the same expressive power as *Structural Descriptions* except to the fact, that they are not interpreted as mere restrictions, but as equivalencies. In KL-ONE, a *Structural Description* allows one to express how *Roles* of a *Concept* interrelate in terms of other *Concepts*. *Structural Descriptions* are usually applied to express the meaning attributed to a specialized *Concept* with respect to (one of) its generalizations. Our application deviates from this usage insofar as we consider only cases where the meaning of the specialization manifests itself merely in an additional *Role* or in a *Role* restriction. Consequently, the meaning associated with this *Role* (or with the restriction) can alternatively be expressed by the *Structural Description*, which is exactly what we want. By means of appropriate definitions, the *Role* 'liquidity' attributed to an asset can be terminologically expanded into 'the possibility of the owner of the asset to concert it into money during its term'.

When applying terminological transformations in generation, the choice of which alternative has to be preferred in a concrete discourse situation is guided by pursuing two partially conflicting Gricean maxims:

- The resulting utterance should be as concise as possible, but still contain the necessary information (including, in particular, co-operative overanswering).
- All parts of the utterance should be comprehensible for the other conversant. Dialog strategies are needed to guide the appropriate selection among the alternatives available.

3 Exploiting the inferential capabilities of the addressee in text planning

In order to produce natural, high quality textual presentations in technical domains, these presentations must not only be adapted to the knowledge attributed to the intended audience, but they must also take into account the inferential capabilities of the addressees. In texts whose aim is to illustrate aspects of problem-solving rather than to present a mere set of facts such as in database descriptions, the elements of the underlying information content are interrelated to a considerable degree. Because of that, humans can infer the intended message conveniently from suitably selected portions of that information content and they usually prefer to do so. However, the majority of generation system do not take inferential relations among presentation ingredients into account. In order to overcome these deficits, a system must exhibit several capabilities to select its presentation content:

- Avoiding the presentation of redundant information, unless doing this would serve another communicative purpose, such as putting emphasis on a particular issue.
- Maintaining coherence in the discourse it produces, as well as in cases where the system wants the user to believe some pieces of information that are implied, but not uttered explicitly.
- Adapting its choices of expressing pieces of information explicitly or leaving them to be uncovered by the addressee's inferential capability, according to evidence about the addressee's domain knowledge and discourse preferences.

We have developed a model that exhibits these capabilities to a certain degree [Horacek, 1997] . It attempts to anticipate contextually-motivated inferences addressees are likely to draw. The model is applicable to explicit representations of reasoning chains which consist of regularities, such as 'group leaders must be assigned to single rooms', and facts that contribute to the underlying reasoning process, such as 'A is a group leader', '1 is a single room'. Furthermore, relations indicate how these facts depend on each other, which regularities are relevant in a given context, and to which entities they apply in a concrete instance, such as the propositions 'group leaders must be assigned to single rooms', and '1 is a single room', which contribute evidence for the assignment of group leader A to room 1. Inferences in understanding utterances embodying these ingredients comprise purely logical conclusions, such as substitution, e.g., 'A must be assigned to a single room', and deduction, e.g., 'Group leader A is in room 1' implies 'room 1 is a single room'. In addition, inferences comprise plausible abductive reasoning, such as 'room 1 is a single room', given the fact 'A is assigned

to room 1' and relevance of the regularity 'group leaders must be assigned to single rooms', as well as contextually motivated assumptions and expectations.

In the course of text planning, rules anticipating these kinds of user inferences are invoked to determine contextually justified derivability of information. For those pieces that are inferable, annotations are introduced in the text structure tree that indicate on which propositions and on which pieces of domain knowledge these inferences rely. Based on that, text variants can be composed from a text plan entailing these annotations about the inferability of pieces of information.

Our model is used to motivate choices in presenting or omitting individual pieces of information; it takes into account the addressees' domain expertise and expectations about logical consequences of purposefully presented information. Moreover, pragmatically-motivated preference criteria can be used to choose among several plausible variants. Several kinds of empirical evidence are incorporated into this text planning process that aims at exploiting conversational implicature, so that a most suitable portion of the plan can be selected for being uttered explicitly. The model is formulated in a reasonably domain-independent way, so that the rules expressing aspects of conversational implicature can be incorporated into typical RST-based text planners. This way, our method adds to discourse planners based on Rhetorical Structure Theory (RST) the ability to omit easily inferable information, so that it overcomes one of the main shortcomings of RST. To summarize our method to incorporate inferential capabilites into text planning, we ground our approach on the following hypotheses:

- Logical reasoning is a good way to model a user's understanding of an explanation.
- The logic must be interpreted in context: assumptions and expectations must be taken into account.
- We have evidence that certain regular interpretation patterns expressible by rules are used by the addressee, which accounts for aspects of conversational implicature.

4 A lexicalization method realized by elementary and compositional mapping schemata

In order to derive a variety of natural language expressions from a common underlying representation, we apply the method of pattern-based mapping schemata described in detail in [Horacek, 1996a]. From the point of view of lexicalization, the conceptual representation serves the purpose of a language-neutral representation covering, for the phenomena of interest, the discrepancies occurring across the natural languages treated.

By means of schemata that express correspondences between elements of conceptual and lexical representation levels, predicates appearing in conceptual representations can be mapped onto predicates appearing on the lexical representation level (that is, lexemes, grammatical functions, and features) in a variety of structure-preserving or structure-changing ways. These schemata refer to individual or small sets of predicates, and they are applicable compositionally and bi-directionally.

The mapping schemata express correspondences between language-neutral representations and lexical elements in a target language, that is, individual lexical items, functions, and features. In the sense of the underlying model, these schemata express the conceptual coverage of lexical items. The schemata consist of pure correspondence specifications and contextual

conditions, referring to adjacent conceptual or lexical items. These schemata only express local correspondences across representation levels. Consistency and composition methods are specified on the respective representation levels, independently from each other. In particular, a subset of the mapping schemata express which conceptual objects can eventually be left implicit in surface expressions, other restrictions permitting. Moreover, the schemata express how such an object is linked to some other explicitly expressed object, that is, which conceptual element or chain of elements corresponds to the relation left implicit. Some of these constructs fall under the category of metonymic relations. Another typical example of this type of mapping is the reference to a person by his/her proper name without expressing the category 'human' or 'person' explicitly. The mapping schemata are supported by a lexicon which entails detailed information about the semantics of lexemes, including data about perspectives and knowledge to derive possible meanings of metonymic relations in the sense of Pustejovsky.

We have identified some (typical) classes of schemata, namely *ZOOM* schemata and *SUBSTITUTION* schemata, where one lexeme, one grammatical function, one feature, one feature value, or the semantics associated with a special operator expresses the meaning associated with a chain of nodes and links in the conceptual representation:

- *ZOOM* schemata serve the purpose of bridging differences in granularity, by relating a lexical predicate to a chunk of conceptual elements; alternative correspondences can be established by implementing results from lexical semantics and insights originating from lexicography, in the degree of accuracy needed for the application at hand. The lexical structures are rebuilt from the language-neutral ontology in a form which is either contracted or expanded compared to a structure which would have been obtained by applying the standard schema. The content bearing parts, however, can immediately be identified in the resulting structure. By applying different compositions of *ZOOM* schemata, textual variations, such as 'John owns a house', 'John is the owner of a house', and 'John's house' can be generated from the same underlying conceptual representation.

- *SUBSTITUTION* schemata serve the purpose of bridging differences in degrees of explicitness, thereby relating pieces of information expressed implicitly on the lexical representation level to their corresponding images in conceptual representations. Thus, information that is expressed explicitly on the language-neutral representation level may be left implicit if this results in coherent expressions on the syntactic functional level. This class of schema is usually applicable, when one phrase on the natural language side bears the role of another one, which does not appear on the surface for reasons of avoiding redundancy on the language level thus providing some sort of paraphrasing capabilities. These capabilities comprise simple cases such as references to a person by his/her name, or references to objects by some prominent property, such as to an asset by means of its associated value in terms of money. Moreover, phenomena also such as metonymy can be treated by these kinds of schemata, to produce expressions in which a metonymic relation is left implicit. Combinations of *ZOOM* and *SUBSTITUTION* schemata contribute to produce structurally divergent expressions such as 'Mary has finished the beer' and 'Mary hat das Beer ausgetrunken' (in German) from the same conceptual representation.

The composition of individual schema application is achieved by unifying the structures

resulting from applications of individual mapping schemata. Controling this process is difficult, although the associated search problem can be drastically reduced by a priori calculating and propagating local incompatibilities, which unfortunately is a rather complicated procedure, too. In a concrete application, however, motivated simplifications and special search heuristics can probably be applied with benefit.

5 Concluding remarks

In this paper, we have presented three methods that enable bridging structural as well as ontological differences between representations underlying formal storage and reasoning systems and natural language presentations. These methods pursue widely complementary goals to bridge differences in degrees of explicitness, detail, and perspective, and they apply distinct mechanisms, including inference rules, terminological logic, and pattern-based mapping schemata that constitute lexical equivalence definitions. The methods proposed can be applied to address a variety of phenomena that bear relevance in producing adequate presentations:

- terminological knowledge expressed in terms of logical equivalence definitions by which largely varying alternative descriptions of specific terms can be produced,
- inferences motivated by causality or by contextually-justified expectations through rules that encapsulate the underlying reasoning, and
- lexically-motivated phenomena such as nominalizations, metonymy, and some sorts of paraphrases that can be described in terms of compositional patterns to enable their proper contextual integration.

Typically, the methods described are applied in sequence, terminological transformations followed by inference rules, and then pattern-based schemata. Since there are apparent interdependencies, especially between terminological equivalencies and inferences, this admittedly simple architecture is not very satisfactory yet. However, this conclusion is not surprising, since all methods encapsulate very detailed reasoning. Nevertheless, the full power of our methods is certainly needed in some parts only for a concrete application, so that this architectural deficit becomes less severe. Consequently, we see these methods as a repertoire of tools, so that a suitable selection and combination of these methods that is oriented on the demands of a specific application should enable one to build a system with improved presentation capabilities.

References

[Horacek, 1996a] H. Horacek. Lexical choice in expressing metonymic relations in multiple language. *Machine Translation*, 11:109–158, 1996.

[Horacek, 1996b] H. Horacek. On bridging the gap between lexical and conceptual representations. Habilitationsschrift, University of Bielefeld, 1996.

[Horacek, 1997] H. Horacek. A model for adapting explanations to the user's likely inferences. *To appear in User Modeling and User-Adapted Interaction*, 1997.

PROVERB: Verbalizing Proofs

Xiaorong Huang
Techne Knowledge Systems Inc.
Toronto, Ontario M5S 3G4
Canada
xiaorong@cs.toronto.edu

Armin Fiedler
Fachbereich Informatik
Universität des Saarlandes
Postfach 15 11 50
D-66041 Saarbrücken, Germany
afiedler@cs.uni-sb.de

Abstract

This paper describes the linguistic part of a fully implemented system called *PROVERB*, which transforms, abstracts, and verbalizes machine-found proofs into formated texts. *PROVERB* employs a pipe line architecture consisting of three components. Its macroplanner linearizes a proof and plans mediating communicative acts by employing a combination of hierarchical planning and focus-guided navigation. The microplanner then maps communicative acts and domain concepts into linguistic resources, paraphrases and aggregates such resources to produce the final Text Structure. A Text Structure contains all necessary syntactic information, and can be executed by our realizer into grammatical sentences.

1 Introduction

PROVERB is a text planner that verbalizes natural deduction (ND) style proofs [Gentzen, 1935, Huang, 1994b]. Like most application-oriented systems, it employs a pipe line architecture consisting of three parts. The architecture of *PROVERB* is illustrated in Figure 1.

The *macroplanner* of *PROVERB* accepts as input a natural deduction style proof, and produces *proof communicative acts* which are structured into hierarchical discourse units. To do so, it uses a strategy which combines *hierarchical planning* and *focus-guided navigation*.

More detailed linguistic decisions are made by the *microplanner*. It makes reference choices, chooses between linguistic resources for domain concepts, combines and reorganizes such resources into paragraphs and sentences. As the representation which supports all these operations, the microplanner of *PROVERB* adopts a

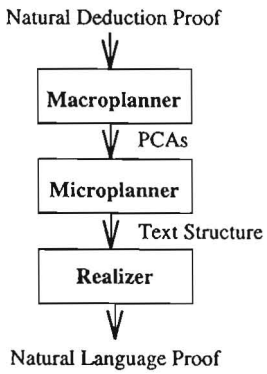


Figure 1: Architecture of *PROVERB*

variation of Meteer's Text Structure, which is also its output.

Our realizer, TAG-GEN, is a syntactic generator based on the grammar formalism TAG [Kilger and Finkler, 1995].

Section 2 and Section 3 are devoted to the macroplanner and the microplanner, respectively. Section 4 contains a complete example. Finally, we shall conclude this paper with a discussion in Section 5.

2 Macroplanning: Hierarchical Planning and Focus-Guided Navigation

PROVERB's macroplanner combines hierarchical planning and local navigation within a uniform planning framework [Huang, 1994a, Huang and Fiedler, 1997]. The *hierarchical planning* splits the task of presenting a particular proof into subtasks of presenting subproofs. *Local navigation* operators simulate the unplanned aspect, where the next conclusion to be presented is chosen under the guidance of a local focus mechanism.

This planning mechanism produces a sequence of *proof communicative acts* that serves as a text plan for an ND-style proof.

2.1 Proof Communicative Acts

Proof communicative acts (PCAs) are the primitive actions planned by the macroplanner of *PROVERB*. Like speech acts, they can be defined in terms of the communicative goals they fulfill as well as their possible verbalizations. An example of a simplistic one conveying the derivation of a new intermediate conclusion is the PCA

```
(Derive Reasons:  (a ∈F, F ⊆G)
        Method: def-subset
        Conclusion:  a ∈G)
```

Depending on the reference choices, the following is a possible verbalization:

> "Since a is an element of F and F is a subset of G, a is an element of G by the definition of subset."

There are also PCAs that convey a partial plan for further presentation and thereby update the global attentional structure. For instance, the PCA

```
(Begin-Cases Goal:  Formula
             Assumptions:  (A B))
```

creates two attentional units with A and B as the assumptions, and *Formula* as the goal by producing the verbalization:

> "To prove *Formula*, let us consider the two cases by assuming A and B."

PCA of the latter sort are also called *meta-comments* [Zukerman, 1991].

2.2 Hierarchical Planning

Hierarchical planning operators represent communicative norms concerning how a proof to be presented can be split into *subproofs*, and how the subproofs can be mapped onto some linear order. Let us look at one such operator, which handles proofs containing case analyses. The corresponding schema of such a proof tree is

$$\frac{\dfrac{\cdots}{\vdots}\quad\dfrac{F}{\vdots}\quad\dfrac{G}{\vdots}}{\dfrac{?L_4 : F \vee G \qquad ?L_2 : Q \qquad ?L_3 : Q}{?L_1 : \triangle \vdash Q}\ \text{CASE}}$$

where the subproof rooted by $?L_4$ leads to $F \vee G$, while subproofs rooted by $?L_2$ and $?L_3$ are the two cases proving Q by assuming F or G, respectively. The applicability encodes the two scenarios of case analysis, where we do not go into details. In both circumstances this operator first presents the part leading to $F \vee G$, and then proceeds with the two cases. It also inserts certain PCAs to mediate between parts of proofs. This procedure is captured by the planning operator below (note that the verbalizations given are only one possible paraphrase):

Case-Implicit

- Applicability Condition: ((task $?L_1$) $\vee$ (local-focus $?L_4$)) $\wedge$ (not-conveyed ($?L_2$ $?L_3$))
- Acts:
  1. if $?L_4$ has not been conveyed, then present $?L_4$ (subgoal 1)
  2. a PCA with a verbalization: "First, let us consider the first case by assuming F."
  3. present $?L_2$ (subgoal 2)
  4. a PCA with a verbalization: "Next, we consider the second case by assuming G."
  5. present $?L_3$ (subgoal 3)
  6. mark $?L_1$ as conveyed
- features: (hierarchical-planning compulsory implicit)

2.3 Local Navigation

The *local navigation* operators simulate the unplanned part of proof presentation. Instead of splitting presentation goals into subgoals, they follow the local derivation relation to find a proof step to be presented next.

The Local Focus

The node to be presented next is suggested by the mechanism of *local focus*. In *PROVERB*, our local focus is the last derived step, while focal centers are semantic objects mentioned in the local focus. Although logically any proof node which uses the local focus as a premise could be chosen for the next step, usually the one with the greatest semantic overlap with the *focal centers* is preferred. In other words, if one has proved a property about some semantic objects, one will tend to continue to talk about these particular objects, before turning to new objects. Let us examine the situation when the proof below is awaiting presentation.

$$\frac{[1]: P(a,b) \quad [2]: Q(a,b)}{[5]: Q(a,b) \wedge R(b,c)}, \quad \frac{[1]: P(a,b), \ [3]: S(c) \quad [4]: R(b,c)}{[5]: Q(a,b) \wedge R(b,c)}$$

Assume that node [1] is the local focus, $\{a, b\}$ is the set of focal centers, [3] is a previously presented node and node [5] is the current task. [2] is chosen as the next node to be presented, since it does not (re)introduce any new semantic objects and its overlap with the focal centers ($\{a, b\}$) is larger than the overlap of [4] with the focal centers ($\{b\}$).

The macroplanner produces as output a sequence of PCAs that is passed on to the microplanner.

3 Microplanning: Choosing and Organizing Linguistic Resources

The task of microplanning comprises, among others, making reference choices; choosing between linguistic resources for functions, predicates and various types of derivations; and combining and reorganizing such resources into paragraphs and sentences. In this paper, we only describe the paraphrasing mechanism. For a more detailed discussion of the microplanner cf. [Huang and Fiedler, 1996]. As the central representation, our microplanner uses Meteer's *Text Structure*.

3.1 Text Structure in *PROVERB*

Text Structure is first proposed by Meteer [Meteer, 1991, Meteer, 1992] in order to bridge the generation gap between the representation in the application program and the linguistic resources provided by the language. By abstracting over concrete linguistic resources, Text Structure should supply the planner with basic vocabularies, with which it chooses linguistic resources. Meteer's Text Structure is organized as a tree, in which each node represents a constituent of the text. In this form it contains three types of linguistic information: *constituency, structural relations among constituents*, and in particular, the *semantic categories the constituents express*.

The main role of the semantic categories is to provide vocabularies which specify type restrictions for nodes. They define how separate Text Structures can be combined, and ensure that the planner only builds expressible Text Structures. For instance if tree A should be expanded at node n by tree B, the resulting type of B must be compatible to the type restriction attached to n. Panaget [Panaget, 1994] argues, however, that Meteer's semantic categories mix the ideational and the textual dimension as argued in the systemic linguistic theory [Halliday, 1994]. Here is one of his examples:

"The ship sank"

is an ideational *event*, and it is textually presented from an EVENT-PERSPECTIVE.

"The sinking of the ship"

is still an ideational *event*, but now presented from an OBJECT-PERSPECTIVE.

On account of this, Panaget split the type restrictions into two orthogonal dimensions: the ideational dimension in terms of the *Upper Model* [Bateman *et al.*, 1990], and the *hierarchy of textual semantic categories* based on an analysis of French and of English. In our work, we basically follow the approach of Panaget.

Technically speaking, the Text Structure in *PROVERB* is a tree recursively composed of kernel subtrees or composite subtrees:

An atomic *kernel subtree* has a head at the root and arguments as children, representing basically a predicate/argument structure.

Composite subtrees can be divided into two subtypes: the first has a special *matrix* child and zero or more *adjunct* children and represents linguistic hypotaxis, the second has two or more *coordinated* children and stands for parataxis.

3.2 Type Restrictions

Each node is typed both in terms of the Upper Model and the hierarchy of textual semantic categories. The Upper Model is a domain-independent property inheritance network of concepts that are hierarchically organized according to how they can be linguistically expressed. Figure 2 shows a fragment of the Upper Model in *PROVERB*. For every domain of appli-

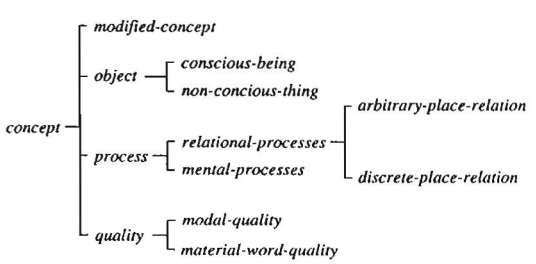


Figure 2: A Fragment of Upper Model in *PROVERB*

cation, domain-specific concepts must be identified and placed as an extension of the Upper Model.

The hierarchy of textual semantic categories is also a domain-independent property inheritance network. The concepts are organized in a hierarchy based on their textual realization. For example, the concept clause-modifier-rankingI[1] is realized as an adverb, clause-modifier-rankingII as a prepositional phrase, and clause-modifier-embedded as an adverbial clause. Figure 3 shows a fragment of the hierarchy of textual semantic categories.

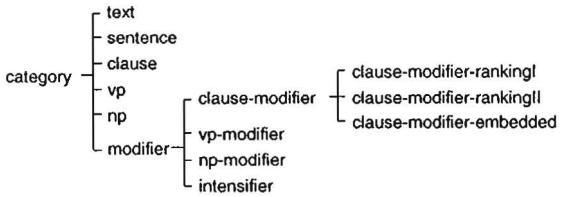


Figure 3: A Fragment of the Hierarchy of Textual Semantic Categories in *PROVERB*

3.3 Mapping APOs to UMOs

The mapping from the content to the linguistic resources now happens in a two-staged way. While Meteer associates the application program objects (APOs) directly with so-called *resources trees*, we map APOs into Upper Model objects, which in turn are expanded to the Text Structures. It is worth noting that there is a practical advantage of this two-staged process. Instead of having to construct resource trees for APOs, the user of our system only needs to define a mapping from the APOs to Upper Model objects (UMOs).

When mapping APOs to UMOs, the microplanner must choose among available alternatives. For example, the application program object para that stands for the logical predicate denoting the parallelism relation between lines may map in five different Upper Model concepts. In the 0-place case, para can be mapped into *object* leading to the noun "parallelism," or *quality*, leading to the adjective "parallel." In the binary case, the choices are *property-ascription* that may be verbalized as "x and y are parallel," *quality-relation* that allows for the verbalization as "x is parallel to y", or *process-relation*, that is the formula "$x \parallel y$."

The mapping of Upper Model objects into the Text Structure is defined by so-called *resource trees*, i.e. reified instances of Text Structure subtrees. The resource trees of an Upper Model concept are assembled in its *realization class*.

3.4 Paraphrasing in *PROVERB*

With the help of a concrete example we illustrate in this section how the Text Structure generator chooses among paraphrases and avoids building inexpressible Text Structures via type checking.

Example We examine a simple logic formula *derive*(para(C1,C2),B). Note that B stands for a conclusion which will not be examined here. We will also not follow the procedure in detail.

In the current implementation, the rhetorical relation *derive* is only connected to one Upper Model concept *derive*, a subconcept of *cause-relation*. The realization class associated to the concept, however, contains several alternative resource trees leading to different patterns of verbalization. We only list five variations below:

- B, since A.
- Since A, therefore B.
- A leads to B.
- Because A, B.
- Because of A, B.

The resource tree of the first alternative is given in Figure 4.

The logic predicate para(C1, C2) can be mapped to one of the following Upper Model concepts, where we always include one possible verbalization:

- *quality-relation*(para, C1, C2)
  (line $C1$ is parallel to $C2$)
- *process-relation*(para, C1, C2)
  ($C1 \| C2$)
- *property-ascription*(para, C1 $\wedge$ C2)
  (lines C1 and C2 are parallel)

Textually, the property-ascription version can be realized in two forms, represented by the two resource trees in Figure 5.

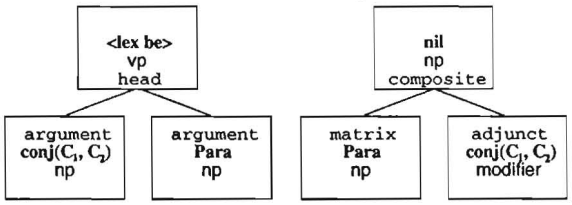
Type checking during the construction of the Text Structure must ensure, that the realization be compatible along both the ideational and the textual dimension. In this example, the combination of the tree in Figure 4 and the first tree in Figure 5 is compatible and will lead to the verbalization:

"B, since $C1$ and $C2$ are parallel."

[1] Concepts of the hierarchy of textual semantic categories are noted in sans-serif text.

```
realizationclass derive reason R conclusion C
  resourcetree compositetree content nil
                                  tsc sentence clause
                                  matrix leaf content C
                                              tsc clause
                                  adjunct compositetree content since
                                                        tsc clause
                                                        matrix leaf content R
                                                                    tsc clause
  ⟨further resource trees ...⟩
```

Figure 4: The Realization Class for *derive*



As a verb phrase As a nominal phrase

Figure 5: Textual Variations in form of Resource Trees

The second tree in Figure 5, however, can only be combined with another realization of *derive*, resulting in:

> "Because of the parallelism of line $C1$ and line $C2$, B."

In our current system we concentrate on the mechanism and are therefore still experimenting with heuristics which control the choice of paraphrases. One interesting rule is to distinguish between general rhetorical relations and domain specific mathematical concepts. While the former should be paraphrased to increase the flexibility, continuity of the latter helps the user to identify technical concepts.

4 A Complete Example

In this section, we shall present a short example of *PROVERB*'s output. The input is a machine-found proof at the assertion level of a theorem taken from a mathematical textbook (Figure 6).

A user may choose the style of the output text by tuning two parameters: implicit vs. explicit, abstract vs. detailed. Given abstract and explicit as options, the macroplanner generates the following sequence of PCAs:

```
THEOREM Reasons group(F, *),
                subgroup(U, F, *),
                unit(F, 1, *),
                unit(U, 1_U, *)
        Conclusion 1_U = 1
```

```
PROOF
ATTENTIONBEGIN
ASSUME Conclusion group(F, *)
ASSUME Conclusion subgroup(U, F, *)
ASSUME Conclusion unit(F, 1, *)
ASSUME Conclusion unit(U, 1_U, *)
ATTENTIONBEGIN
ATTENTIONBEGIN
DERIVE Reasons unit(U, 1_U, *)
       Conclusion 1_U ∈U
DERIVE Conclusion ∃x.x ∈U
ATTENTIONEND
BEGINASSUMECHOICE Conclusion u_1 ∈U
                  Parameters u_1, x
ATTENTIONBEGIN

DERIVE Reasons group(F, *)
       Conclusion semigroup(F, *)

DERIVE Conclusion 1_U = 1
ATTENTIONEND
ENDASSUMECHOICE Conclusion 1_U = 1
                Parameters u_1
ATTENTIONEND
ATTENTIONEND
QED
```

Note that the PCAs are segmented into a hierarchy of attentional spaces, which is used to make reference decisions [Huang, 1997]. The microplanner maps this sequence of PCAs via Upper Model objects into a Text Structure used as linguistic specification of the text. It aggregates Upper Model objects and paraphrases the concepts by combining various resource trees. Finally, the realization component TAG-GEN produces the following text:

Theorem:
Let F be a group, let U be a subgroup of F, and let 1 and 1_U be unit elements of F and U. Then 1_U equals 1.
Proof:
Let F be a group, let U be a subgroup of F, and let 1 and 1_U be unit elements of F and U.
Because 1_U is an unit element of U, $1_U \in U$. Therefore, there is x such that $x \in U$.

NNo	S;D		Formula	Reason
7.	7;	$\vdash$	$group(F,*) \wedge subgroup(U,F,*) \wedge unit(F,1,*) \wedge unit(U,1_U,*)$	(Hyp)
8.	7;	$\vdash$	$U \subset F$	(Def-subgroup 7)
9.	7;	$\vdash$	$1_U \in U$	(Def-unit 7)
10.	7;	$\vdash$	$\exists_x x \in U$	($\exists$ 9)
11.	;11	$\vdash$	$u \in U$	(Hyp)
12.	7;11	$\vdash$	$u * 1_U = u$	(Def-unit 7 11)
13.	7;11	$\vdash$	$u \in F$	(Def-subset 8 11)
14.	7;11	$\vdash$	$1_U \in F$	(Def-subset 8 9)
15.	7;11	$\vdash$	$semigroup(F,*)$	(Def-group 7)
16.	7;11	$\vdash$	$solution(u,u,1_U,F,*)$	(Def-solution 12 13 14 1
17.	7;11	$\vdash$	$u * 1 = u$	(Def-unit 7 13)
18.	7;11	$\vdash$	$1 \in F$	(Def-unit 7)
19.	7;11	$\vdash$	$solution(u,u,1,F,*)$	(Def-solution 13 17 18 1
20.	7;11	$\vdash$	$1 = 1_U$	(Th-solution 17 16 19)
21.	7;	$\vdash$	$1 = 1_U$	(Choice 10 20)
22.	;	$\vdash$	$group(F,*) \wedge subgroup(U,F,*) \wedge unit(F,1,*) \wedge unit(U,1_U,*) \Rightarrow 1 = 1_U$	(Ded 7 21)

Figure 6: Abstracted Proof about Unit Element of Subgroups

Let u_1 be such an x. Since $u_1 \in U$ and 1_U is an unit element of U, $u_1 * 1_U = u_1$. Since F is a group, F is a semigroup. Since U is a subgroup of F, $U \subset F$. Because $U \subset F$ and $1_U \in U$, $1_U \in F$. Similarly, because $u_1 \in U$ and $U \subset F$, $u_1 \in F$. Then, 1_U is a solution of $u_1 * x = u_1$.

Because $u_1 \in F$ and 1 is an unit element of F, $u_1 * 1 = u_1$. Since 1 is an unit element of F, $1 \in F$. Then, 1 is a solution of $u_1 * x = u_1$.

Therefore, 1_U equals 1. This conclusion is independent of the choice of u_1. ∎

Please note the variation in the text, as well as in the structure of the sentences, and in using mathematical symbols or words. Moreover aggregation techniques reduced redundancies as in "let 1 and 1_U be unit elements of F and U", where to clauses were grouped into a single one.

5 Conclusion

This paper describes the linguistic part of a fully implemented system called *PROVERB*, which transforms, abstracts, and verbalizes machine-found proofs into formated texts. *PROVERB* employs a pipe line architecture consisting of three components. Its macroplanner linearizes a proof and plans mediating communicative acts by employing a combination of hierarchical planning and focus-guided navigation. The microplanner then maps communicative acts and domain concepts into linguistic resources, paraphrases and aggregates such resources to produce the final Text Structure. The Text Structure is finally executed by our realizer TAG-GEN into grammatical sentences.

PROVERB works particularly well with textbook size examples and runs fully automatically for every new example. The output texts are close to detailed proofs in textbooks and are basically accepted by the community of automated reasoning. To benefit from the microplanning techniques which significantly improve the fluency of text, however, linguistic resources must be introduced with each new domain of application. We are working on an interface to simplify this process.

References

[Bateman *et al.*, 1990] John A. Bateman, Robert T. Kasper, Johanna D. Moore, and Richard A. Whitney. A general organization of knowledge for natural language processing: the Penman upper model. technical report, 1990. ISI Penman Note.

[Gentzen, 1935] Gerhard Gentzen. Untersuchungen über das logische Schließen I. *Mathematische Zeitschrift*, 39:176–210, 1935.

[Halliday, 1994] M. A. K. Halliday. *An Introduction to Functional Grammar*. Edward Arnold, 2. edition, 1994.

[Huang and Fiedler, 1996] Xiaorong Huang and Armin Fiedler. Paraphrasing and aggregating argumentative texts using text structure. In *Proceedings of the 8th International Natural Language Generation Workshop*, pages 21–30, Herstmonceux Castle, Sussex, UK, 1996.

[Huang and Fiedler, 1997] Xiaorong Huang and Armin Fiedler. Proof presentation as an application of NLG. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI)*, Nagoya, Japan, 1997.

[Huang, 1994a] Xiaorong Huang. Planning argumentative texts. In *Proceedings of 15th International Conference on Computational Linguistics*, pages 329–333, Kyoto, Japan, 1994.

[Huang, 1994b] Xiaorong Huang. Reconstructing proofs at the assertion level. In Alan Bundy, ed-

itor, *Proceedings of the 12th Conference on Automated Deduction*, number814 in LNAI, pages 738–752, Nancy, France, 1994. Springer Verlag.

Huang, 1997] Xiaorong Huang. Planning reference choices for argumentative texts. 1997.

Kilger and Finkler, 1995] Anne Kilger and Wolfgang Finkler. Incremental generation for real–time applications. Research Report RR-95-11, DFKI, Saarbrücken, Germany, July 1995.

Meteer, 1991] Marie W. Meteer. Bridging the generation gap between text planning linguistic realization. *Computational Intelligence*, 7(4), 1991.

Meteer, 1992] Marie W. Meteer. *Expressibility and the Problem of Efficient Text Planning*. Pinter Publishes, London, 1992.

Panaget, 1994] Franck Panaget. The micro-planning component of a generation system. technical report, IRST, 1994.

Zukerman, 1991] Ingrid Zukerman. Using meta-comments to generate fluent text in a technical domain. *Computational Intelligence*, 7:276–295, 1991.

The Project *ACNLG*: Applying Natural Language Generation in China

Xiaorong Huang
German Research Center for Artificial Intelligence
Stuhtsatzenhaus Weg 3
D–66123 Saarbrücken, Germany
Email: huang@cs.uni-sb.de
Tianfang Yao Huanye Sheng
Shanghai Jiaotong University
Shanghai 200030
VR China
Email: yao-tf@cs.sjtu.edu.cn hysheng@sjtu.edu.cn

1 Overview of *ACNLG*

ACNLG is organized as a cooperation between the German Research Center for Artificial Intelligence (DFKI) and Shanghai Jiaotong University. Financially, the project is supported by the VW-Stiftung, the Shanghai Jiaotong University, the Chinese National Science Foundation and the Shanghai Commission for Science and Technology for a time period of three years.

The primary goals of the project can be summarized as following [10]:

- to develop an *architecture* for applied multilingual generation, which handles topologically different languages such as Chinese, English and German,
- to build a *computational grammar* for Chinese NLG,
- to test our approach towards multilingual NLG in one or two real-world applications.

As the first two applications we are currently investigating:

- a multilingual weather forecast assistant (MLWFA), together with the Shanghai Meteorological Center,
- Generation of multilingual statistical reports, together with the Bank of China.

2 The architecture of *ACNLG*

2.1 Applied NLG

Over the past two decades a multitude of techniques has been developed to produce coherent text from internal representations ([18], see also [1] for an online review). Our architecture is designed based on two criteria: the linguistic flexibility our applications need and software manageability of the techniques in concern.

Although quite different architectures have been proposed for NLG systems, most application oriented systems employ a pipeline architecture consisting of three parts:

- A macroplanner or content planner that chooses and orders information to be included in the text.
- A microplanner or sentence planner that chooses appropriate linguistic resources for the pieces of information chosen, and arranges them into paragraphs and sentences.
- A surface generator that handles syntactic operations and produces grammatical natural language utterances.

Following this structure, our architecture of the kernel generation system of *ACNLG* is illustrated in Fig.1.

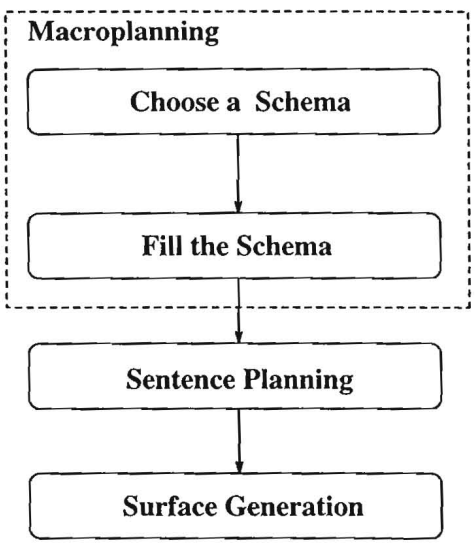


Figure 1: Kernel Generator of *ACNLG*

Below we briefly discuss these components and the techniques we adopt for *ACNLG*:

- Content planning: For our applications, we employ the schema-based approach [13, 15] approach. To handle recursive data, we might also use schemata as plan operators for hierarchical planning.

- Sentence Planning: Flexible sentence planning is necessary [4]. In *ACNLG* we are investigating various sentence planning techniques such as paraphrasing and aggregation [3, 8, 9]. As the representation we will explore a different version of the Text Structure proposed by Meteer [14].

- The system should keep track of both local and global focus of attention, to enable reasonable reference choices [19] and cue-words generation [17].

- We tested our first Chinese grammar [20] with the syntactic generator TAG-GEN, which is developed at DFKI [11]. However, our applications need a leaner and more efficient generator, which we developmed in C++, the standard TAG as underlying formalism. The TAG-GEN Chinese grammar is also modified into standard TAG.

For more details, readers are referred to [10].

3 A Practical Approach towards Multilingual NLG

At least the following three approaches towards multilingual generation can be observed [6]:

- identifying and integrating of certain interlingua at different levels of processing [12],

- using some integrated network with language-dependent and language-independent parts as representation. One example is the system KPML [2], as well as planners using PENMAN/KPML [16],

- using universal machineries with language-specific declarative knowledge (schemata, microplanning rules, grammar). This is the architecture the first author used in the system *PROVERB*, which verbalizes machine-found mathematical proofs [5, 8, 7]. We will investigate it further for our new application. The primary advantage of our architecture is that this makes a large scale software system easier to maintain. A potential drawback is a massive redundancy that may lead to software maintenance problems as well.

4 Current Status and Future Plan

ACNLG just finished its first year. We implemented a first prototype consisting of a schema-based macroplanner and a TAG-Based syntactic generator. We also developed a Chinese grammar in TAG-GEN, which is updated into the current grammar used by the new generator. We are planning to put this prototype into test use, and at the same time develop a microplanner to produce more flexible text. After the first application, we will start with the second application.

References

1. Survey of the state of the art in human language technology. http://www.cse.ogi.edu/CSLU/HLTsurvey/HLTsurvey.html, 1995.

2. John Bateman. Basic technology for multilingual theory and practise: the kpml development environment. In Richard Kittredge, Sergei Nirenburg, Dietmar Rösner, and Donia Scott, editors, *Proc. IJCAI-95 Workshop on MULTILINGUAL TEXT GENERATION*, pages 1–12, 1995.

3. Hercules Dalianis and Eduard Hovy. Aggregation in natural language generation. In Michael Zock, Giovanni Adorni, and Giacomo Ferrari, editors, *Proc. 4th European Workshop on Natural Language Generation*, pages 67–78, 1993.

4. Eduard H. Hovy. Unresolved issues in paragraph planning. In Robert Dale, editor, *Current Issues in Natural Language Generation*, pages 17–45. Academic Press, 1990.

5. Xiaorong Huang. Planning argumentative texts. In *Proc. of 15th International Conference on Computational Linguistics*, pages 329–333, Kyoto, Japan, 1994.

6. Xiaorong Huang. Choosing among architectures for applied multilingual nlg. In Dominique Estival and Robert Dale, editors, *Proc. PRICAI-96 Workshop on Future Issues for Multi-lingual Text Processing*, pages 25–31. Griffith University, Canada, 1996.

7. Xiaorong Huang. Planning reference choices for argumentative texts. In *Proc. ACL'97/EACL97 Joint Conference*, 1997.

8. Xiaorong Huang and Armin Fiedler. Paraphrasing and aggregating argumentative text using text structure. In *Proc. of 8th International Workshop on Natural Language Generation*, pages 21–30, Herstmonceux, Sussex, UK, 1996.

9. Xiaorong Huang and Armin Fiedler. Proof verbalization as an application of nlg. In *Proc. Joint Conference on Artificial Intelligence97*, 1997.

10. Xiaorong Huang, Tianfang Yao, and Guodong Gao. Generating chinese weather forecast with stylistic variations. In *17th International Conference on Computer Processing of Oriental Language*. Oriental Languages Computer Society, Inc., 1997.

11. Anne Kilger and Wolfgang Finkler. Incremental generation for real-time applications. Research Report RR-95-11, DFKI, Saarbrücken, Germany, 1995.

12. Richard Kittredge. Efficiency vs. generality in interlingual design: Some linguistic considerations. In Richard Kittredge, Sergei Nirenburg, Dietmar Rösner, and Donia Scott, editors, *Proc. IJCAI-95 Workshop on MULTILINGUAL TEXT GENERATION*, pages 64–74, 1995.

13. Kathleen R. McKeown. *Text Generation.* Cambridge University Press, Cambridge, UK., 1985.

14. Marie W. Meteer. *Expressibility and the Problem of Efficient Text Planning.* Pinter Publishes, London, 1992.

15. Johanna Doris Moore and Cécile L. Paris. Planning text for advisory dialogues. In *Proc. 27th Annual Meeting of the Association for Computational Linguistics*, pages 203–211, Vancouver, British Columbia, 1989.

16. Cécile Paris, Keith Vander Linden, Markus Fischer, Anthony Hartley, Lyn Pemberton, Richard Power, and Donia Scott. A support tool for writing multilingual instructions. In Chris S. Mellish, editor, *Proc. of IJCAI-95*, volume 2, pages 1398–1304, Montreal, Canada, 1995. Morgan Kaufmann.

17. Rachel Reichman. *Getting Computers to Talk Like You and Me. Discourse Context, Focus, and Semantics.* MIT Press, 1985.

18. Ehud Reiter. Has a consensus nl generation architecture appeared, and is it psycholinguistically plausible. In *Proc. of 7th International Workshop on Natural Language Generation*, pages 163–170, Kennebunkport, Maine, USA, 1994.

19. Ehud Reiter and Robert Dale. A fast algorithm for the generation of referring expressions. In *Proc. of COLING-92*, volume 1, pages 232–238, 1992.

20. Yu-Fang Wang. A tree adjoining grammar for chinese weatherforecasts. Master's thesis, Universität des Saarlandes, Saarbrücken, Germany, 1997.

Microplanning in Verbmobil as a Constraint–Satisfaction Problem

Anne Kilger

June 20, 1997

Abstract

Microplanning in the dialog translation system Verbmobil bridges the gap between the output of the transfer component and the input to the syntactic generator. It solves parts of the tasks of lexical selection and choice of syntactic features using techniques from the area of constraint–satisfaction problems.

1 Characterization of Microplanning

There is a **Generation Gap** between Text Planning and Text Realization, which has been named and discussed in detail by [Met90]. Generation systems have to make sure that all that is constructed by the planner is also expressible with syntactic means of the target language. Meteer proposes the insertion of a **Text Structure Level** between Text Planning and Text Realization, where expressible combinations of concrete linguistic resources are grouped, preventing the Text Planner from choosing sets of features that cannot be verbalized.
Following [Hov96], the stage of generation that has been introduced over the past years to bridge the Generation Gap is often named as **Microplanning** or **Sentence Planning**. It consists of "several distinct, rather different subtasks. Each subtask addresses some aspect of the information selected to be said by the text planner, and performs some operation on it, often adding additional information that eventually results in higher quality surface form." These tasks include clause conjunction and subordination into longer sentences, clause–internal ordering of constituents, aggregation (elision) to remove redundancies, theme control, focus control, reference (anaphora) specification, and lexical selection.
Current systems mostly contain solutions for only some of these tasks, sometimes treated as separate modules. There is need of an approach that comprises the efficient computation of distinct tasks while regarding their multiple interdependencies.

2 The Microplanning Task in Verbmobil

For generation within a dialog translation system like Verbmobil (see [Wah93, BWW97]), macroplanning is not needed since the information to be generated is selected and organized by the speaker of the source language utterance. Furthermore, Verbmobil realizes a semantic–based transfer (see [DE96]), i.e. part of the generation (microplanning) task is already dealt with by the transfer component and part of the language–specific knowledge relevant for generation is encoded within the transfer rules.
The input interface language chosen for the Verbmobil generator (VIT, Verbmobil Interface Term) comprises the encoding of language–specific semantic information following the Discourse Representation Theory (**DRT**, see [KR93]). Each individual indicated by some input utterance

is formally represented by a **discourse referent**. Information about the individual is encoded within a **DRS–condition**, combining a predicate with the chosen discourse referent. Relations between descriptions of different discourse referents lead to a global hierarchical semantic structure. So–called "holes" are used to define underspecified relations plus constraints for their filling. Each VIT additionally contains semantic, pragmatic and syntactic information usable for generating an adequate output.
The output of the Verbmobil microplanner is a sentence plan that serves as input for the syntactic realization component. It describes a dependency tree over lexical items annotated with syntactic, semantic and pragmatic information which is relevant to produce an acceptable utterance and guide the speech synthesis component.

3 The System VM–IMP

The Verbmobil microplanner VM–IMP[1] solves several of the tasks described in Section 1 in an integrated way by making use of an constraint–satisfaction approach. The conceptual basis and some details of its realization are explained in the next two sections.

3.1 Conceptualization

The subtasks of microplanning are subject to multidirectional dependencies, e.g. between lexical selection and choice of syntactic specifications. Regarding the input DRS–elements as variables, the microplanning task can be described as mapping each variable to a syntactic specification such that a globally consistent solution can be derived. The domains of the variables correspond to possible syntactic realizations of the semantic elements (the right sides of microplanning rules, see Section 3.2), including specifications of lexical items and syntactic features. Those variables can be used as basis for the description of a **constraint–satisfaction problem** (CSP, see e.g. [Kum92]). The predicates relating the variables have to define some sort of matching mechanism such that a global variable instantiation can be guaranteed to be a valid input for the sentence realization component.
The advantages of a constraint system do not only lie in the declarativity of the knowledge sources. Having defined a suitable representation of the problem to be solved, a constraint–based approach also establishes a testbed for examining the pros and cons of different evaluation methods, including backtracking, constraint propagation, heuristics for the order of the instantiation of variable values etc.
The second important design principle used not only for microplanning but also for syntactic generation in Verbmobil is **off–line preprocessing**. By anticipating relevant parts of the generation task and doing some work in advance — thereby changing the knowledge sources — on–line processing can be speeded up. Additionally, knowledge descriptions can be modularized for easing the task of defining rules, at the same time allowing their combination by off–line preprocessing in the course of which contradictory combinations can be filtered out.

3.2 Realization

The knowledge sources and features of processing of the microplanning component are described on the basis of an example input VIT as visualized in Figure 1. The figure only shows some slots of a VIT which are necessary to explain the main features of microplanning. The arguments of conditions given in the semantics slot (the DRS) are specified by symbols starting with L, H or I

[1] VerbMobil Incremental MicroPlanner, incrementality is currently not fully realized.

```
vit(segment_description(...,wir einigen uns auf einen neuen termin),
    [decl(l19,h5),                                          % Semantics
     agree_on(l18,i5),
     arg2(l18,i5,i6),
     pron(l23,i7),
     appointment(l20,i6),
     arg1(l18,i5,i7),
     new(l24,i6),
     indef(l22,i6,l15,ht4)],
    l19,                                                    % Main Label
    [leq(l16,h5), ccom_plug(h5,l16)],                       % Scope
    [sem_group(l16,[l18]), sem_group(l15,[l20])])           % Groupings
```

Figure 1: An Example VIT as input to VM–GECO

for labels (unique identifiers), holes (underspecified scope) and instances (discourse referents). In the main label slot, the "hook" for the DRS is defined, the scope slot contains some constraints about the filling of holes, the groupings slot allows for defining group–labels as sets of labels.
VM–IMP doesn't traverse the input DRS in a top–down fashion but makes use of its non–recursive representation by triggering activities by the single conditions. Thereby it is prepared for incremental processing which hopefully will be realized in a later stage of Verbmobil. The microplanner is driven by (bundles of) conditions, discourse referents, and holes found in the input DRS. Each of these features is reflected by a distinct set of microplanning rules that are applied conjointly during the process of microplanning (see Section 3.3).
The microplanning rules are represented as pattern–action pairs (or pattern–condition–action triples). A pattern is to be matched with part of the input, the action describes a bundle of syntactic features realizing the message part in an adequate way.

3.2.1 Microplanning Content Rules

Microplanning Content Rules define the mapping from (bundles of) semantic predicates to syntactic features. Thereby relations to semantic complements given in the input are translated into semantic/syntactic relations that form a part of the dependency tree in the microplanner output.
Two (simplified) microplanning content rules for the semantic condition AGREE_ON are shown in Figure 2. Each entry consists of at least three elements. The first is an expression that is

```
;; normal finite form
((AGREE_ON (L I))                                           ;;pattern
 (($not ($sem-match NOM (L I))))                            ;;condition
 ($IDENT$ L AGREE_ON)                                       ;;body
 (AGREE_ON (CAT V) (HEAD AGREE_ON_V1) (MOOD $get-mood-info I)
           (VOICE $get-voice-info I) (FORM ordinary)
           (TENSE $get-tense-info I) (ARG1-TYPE normal)
           (ARG1-FUNC ARG1) (ARG2-TYPE prep) (ARG2-FUNC ARG2)
           (ARG2-PREP ON)))
;; nominalized form
((AGREE_ON (L I))                                           ;;pattern
 NIL                                                        ;;condition
 ($IDENT$ L AGREE_ON)                                       ;;body
 (AGREE_ON (CAT N) (HEAD AGREEMENT_N3) (NUM $get-num-info AGREE_ON)
           (ARG1-TYPE normal prep) (ARG1-FUNC ARG1) (ARG1-PREP OF)
           (ARG2-TYPE prep) (ARG2-FUNC ARG2) (ARG2-PREP ON)))
```

Figure 2: Example Microplanning Content Rules

to be matched with the semantics slot of the input VIT and represents the part of the VIT

that is mapped to a syntactic specification. The second describes additional requirements to be fulfilled by the input. Here, contextual tests on the global VIT may be stated, e.g. conditions that have to appear in the semantics slot to form a valid context for the rule or some specific features from other slots. The third and further elements of a microplanning rule contain syntactic specifications, i.e. the action. Each action element introduces a "syntactic" identifier for the syntactic specification that either reflects a unique input element or uses a new name for additionally introduced elements on the output side. Pairs of the form (feature value) describe relevant features used for computing the global syntactic specification that is handed over to the syntactic generator. There also may be identifications of label names with syntactic identifiers which are relevant for mapping the DRS-relations to syntactic relations. They are introduced via the keyword $IDENT$.
The rules in Figure 2 describe possible mappings of the condition AGREE_ON to the verb AGREE_ON_V1 or the noun AGREEMENT_N3. In the condition part of the verbal mapping, the existance of a NOM-condition within the semantics slot is tested (which would forbid the verbal form by demanding a NOMinalized form). The body describes the result of lexical selection plus generic functions for computing relevant syntactic features like tense and mood. Via the feature names ARGi-TYPE and ARGi-FUNC, constraints for the mapping of semantic arguments to syntactic realizations are defined, e.g. the semantic relation ARG2 must be filled by a prepositional phrase with the preposition ON ("We agree on a new appointment.").

The microplanning content rules are not directly entered by a rule writer but are compiled off-line from distinct knowledge sources for word choice rules, rules for syntactic decisions and rules for mapping semantic roles to syntactic relations. The first rule in Figure 2 is a compilation result from the three rules sketched in Figure 3. Keeping word choice rules and syntactic choice rules

```
;; Syntactic choice rules
((AGREE_ON (L I))
 (($not ($sem-match NOM (L I))))
 (AGREE_ON (CAT V) (MOOD $get-mood-info I)
           (VOICE $get-voice-info I) (FORM ordinary)
           (TENSE $get-tense-info I)))
((AGREE_ON (L I))
 NIL
 (AGREE_ON (CAT N)))
;; Word choice rules
((AGREE_ON (L I))
 NIL
 ($IDENT$ L AGREE_ON)
 (AGREE_ON (CAT V) (HEAD AGREE_ON_V1)))
((AGREE_ON (L I))
 NIL
 ($IDENT$ L AGREE_ON)
 (AGREE_ON (CAT N) (HEAD AGREEMENT_N3) (NUM $get-num-info AGREE_ON)))
;; Complement specification
((AGREE_ON
 (AGREE_ON_V1 arg1 NIL (arg1 ARG1 normal) (arg2 ARG2 (prep ON)))
 (AGREEMENT_N3 NIL NIL (arg1 ARG1 normal (prep OF)) (arg2 ARG2 (prep ON)))))
```

Figure 3: Sources for Microplanning Content Rules

seperated helps avoiding redundancy. The set of complement specifications describes mappings of semantic relations as defined for the DRS-conditions onto relation names used within the syntactic grammar and can be compiled off-line from the grammar.

3.2.2 Microplanning Relation Rules

The microplanning relation rules map bundles of semantic conditions to a set of semantic relations between head and modifiers. Conditions describing the same discourse referent form those bundles which have to be reflected at the syntactic level. The relation rules define all allowed dependency relations between pairs of conditions with the same discourse referent. In this way the basis is built for computing all possible dependency structures for a bundle. The relation rules are currently encoded by hand but will soon be automatically compiled from the grammar of the syntactic generator.
The example entry in Figure 4 shows a characterization of the relation between noun and adjective. Each relation rule consists of three parts, the first describing the possible head, the

```
;; Noun and Adjective
(((SEM-CLASS CN))
 ((SEM-CLASS ADJ))
 ((2 (GOVERNED-BY N) (REGENT 1) (REGENT-FUNC adjunct))))
```

Figure 4: Microplanning Relation Rules

second describing the dependent element, the third defining the relation (1 refers to the head, 2 to the modifier) and (optionally) some contextual constraints for the relation to hold. For characterizing matching predicates, so–called semantic classes (SEM–CLASS) can be used to refer to groups of predicates. The entry of Figure 4 refers to a "common noun" as the head and an adjective as the modifier of the noun, describing the relation by specifying the REGENT-feature of modifier.
Applying the relation rules to all semantic elements from our example VIT which describe discourse referent I6 leads to the following syntactic specification:

```
(($IDENT$ I6 APPOINTMENT)
 (NEW (REGENT APPOINTMENT) (REGENT-FUNC adjunct)))
```

It defines discourse referent I6 to be syntactically represented by element APPOINTMENT. NEW is the modifier of this element. The complement INDEF is inserted via a content rule.

3.2.3 Microplanning Hole Rules

Microplanning Hole Rules define the mapping of holes as well as groups of labels used within semantic descriptions to one syntactic element. In the scope–slot of the VIT each hole is associated with one or several labels referring to its possible fillers. In the groupings slot labels are redefined as groups of labels.
In our first approach to finding a representative for the set of labels, we chose to look for the head of the syntactic subtree built up by the syntactic elements that result from the mapping of single labels. For hole *h5* in the VIT in Figure 1, the leq–statement leq(*l16,h5*) maps it to label *l16* which is identified via the statement sem_group(*l16*,[*l18*]) with a (one–element) set of labels. In the example given the hole refers to a unique element *l18* which is identified with AGREE_ON by a content rule. If there were a set of possible fillers for *l16* (e.g. *l18* plus *l4711*), the microplanner would produce mapping alternatives constrained by the actual dependency relations between the elements:

```
(($IDENT$ L16 L18)
 (L4711 (REGENT L18)))
(($IDENT$ L16 L4711)
 (L18 (REGENT L4711)))
```

3.3 Performing Microplanning

During microplanning each element of the semantic structure is mapped onto a set of possible syntactic realizations[2], each bundle of conditions with the same discourse referent is mapped onto a set of possible head–modifier relations, and each hole or group is mapped onto a set of possible identifiers of syntactic elements. For the input VIT shown in Figure 1 this leads to the variable set (DECL AGREE_ON ARG2 PRON APPOINTMENT ARG1 NEW INDEF H5 L16 L15).
Unfortunately, it is not enough to define binary matching constraints between each pair of variables that purely test the compatibility of the described syntactic features. Some syntactic specifications may contain identifications of e.g. discourse referents and syntactic identifiers (via the feature $IDENT$). When choosing one of the alternative results of the microplanning relation rules, this should influence the result of the compatibility test between a pair of variables refering to the identifiers related in the $IDENT$–rule. That is why the constraint net is not easily subdivided into subnets that can be efficiently evaluated. The immense amount of combinations of alternative values has to be handled by the known means for CSP:

- All variables with 1–value domains are united, applying the matching mechanism to their values. A fail immediately leads to a global fail of microplanning.
- 2–consistency is partially computed by matching value pairs and filtering out inconsistent ones. Thereby, matching results and knowledge about binary incompatibility are stored and reused during further processing (global matching).
- By comparing the domains of the variables, the microplanning task can be subdivided into recursive subtasks in an intelligent way, reducing the risk of repeatedly computing and using wrong partial solutions in larger contexts.
- Intelligent backtracking can be guided by identifying variables that are possible candidates for sources of errors.

Although some naive approaches towards constraint–satisfaction systems suffer from inefficiency, the current system has acceptable runtime and we expect that the accurate examination of the task will lead to the use or development of a special algorithm with even better performance.
The (partial) result of the constraint–satisfaction process for the input shown in Figure 1 is graphically presented in Figure 5. The dependency hierarchy is visualized by nodes representing syntactic elements and (parts of) the features chosen for their realization. They are linked by roles compatible with the grammar.

4 Future Work

Currently, we are testing several constraint propagation and backtracking mechanisms for their suitability with respect to the microplanning task. Although we have not yet completed the design of the microplanner, we have gained valuable experience with

- different representations of the problem and their advantages and disadvantages for the processes of mapping and constraint–satisfaction,
- possible influences of the weighing of alternatives on the instantiation of variables,

[2]There are also n:m mappings.

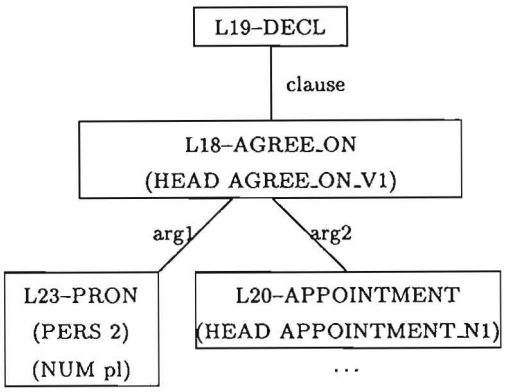


Figure 5: Microplanning Result as Dependency Tree

- the usage of constraint hierarchies for the sake of robustness, e.g. by allowing for "minor semantic and/or syntactic contradictions" in the output. In that way variable levels of both "correctness" and "acceptability" can be encoded. Furthermore, we examined
- the suitability of incremental constraint–satisfaction techniques for the microplanning task,

which will fasten our progress in developing constraint–based microplanning.

References

[BWW97] Th. Bub, W. Wahlster, and A. Waibel. Verbmobil: The combination of deep and shallow processing for spontaneous speech translation. In *Proceedings of ICASSP '97*, 1997. (forthcoming).

[DE96] M. Dorna and M. Emele. Semantic–based transfer. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING '96)*, 1996.

[Hov96] E. Hovy. An overview of automated natural language generation. In X. Huang, editor, *Proceedings of the International Symposium on Natural Language Generation and the Processing of the Chinese Language, INP(C)–96*, pages 15–31, Shanghai, China, 1996.

[KR93] H. Kamp and U. Reyle. *From Discourse to Logic*, volume 42 of *Studies in Linguistics and Philosophy*. Kluwer Academic Publishers, Dordrecht, 1993.

[Kum92] V. Kumar. Algorithms for constraint–satisfaction problems: A survey. *AI Magazine*, 13(1):32–44, 1992.

[Met90] M. W. Meteer. *The "Generation Gap": the Problem of Expressibility in Text Planning*. PhD thesis, Department of Computer and Information Science, University of Massachusetts, Amherst, MA, 1990. BBN Report No. 7347.

[Wah93] W. Wahlster. Verbmobil: Translation of face–to–face dialogs. Research Report RR-93-34, German Research Center for Artificial Intelligence (DFKI GmbH), Saarbrücken, Germany, 1993.

AGILE: Automatic drafting of technical documents in Czech, Russian and Bulgarian

PROJECT NOTE

Elke Teich, Erich Steiner,
Renate Henschel and John Bateman[1]

Universität des Saarlandes
Institut für Angewandte Sprachwissenschaft sowie Übersetzen und Dolmetschen
66041 Saarbrücken

The AGILE (Automatic Generation of Instructions in Languages of Eastern Europe) project is an INCO (COPERNICUS) project to be started in the fall of 1997. It involves four partners from Eastern Europe—Charles University, Prague, the Russian Research Institute for AI, Moscow, the Bulgarian Academy of Sciences, Sofia and the Bulgarian software company DATECS Ltd—and two partners from Western Europe—the Information Technology Research Institute (ITRI) of the University of Brighton and the Institut für Angewandte Sprachwissenschaft sowie Übersetzen und Dolmetschen, Universität des Saarlandes, Saarbrücken.

The goal of the project is to develop a suite of software tools to assist technical writers in producing software documentation for CAD-CAM in Czech, Russian and Bulgarian. This involves building up the linguistic resources necessary for automatically generating text in these three languages and localizing and further developing the technical writing software of the DRAFTER project [Paris *et al.*, 1995, Paris and Vander Linden, 1996] for these three languages. The contributions of the project thus lie in the areas of computational linguistics, more precisely natural language generation, and authoring tools.

Natural Language Generation. The implementation of generation resources will be based on corpus analyses of CAD-CAM manuals in Czech, Russian and Bulgarian. The corpus analyses will follow attested methods of register analysis, such as e.g., [Biber, 1995], and functionally oriented discourse analysis, notably as developed in Systemic Functional Linguistics [Halliday and Hasan, 1976, Martin, 1992] and in the Prague School [Sgall *et al.*, 1986]. Since we attempt *resource sharing across languages*, it will be essential to conduct these analyses contrastively, i.e., relate the results of the monolingual analysis of one language to those of the other languages. Proceeding this way is supported by the KPML development environment for multilingual generation resources (see below).

In spite of the focus on the CAD-CAM domain, we intend to develop gen-

[1]Elke Teich is currently supported by the Australian-European Awards Program; current address: Macquarie University, Department of English, Linguistics and Media, North Ryde NSW 2109, Australia.–John Bateman is currently employed at the University of Stirling, Department of English Studies, Stirling FK9 4LA, UK.

eration resources that are as generic as possible in order to achieve reusability across application contexts. Also, as already mentioned, we attempt resource sharing across languages, following the *transfer comparison* method set out in [Halliday *et al.*, 1964]. According to this method, the linguistic categories and descriptions of one language serve as a basis for describing additional languages. This is a well tested method for building up generation resources first suggested by [Bateman *et al.*, 1991] and implemented in the KPML (Komet-Penman Multilingual) development environment [Bateman, 1997], when a rapid initial prototyping with languages newly to be covered is needed. KPML's linguistic resources have partly been built up by the transfer comparison method taking the NIGEL grammar of English as a basis [Matthiessen, 1995] and the system now covers covers substantial fragments of German [Teich, 1992, Grote, 1994a], Dutch [Degand, 1993], and French [Paris and Scott, 1996] and smaller (sublanguage) grammars of Japanese and Greek.

The concept of resource sharing across languages rests on the assumption that languages will always show commonalities *and* differences at the same time. For two grammars of two different languages to share a description then means that that description will have parts that are valid for both languages, but is also allowed to contain language-specific information that only applies to one of the languages but not the other. For instance, two languages that are typologically rather distant, such as English and Russian, can be described as sharing the same potential of engaging in symbolic interaction, encoded in the grammar in the system of *mood*, with the features declarative, interrogative, and imperative. We speak of the mood system being shared among English and Russian. At the syntagmatic, surface-syntactic level, these mood features are of course realized in different ways in English and Russian. For instance, polar interrogatives are realized in English with the help of the auxiliary 'do' and inversion of the Subject and the auxiliary. In Russian, in contrast, polar interrogatives are realized by intonation only - there is no difference in syntactic structure to declarative clauses. Both systemic commonalities and the realizational differences such as the one just exemplified can be accommodated in the systemic-functional-style descriptions of the KPML system.[2] It is in this sense that we speak of resource sharing among languages and it will be very interesting to see to what extent this kind of resource sharing will work among Germanic and Slavonic languages, and also what kinds of new contrastive-linguistic insights such a functional perspective of comparing languages might bring.

More generally, in terms of linguistic descriptive methods we follow functional linguistic theories of language, notably Systemic Functional Linguistics (SFL) and the Prague School. Of particular interest here will be attempting to accommodate Praguean-School-style descriptions, for instance of topic-focus articulation, into the systemically based KPML tactical generator kernel.

Apart from developing tactical generation resources for Czech, Russian and Bulgarian, a simple text planning mechanism for the text type at hand will be designed and implemented. To this end, the corpus analyses will have to put special focus on the relation between the correlation of grammatical selection and local (cohesion) and global (coherence) discourse phenomena. Including the level of text *and* taking a contrastive-linguistic perspective at the same time thus promises to shed more light on the question of interlingua in the domain of

[2]The system is freely available and can be downloaded from `http://www.darmstadt.gmd.de/publish/komet/kpml.html`.

discourse relations and text structuring.

Authoring tools. Hitherto the common practice in producing equivalent software documentation in more than one language typically involved the translation of a source text to a target language text. Recently, a change of this practice to producing equivalent text in multiple languages from the start can be observed. This calls for tools that support authoring in multiple languages, including style and grammar checking, spell checking, electronic dictionaries and thesauri etc. The existing tools, however, mainly offer assistence at clause and word level and are predominantly designed to be applied *after* the composition of a text. The functionalities of the tool set to be developed in AGILE go further in that the actual drafting of a document is supported (cf. the DRAFTER system [Paris *et al.*, 1995]). This comprises one set of tools that support the technical writer in building up a (language-independent) domain model, using, for instance, graphical interaction facilities as described in [Paris *et al.*, 1995], and a second type of tool that assists the technical writer in transforming the output of the first set of tools into draft documents in Czech, Russian and Bulgarian. This second type of tool is actually the multilingual generator, which produces parallel text directly from the user interface/domain model. The automatically produced text can be changed again by the user and the changes can be fed back to the system so that they can be carried out consistently throughout the text and across the different languages.

The system to be developed in AGILE will extensively build upon the ideas underlying DRAFTER and the results achieved in DRAFTER. The basic desiderata for a suppport drafting tool, such as support for knowledge re-use, support for alternative formulations, early drafts, propagation of changes throughout documents and languages are rather general desiderata that motivate the basic functionalities of the system and are language-independent. Thus, the basic DRAFTER system architecture is initially adopted for AGILE (cf. [Paris *et al.*, 1995] for a description of the components of the DRAFTER system).

To summarize, the results targetted in AGILE are both of a theoretical nature and have a practical value. On the theoretical side, new contrastive linguistic insights for the three Slavonic languages involved, not only on the grammatical level, but also on the level of discourse, are to be expected. These can then be related to accounts of the features of instructional text of other languages—for example, of English and German, for both of which extensive descriptions of the register of instructions exist (e.g., [Grote, 1994b, Rösner and Stede, 1994, Delin *et al.*, 1993, Delin *et al.*, 1994]. Furthermore, on the theoretical side, it will be interesting to see how the two functional theories of language we take as a linguistic-methodological basis—SFL and the Prague School—can complement each other. On the practical side, the three Eastern academic partners will have a linguistic resource for NL generation that is resuable for other applications and projects, and the commercial partner will be equipped with a piece of state-of-the-art writing software that can be experimented with also in other domains and applications.

References

[Bateman *et al.*, 1991] John A. Bateman, Christian M.I.M. Matthiessen, Keizo Nanri, and Licheng Zeng. The re-use of linguistic resources across languages in multilingual generation components. In *Proceedings of the 1991 International Joint Conference on Artificial Intelligence, Sydney, Australia*, volume 2, pages 966 – 971. Morgan Kaufmann Publishers, 1991.

[Bateman, 1997] John A. Bateman. *KPML Development Environment: multilingual linguistic resource development and sentence generation.* German National Center for Information Technology (GMD), Institute for Integrated Publication and Information Systems (IPSI), Darmstadt, Germany, March 1997. (Release 1.0).

[Biber, 1995] Douglas Biber. *Dimensions of register variation: a cross-linguistic comparison.* Cambridge University Press, 1995.

[Degand, 1993] Liesbeth Degand. Dutch grammar documentation. Technical report, GMD/Institut für Integrierte Publikations- und Informationssysteme, Darmstadt, Germany, 1993.

[Delin *et al.*, 1993] Judy Delin, Donia Scott, and Tony Hartley. Knowledge, intention, rhetoric: levels of variation in multilingual instructions. In Owen Rambow, editor, *Intentionality and structure in discourse relations*, pages 7 – 10. Association for Computational Linguistics, 1993. (Proceedings of a Workshop sponsored by the Special Interest Group on Generation, 21 June, 1993, Columbus, Ohio).

[Delin *et al.*, 1994] Judy Delin, Anthony Hartley, Cécile L. Paris, Donia Scott, and Keith Vander Linden. Expressing Procedural Relationships in Multilingual Instructions. In *Proceedings of the Seventh International Workshop on Natural Language Generation, Kennebunkport, Maine, USA, June 21-24, 1994*, pages 61 – 70, Kennebunkport, Maine, USA, 1994.

[Grote, 1994a] Brigitte Grote. Grammatial revision of the German prepositional phrase in KOMET. Technical report, GMD/Institut für Integrierte Publikations- und Informationssysteme, Darmstadt, Germany, May 1994.

[Grote, 1994b] Brigitte Grote. Linguistic properties of the text type 'instruction'. Technical report, Institut für Integrierte Publikations- und Informationssysteme (IPSI), GMD and ITRI, University of Brighton, Darmstadt, Germany and Brighton, UK, September 1994. technical report.

[Halliday and Hasan, 1976] Michael A.K. Halliday and Ruqaiya Hasan. *Cohesion in English.* Longman, London, 1976.

[Halliday *et al.*, 1964] Michael A.K. Halliday, A. McIntosh, and Peter Strevens. *The linguistic sciences and language teaching.* Longman, London, 1964.

[Martin, 1992] James R. Martin. *English text: systems and structure.* Benjamins, Amsterdam, 1992.

[Matthiessen, 1995] Christian M.I.M. Matthiessen. *Lexicogrammatical cartography: English systems.* International Language Science Publishers, Tokyo, Taipei and Dallas, 1995.

[Paris and Scott, 1996] Cécile L. Paris and Donia Scott. Stylistic variation in multilingual instructions. In *Proceedings of the Seventh International Workshop on Natural Language Generation*, Kennebunkport, Maine, USA, June 21-24, 1994.

[Paris and Vander Linden, 1996] Cécile L. Paris and Keith Vander Linden. DRAFTER: an interactive support tool for writing multilingual instructions. *IEEE Computer*, 1996.

[Paris *et al.*, 1995] Cécile Paris, Keith Vander Linden, Markus Fischer, Anthony Hartley, Lyn Pemberton, Richard Power, and Donia Scott. A Support Tool for Writing Multilingual Instructions. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI) 1995*, pages 1398 – 1404, Montréal, Canada, 1995.

[Rösner and Stede, 1994] Dietmar Rösner and Manfred Stede. Generating multilingual documents from a knowledge base: the TECHDOC project. In *Proceedings of the 15th. International Conference on Computational Linguistics (*COLING 94*)*, volume I, pages 339 – 346, Kyoto, Japan, 1994.

[Sgall *et al.*, 1986] Petr Sgall, Eva Hajičová, and J. Panevová. *The Meaning of the Sentence in Its Semantic and Pragmatic Aspects.* Reidel Publishing Company, Dordrecht, 1986.

[Teich, 1992] Elke Teich. KOMET: Grammar Documentation. Technical report, GMD/Institut für Integrierte Publikations- und Informationssysteme, Darmstadt, Germany, 1992.

EFFENDI

EFizientes FormulierEN von DIalogbeiträgen

Project Overview

Peter Poller
DFKI GmbH
German Research Center for Artificial Intelligence
Stuhlsatzenhausweg 3
D–66123 Saarbrücken
Germany
E–Mail: poller@dfki.uni-sb.de

June 20, 1997

Abstract

This article gives a short overview of the particular modules that have been developed in the EFFENDI–project. The goal of the project was the development and implementation of an efficient, multilingual, incremental syntactic generation component for the system responses of a speech dialogue system for train inquiries based on the incremental syntactic generator TAG–GEN.

1 The EFFENDI–Project

The EFFENDI–project was a cooperation between the DFKI Saarbrücken and Daimler–Benz research Ulm. The project goal was the development of a syntactic generation component that is especially adapted to the specific needs of a speech dialogue system on syntactic generation. This includes

general requirements on the efficiency of the generation itself as well as specific generator features that are necessary to allow for a natural behaviour of the overall dialogue system.

The EFFENDI generator is based on the incremental syntactic generator TAG–GEN ([Harbusch et al. 1991]) that has been developed inside the WIP–project ([Wahlster et al. 1992], [Wahlster et al. 1993]) at DFKI. The following sections give short summaries of the particular features that have been developed and implemented to form the "dialogue–system–specific" EFFENDI–generator based on TAG–GEN. Detailed descriptions of the particular features can be found in the respective citations. Details about the core generator TAG–GEN itself can be found in, e.g., [Harbusch et al. 1991], [Kilger & Finkler 1992] or [Kilger 1994].

2 Templates

One of the main requirements on the generation component of a dialogue system is efficiency to allow for dialogues in real time. First, incremental syntactic generation itself increases the reaction time of the overall system because the output can already start before the generator input is completely processed. Additionally, there are specific formulations or utterances that occur repeatedly in system answers which can be reused for speeding up computation. Therefore EFFENDI contains a template processing module which allows for the reuse of syntactic structures instead of their repeated generation ([Poller & Heisterkamp 1995]). The most important features of this module are:

- There are syntactic structures that may be reused for the generation of system answers (templates).
- Templates can either be predefined or be dynamically extracted from system answers in an ongoing dialogue.
- Templates can be uniquely identified by the generator and the dialogue management component.
- Template–based generation and "free" generation can be mixed within the same utterance.
- There are three different kinds of templates: complete utterances, sentence parts (e.g. a prepositional phrase) and patterns (i.e. sentence schemas with variables).

In the average case, generation with templates is between 15 % and 60 % faster than "free" generation of the same utterance depending on the ratio between template structures and freely generated structures.

3 Tools

The first EFFENDI demonstrator was designed for the application in a dialogue system for train inquiries. Nevertheless, EFFENDI is a generation module that is independent from the application system. In order to adapt the generator to a new application, only its knowledge bases (grammar, lexicon, input interface) have to be extended by domain–specific knowledge. In order to support the developer to access the knowledge bases of the EFFENDI system there are elementary, appropriate tools for each knowledge base ([Poller & Heisterkamp 1996]) which allow for easier extensions or modifications on it.

Furthermore, there is a graphic user interface for displaying detailed information about the individual structures produced by the generator and the input and output times of the individual generator input elements.

4 Interruptability

There are some special features of the dialogue system necessary to allow for natural dialogues. Most of these features have to be realized in the generation component because its output is the interface of the whole system to the user.

In natural dialogues the dialogue partners sometimes interrupt each other, e.g., in cases of obvious misunderstandings. Accordingly, one of the special features of the EFFENDI generator is its interruptability which means that the user can interrupt a running generation process before its termination ([Poller & Heisterkamp 1996]). After having been interrupted the generator expects instructions on how to continue. There are two possibilities namely the continuation of the interrupted utterance or the generation of a completely new utterance.

5 Reformulation

Another important feature contributing to the naturalness of the dialogue system is a special routine for reformulations of system utterances which

the user did not understand. In EFFENDI there are two ways of reformulation realized ([Poller & Heisterkamp 1996]). The first is "grammatical" reformulation which means that the generator is able to generate another word order than before for the same message. The second is "lexical" reformulation. This means that the input interface of the generator tries to make another word choice than before for the same semantic input concepts based on synonyms.

6 Ellipses, Anaphors, Focus

In natural dialogues all dialogue partners use the context in order to produce short but appropriate utterances which include references to previously mentionend concepts (anaphors) or in which some elements are omitted (ellipses). In both cases a unique mapping to the omitted or referred concepts is possible which ensures the understandability. Furthermore there may be concepts especially marked as focus to guide the user's attention.

For all these cases the semantic generator input will contain accordingly marked semantic input concepts which are interpreted as generator instructions ([Poller & Heisterkamp 1996]). EFFENDI then generates appropriate formulations which will differ from the "normal" verbalization of the concepts. Focussed elements are topicalized but if that is impossible because of linearization constrains they are especially marked as focussed elements in the interface protocol to the synthesis component (cf. the following section).

7 Synthesis Interface

The speech output of the dialogue system is the only "visible" part for the user. Therefore the quality of the speech output is very important for the user acceptance of the system. In order to produce natural sounding speech, the synthesizer requires not only knowledge about what words to say in what order, but also information about how these words are structurally related to each other. The latter information is expressed acoustically in the form of prosody, i.e. how the voice raises and falls during an utterance, the rhythm, the setting/placing of pauses, etc. Prosody is also influenced by the properties associated with given words in the context of an utterance, e.g. the focus of a sentence or certain emphatic elements.

In cooperation with the speech synthesis group at Daimler–Benz we developed a special input representation (so called *interface protocol*) for the

synthesis component that allows for conveying this information to the synthesis component ([Poller et al. 1996], [Poller & Heisterkamp 1996]). The protocol contains the following information which can be found in the example below:

- the type of each sentence to be uttered (`$..`),
- a specification of each atomic group (`**..`) along with its associated group category (`#..`) including a list of all words in the order to be uttered along with their associated categories in parentheses and special attributes in square brackets (e.g. focus, contrast, ...) if any, and
- a description of logical (syntactically motivated) connections between atomic groups expressed by relative pointers from a group to following (`<+.`) or preceding (`<-.`) groups.

The interface protocol for the sentence "Sie möchten wissen, wann der nächste Zug nach Ulm fährt" (literally: You would like to know, when the next train to Ulm goes.) where "Ulm" marks the focus looks like this:

```
$AS
 **   Sie(PRON)                                  #SP   >+1
 **   möchten(H) wissen(VU)                      #VP   >-1   >+1
 **   wann(KONJ) der(DET-S) nächste(ADJ) Zug(N)  #KP   >-1   >+2
 **   nach(PRAEP) Ulm(N)[focus]                  #PP   >+1
 **   fährt(V)                                   #VP   >-2   >-1
```

8 Self–Repair by elliptical resumption

The incrementality of the generator output sometimes requires intrasentential corrections. In case of written output incorrect phrases can sometimes be overwritten, while spoken output cannot be made undone. Instead, corrected phrases have to be attached appropriately to the already spoken incorrect output. In this case some parts of the output have to be repeated several times in order to ensure that the whole output remains understandable and still has a correct word order. Observations show that most of the corrections become necessary because of the delay in incremental input consumption and the fact that generation and output production may begin before all input elements have been consumed. So, it is possible that a

previously unknown input element has to be placed before already uttered elements which requires an appropriate correction of the already produced output.

To reduce the number of overt corrections including partial repetitions to a limited extend we developed a special routine that avoids corrections for some specific cases ([Poller & Heisterkamp 1996]) as soon as it becomes obvious during generation that a repair of the output produced so far will become necessary. Instead the generator is stopped immediately and restarted with the same input elements but in this resumption all already uttered input elements are marked explicitely as elliptic so that previously uttered parts will not be produced again. In this way invalid word orders become possible for the complete output because output elements of resumpted generator calls are always attached to the output elements produced so far. Nevertheless, the complete output remains understandable, so we decided to accept such invalid word orders in favor of avoiding intrasentential corrections. The following example shows the possible effects of such a behaviour for the incremental generation of the german sentence "Sie möchten um 1 Uhr nach Ulm fahren." ("You want to leave to Ulm at 1 o'clock"):

```
Sie moechten fahren

REPAIR DETECTED --> DOING ELLIPTICAL RESUMPTION INSTEAD !!

um 1 Uhr nach Ulm.
```

In this case the complete verb phrase has been uttered before the two PP's are known. Instead of generating an especially corrected output in which the PP's are placed before the infinitive "fahren" the elliptical resumption only utters the two PP's because the already uttered parts become ellipses and therefore are not uttered again.

References

[Harbusch et al. 1991] **K. Harbusch, W. Finkler, A. Schauder**: *Incremental Syntax Generation with Tree Adjoining Grammars*, in: Brauer, W. Hernández, D. (eds.): 4^{th} International GI Congress on Knowledge–Based Systems, 1991, 363–374; also as: DFKI Research Report RR–91–25, DFKI, Saarbrücken, 1991.

[Kilger 1994] **A. Kilger**: *Using UTAGs for Incremental and Parallel Generation*, in: Computational Intelligence, 10 (1994) 4, 591–603.

[Kilger & Finkler 1992] **A. Kilger, W. Finkler**: *Effects of Incremental Output on Incremental Natural Language Generation*, in: B. Neumann (ed.): 10^{th} European Conference on Artifical Intelligence, August 3–7, 1992, Vienna, Austria, 505–507.

[Poller & Heisterkamp 1995] **P. Poller, P. Heisterkamp**: *EFFENDI – Effizientes Formulieren von Dialogbeiträgen – Zwischenbericht*, Technischer Bericht Nr. F3–95–014, Daimler–Benz Forschung und Technik, Ulm, 1995.

[Poller & Heisterkamp 1996] **P. Poller, P. Heisterkamp**: *EFFENDI – Effizientes Formulieren von Dialogbeiträgen – Bericht zum Projektende – Handbuch zur SIL–Schnittstelle*, Technischer Bericht Nr. F3–96–014, Daimler–Benz Forschung und Technik, Ulm, 1996.

[Poller et al. 1996] **P. Poller, P. Heisterkamp, D. Stall**: *An Interface Protocol from the Speech Generator to the Speech Synthesis Module of a Dialogue System*, in: John A. Bateman (ed.): Speech Generation in Multimodal Information Systems and its Practical Applications, Proceedings of the 2^{nd} 'SPEAK!' Workshop, GMD–Studien Nr. 302, IPSI, GMD, Darmstadt, 1996.

[Wahlster et al. 1992] **W. Wahlster, E. André, S. Bandyopadhyay, W. Graf, T. Rist**: *WIP: The Coordinated Generation of Multimodal Presentations from a Common Representation*, in: A. Ortony, J. Slack, O. Stock (eds.), Communication from an Artifical Intelligence Perspective: Theoretical and Applied Issues, pp. 121–144, Heidelberg, Springer, 1992; also as: DFKI Research Report RR–91–08, DFKI, Saarbrücken, 1991.

[Wahlster et al. 1993] **W. Wahlster, E. André, W. Finkler, H. J. Profitlich, T. Rist**: *Plan–based Integration of Natural Language and Graphics Generation*, Artifical Intelligence, 63: pp. 387–427, 1993; also as: DFKI Reserach Report RR–93–02, DFKI, Saarbrücken, 1993.

Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

-Bibliothek, Information
und Dokumentation (BID)-
PF 2080
67608 Kaiserslautern
FRG

Telefon (0631) 205-3506
Telefax (0631) 205-3210

e-mail
dfkibib@dfki.uni-kl.de
WWW
http://www.dfki.uni-
sb.de/dfkibib

Veröffentlichungen des DFKI

Die folgenden DFKI Veröffentlichungen sowie die aktuelle Liste von allen bisher erschienenen Publikationen können von der oben angegebenen Adresse oder (so sie als per ftp erhaeltlich angemerkt sind) per anonymous ftp von ftp.dfki.uni-kl.de (131.246.241.100) im Verzeichnis pub/Publications bezogen werden. Die Berichte werden, wenn nicht anders gekennzeichnet, kostenlos abgegeben.

DFKI Publications

The following DFKI publications or the list of all published papers so far are obtainable from the above address or (if they are marked as obtainable by ftp) by anonymous ftp from ftp.dfki.uni-kl.de (131.246.241.100) in the directory pub/Publications.
The reports are distributed free of charge except where otherwise noted.

DFKI Research Reports

1997

RR-97-08
Stefan Müller
Complement Extraction Lexical Rules and Argument Attraction
14 pages

RR-97-07
Stefan Müller
Yet Another Paper about Partial Verb Phrase Fronting in German
26 pages

RR-97-06
Stefan Müller
Scrambling in German – Extraction into the *Mittelfeld*
24 pages

RR-97-04
Serge Autexier, Dieter Hutter
Parameterized Abstractions used for Proof-Planning
13 pages

RR-97-03
Dieter Hutter
Using Rippling to Prove the Termination of Algorithms
15 pages

RR-97-02
Stephan Busemann, Thierry Declerck, Abdel Kader Diagne, Luca Dini, Judith Klein, Sven Schmeier
Natural Language Dialogue Service for Appointment Scheduling Agents
15 pages

RR-97-01
Erica Melis, Claus Sengler
Analogy in Verification of State-Based Specifications: First Results
12 pages

1996

RR-96-06
Claus Sengler
Case Studies of Non-Freely Generated Data Types
200 pages

RR-96-05
Stephan Busemann
Best-First Surface Realization
11 pages

RR-96-04
Christoph G. Jung, Klaus Fischer, Alastair Burt
Multi-Agent Planning
Using an *Abductive*
EVENT CALCULUS
114 pages

RR-96-03
Günter Neumann
Interleaving
Natural Language Parsing and Generation
Through Uniform Processing
51 pages

RR-96-02
E.André, J. Müller , T.Rist:
PPP-Persona: Ein objektorientierter Multimedia-Präsentationsagent
14 Seiten

RR-96-01
Claus Sengler
Induction on Non-Freely Generated Data Types
188 pages

1995

RR-95-20
Hans-Ulrich Krieger
Typed Feature Structures, Definite Equivalences, Greatest Model Semantics, and Nonmonotonicity
27 pages

RR-95-19
Abdel Kader Diagne, Walter Kasper, Hans-Ulrich Krieger
Distributed Parsing With HPSG Grammar
20 pages

RR-95-18
Hans-Ulrich Krieger, Ulrich Schäfer
Efficient Parameterizable Type Expansion for Typed Feature Formalisms
19 pages

RR-95-17
Hans-Ulrich Krieger
Classification and Representation of Types in TDL
17 pages

RR-95-16
Martin Müller, Tobias Van Roy
Title not set
0 pages

Note: The author(s) were unable to deliver this document for printing before the end of the year. It will be printed next year.

RR-95-15
Joachim Niehren, Tobias Van Roy
Title not set
0 pages

Note: The author(s) were unable to deliver this document for printing before the end of the year. It will be printed next year.

RR-95-14
Joachim Niehren
Functional Computation as Concurrent Computation
50 pages

RR-95-13
Werner Stephan, Susanne Biundo
Deduction-based Refinement Planning
14 pages

RR-95-12
Walter Hower, Winfried H. Graf
Research in Constraint-Based Layout, Visualization, CAD, and Related Topics: A Bibliographical Survey
33 pages

RR-95-11
Anne Kilger, Wolgang Finkler
Incremental Generation for Real-Time Applications
47 pages

RR-95-10
Gert Smolka
The Oz Programming Model
23 pages

RR-95-09
M. Buchheit, F. M. Donini, W. Nutt, A. Schaerf
A Refined Architecture for Terminological Systems: Terminology = Schema + Views
71 pages

RR-95-08
Michael Mehl, Ralf Scheidhauer, Christian Schulte
An Abstract Machine for Oz
23 pages

RR-95-07
Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, Werner Nutt
The Complexity of Concept Languages
57 pages

RR-95-06
Bernd Kiefer, Thomas Fettig
FEGRAMED
An interactive Graphics Editor for Feature Structures
37 pages

RR-95-05
Rolf Backofen, James Rogers, K. Vijay-Shanker
A First-Order Axiomatization of the Theory of Finite Trees
35 pages

RR-95-04
M. Buchheit, H.-J. Bürckert, B. Hollunder, A. Laux, W. Nutt,
M. Wójcik
Task Acquisition with a Description Logic Reasoner
17 pages

RR-95-03
Stephan Baumann, Michael Malburg, Hans-Guenther Hein, Rainer Hoch,
Thomas Kieninger, Norbert Kuhn
Document Analysis at DFKI
Part 2: Information Extraction
40 pages

RR-95-02
Majdi Ben Hadj Ali, Frank Fein, Frank Hoenes, Thorsten Jaeger,
Achim Weigel
Document Analysis at DFKI
Part 1: Image Analysis and Text Recognition
69 pages

RR-95-01
Klaus Fischer, Jörg P. Müller, Markus Pischel
Cooperative Transportation Scheduling
an application Domain for DAI
31 pages

1994

RR-94-39
Hans-Ulrich Krieger
Typed Feature Formalisms as a Common Basis for Linguistic Specification.
21 pages

RR-94-38
Hans Uszkoreit, Rolf Backofen, Stephan Busemann, Abdel Kader Diagne,
Elizabeth A. Hinkelman, Walter Kasper, Bernd Kiefer,
Hans-Ulrich Krieger,
Klaus Netter, Günter Neumann, Stephan Oepen, Stephen P. Spackman.
DISCO–An HPSG-based NLP System and its Application for Appointment Scheduling.
13 pages

RR-94-37
Hans-Ulrich Krieger, Ulrich Schäfer
TDL - A Type Description Language for HPSG, Part 1: Overview.
54 pages

RR-94-36
Manfred Meyer
Issues in Concurrent Knowledge Engineering. Knowledge Base and Knowledge Share Evolution.
17 pages

RR-94-35
Rolf Backofen
A Complete Axiomatization of a Theory with Feature and Arity Constraints
49 pages

RR-94-34
Stephan Busemann, Stephan Oepen, Elizabeth A. Hinkelman,
Günter Neumann, Hans Uszkoreit
COSMA – Multi-Participant NL Interaction for Appointment Scheduling
80 pages

RR-94-33
Franz Baader, Armin Laux
Terminological Logics with Modal Operators
29 pages

RR-94-31
Otto Kühn, Volker Becker, Georg Lohse, Philipp Neumann
Integrated Knowledge Utilization and Evolution for the Conservation of Corporate Know-How
17 pages

RR-94-23
Gert Smolka
The Definition of Kernel Oz
53 pages

RR-94-20
Christian Schulte, Gert Smolka, Jörg Würtz
Encapsulated Search and Constraint Programming in Oz
21 pages

RR-94-19
Rainer Hoch
Using IR Techniques for Text Classification in Document Analysis
16 pages

RR-94-18
Rolf Backofen, Ralf Treinen
How to Win a Game with Features
18 pages

RR-94-17
Georg Struth
Philosophical Logics—A Survey and a Bibliography
58 pages

RR-94-16
Gert Smolka
A Foundation for Higher-order Concurrent Constraint Programming
26 pages

RR-94-15
Winfried H. Graf, Stefan Neurohr
Using Graphical Style and Visibility Constraints for a *Meaningful Layout in Visual Programming Interfaces*
20 pages

RR-94-14
Harold Boley, Ulrich Buhrmann, Christof Kremer
Towards a Sharable Knowledge Base on Recyclable Plastics
14 pages

RR-94-13
Jana Koehler
Planning from Second Principles—A Logic-based Approach
49 pages

RR-94-12
Hubert Comon, Ralf Treinen
Ordering Constraints on Trees
34 pages

RR-94-11
Knut Hinkelmann
A Consequence Finding Approach for Feature Recognition in CAPP
18 pages

RR-94-10
Knut Hinkelmann, Helge Hintze
Computing Cost Estimates for Proof Strategies
22 pages

RR-94-08
Otto Kühn, Björn Höfling
Conserving Corporate Knowledge for Crankshaft Design
17 pages

RR-94-07
Harold Boley
Finite Domains and Exclusions as First-Class Citizens
25 pages

RR-94-06
Dietmar Dengler
An Adaptive Deductive Planning System
17 pages

RR-94-05
Franz Schmalhofer, J. Stuart Aitken, Lyle E. Bourne jr.
Beyond the Knowledge Level: Descriptions of Rational Behavior for Sharing and Reuse
81 pages

RR-94-03
Gert Smolka
A Calculus for Higher-Order Concurrent Constraint Programming with Deep Guards
34 pages

RR-94-02
Elisabeth André, Thomas Rist
Von Textgeneratoren zu Intellimedia-Präsentationssystemen
22 Seiten

RR-94-01
Elisabeth André, Thomas Rist
Multimedia Presentations: The Support of Passive and Active Viewing
15 pages

DFKI Technical Memos

1997

TM-97-01
Markus Perling
GeneTS: A Relational-Functional Genetic Algorithm for the Traveling Salesman Problem
26 pages

1996

TM-96-02
Harold Boley
Knowledge Bases in the World Wide Web:
A Challenge for Logic Programming
8 pages

TM-96-01
Gerd Kamp, Holger Wache
CTL — a description Logic with expressive concrete domains
19 pages

1995

TM-95-04
Klaus Schmid
Creative Problem Solving
and
Automated Discovery
— An Analysis of Psychological and AI Research - -
152 pages

TM-95-03
Andreas Abecker, Harold Boley, Knut Hinkelmann, Holger Wache,
Franz Schmalhofer
An Environment for Exploring and Validating Declarative Knowledge
11 pages

TM-95-02
Michael Sintek
FLIP: Functional-plus-Logic Programming
on an Integrated Platform
106 pages

TM-95-01
Martin Buchheit, Rüdiger Klein, Werner Nutt
Constructive Problem Solving: A Model Construction Approach towards Configuration
34 pages

1994

TM-94-05
Klaus Fischer, Jörg P. Müller, Markus Pischel
Unifying Control in a Layered Agent Architecture
27 pages

TM-94-04
Cornelia Fischer
PAntUDE – An Anti-Unification Algorithm for Expressing Refined Generalizations
22 pages

TM-94-03
Victoria Hall
Uncertainty-Valued Horn Clauses
31 pages

TM-94-02
Rainer Bleisinger, Berthold Kröll
Representation of Non-Convex Time Intervals and Propagation of Non-Convex Relations
11 pages

TM-94-01
Rainer Bleisinger, Klaus-Peter Gores
Text Skimming as a Part in Paper Document Understanding
14 pages

DFKI Documents

1997

D-97-06
Tilman Becker, Stephan Busemann, Wolfgang Finkler
DFKI Workshop on Natural Language Generation
67 pages

D-97-04
Claudia Wenzel, Markus Junker
Entwurf einer Patternbeschreibungssprache
für die Informationsextraktion
in der Dokumentanalyse
24 Seiten

D-97-03
Andreas Abecker, Stefan Decker, Knut Hinkelmann, Ulrich Reimer
Proceedings of the Workshop „Knowledge-Based Systems for Knowledge Management in Enterprises“ 97
167 pages

D-97-02
Tilman Becker, Hans-Ulrich Krieger
Proceedings of the Fifth Meeting on Mathematics of Language (MOL5)
168 pages

D-97-01
Thomas Malik
NetGLTool Benutzeranleitung
40 Seiten

1996

D-96-07
Technical Staff
DFKI Jahresbericht 1995
55 Seiten

Note: This document is no longer available in printed form.

D-96-06
Klaus Fischer (Ed.)
Working Notes of the KI'96 Workshop on Agent-Oriented Programming and Distributed Systems
63 pages

D-96-05
Martin Schaaf
Ein Framework zur Erstellung verteilter Anwendungen
94 pages

D-96-04
Franz Baader, Hans-Jürgen Bürckert, Andreas Günter, Werner Nutt (Hrsg.)
Proceedings of the Workshop on Knowledge Representation and Configuration WRKP'96
83 pages

D-96-03
Winfried Tautges
Der DESIGN-ANALYZER - Decision Support im Designprozess
75 Seiten

D-96-01
Klaus Fischer, Darius Schier
Ein Multiagentenansatz zum Lösen von Fleet-Scheduling-Problemen
72 Seiten

1995

D-95-12
F. Baader, M. Buchheit, M. A. Jeusfeld, W. Nutt (Eds.)
Working Notes of the KI'95 Workshop:
KRDB-95 - Reasoning about Structured Objects:
Knowledge Representation Meets Databases
61 pages

D-95-11
Stephan Busemann, Iris Merget
Eine Untersuchung kommerzieller Terminverwaltungssoftware im Hinblick auf die Kopplung mit natürlichsprachlichen Systemen
32 Seiten

D-95-10
Volker Ehresmann
Integration ressourcen-orientierter Techniken in das wissensbasierte Konfigurierungssystem TOOCON
108 Seiten

D-95-09
Antonio Krüger
PROXIMA: Ein System zur Generierung graphischer Abstraktionen
120 Seiten

D-95-08
Technical Staff
DFKI Jahresbericht 1994
63 Seiten

Note: This document is no longer available in printed form.

D-95-07
Ottmar Lutzy
Morphic - Plus
Ein morphologisches Analyseprogramm für die deutsche Flexionsmorphologie und Komposita-Analyse
74 Seiten

D-95-06
Markus Steffens, Ansgar Bernardi
Integriertes Produktmodell für Behälter aus Faserverbundwerkstoffen
48 Seiten

D-95-05
Georg Schneider
Eine Werkbank zur Erzeugung von 3D-Illustrationen
157 Seiten

D-95-04
Victoria Hall
Integration von Sorten als ausgezeichnete taxonomische Prädikate in eine relational-funktionale Sprache
56 Seiten

D-95-03
Christoph Endres, Lars Klein, Markus Meyer
Implementierung und Erweiterung der Sprache $\mathcal{ALCP}$
110 Seiten

D-95-02
Andreas Butz
BETTY
Ein System zur Planung und Generierung informativer Animationssequenzen
95 Seiten

D-95-01
Susanne Biundo, Wolfgang Tank (Hrsg.)
PuK-95, Beiträge zum 9. Workshop „Planen und Konfigurieren", Februar 1995
169 Seiten

Note: This document is available for a nominal charge of 25 DM (or 15 US-$).

1994

D-94-15
Stephan Oepen
German Nominal Syntax in HPSG
— On Syntactic Categories and Syntagmatic Relations —
80 pages

D-94-14
Hans-Ulrich Krieger, Ulrich Schäfer
TDL - A Type Description Language for HPSG, Part 2: User Guide.
72 pages

D-94-12
Arthur Sehn, Serge Autexier (Hrsg.)
Proceedings des Studentenprogramms der 18. Deutschen Jahrestagung für Künstliche Intelligenz KI-94
69 Seiten

D-94-11
F. Baader, M. Buchheit, M. A. Jeusfeld, W. Nutt (Eds.)
Working Notes of the KI'94 Workshop: KRDB'94 - Reasoning about Structured Objects: Knowledge Representation Meets Databases
65 pages

Note: This document is no longer available in printed form.

D-94-10
F. Baader, M. Lenzerini, W. Nutt, P. F. Patel-Schneider (Eds.)
Working Notes of the 1994 International Workshop on Description Logics
118 pages

Note: This document is available for a nominal charge of 25 DM (or 15 US-$).

D-94-09
Technical Staff
DFKI Wissenschaftlich-Technischer Jahresbericht 1993
145 Seiten

D-94-08
Harald Feibel
IGLOO 1.0 - Eine grafikunterstützte Beweisentwicklungsumgebung
58 Seiten

D-94-07
Claudia Wenzel, Rainer Hoch
Eine Übersicht über Information Retrieval (IR) und NLP-Verfahren zur Klassifikation von Texten
25 Seiten

Note: This document is no longer available in printed form.

D-94-06
Ulrich Buhrmann
Erstellung einer deklarativen Wissensbasis über recyclingrelevante Materialien
117 Seiten

D-94-04
Franz Schmalhofer, Ludger van Elst
Entwicklung von Expertensystemen: Prototypen, Tiefenmodellierung und kooperative Wissensevolution
22 Seiten

D-94-03
Franz Schmalhofer
Maschinelles Lernen: Eine kognitionswissenschaftliche Betrachtung
54 Seiten

Note: This document is no longer available in printed form.

D-94-02
Markus Steffens
Wissenserhebung und Analyse zum Entwicklungsprozeß eines Druckbehälters aus Faserverbundstoff
90 pages

D-94-01
Josua Boon (Ed.)
DFKI-Publications: The First Four Years
1990 - 1993
75 pages

DFKI Workshop on Natural Language Generation

Tilman Becker, Stephan Busemann, Wolfgang Finkler (eds.)

D-97-06
Document