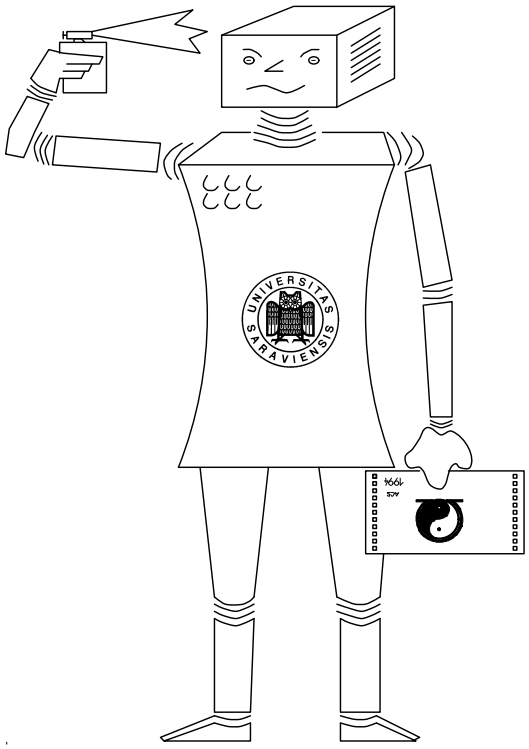
UNIVERSITÄT DES SAARLANDES
FACHBEREICH INFORMATIK
D-66041 SAARBRÜCKEN
GERMANY

WWW: http://js-sfbsun.cs.uni-sb.de/pub/www/



Generating Multilingual Proofs

Xiaorong Huang and Armin Fiedler

Published as: Proc. of IJCAI-95 Workshop on Multilingual Text Generation

Generating Multilingual Proofs

Xiaorong Huang, Armin Fiedler
Fachbereich Informatik, Universität des Saarlandes
Postfach 15 11 50, D-66041 Saarbrücken, Germany
e-mail: {huang,afiedler}@cs.uni-sb.de

Abstract

This paper outlines the microplanner of *PROVERB*, a system that generates multilingual text from machine-found mathematical proofs. The main representational vehicle is the text structure proposed by Meteer. Following Panaget, we also distinguish between the ideational and the textual semantic categories, and use the upper model to replace the former. Based on this, a further extension is made to support aggregation before realization decisions are made. While our the framework of our macroplanner is kept language independent, our microplanner draws on language specific linguistic sources such as realization classes and lexicon. Since English and German are the first two languages to be generated and because the sublanguage of our mathematical domain is relatively limited, the upper model and the textual semantic categories are designed to cope with both languages. Since the work reported is still in progress, we also discuss open problems we are facing.

1 Introduction

Most current natural language generation systems adopt a two-staged approach: Firstly a text planner selects and organizes information to be communicated in a goal directed way. Then the text plan is passed over to the realization component, which carries out the plan and verbalizes the messages. However, there is usually a gap between a text plan and a complete specification of sentences in a particular language [Hov92]. Apart from a few exceptions [Met91, HLM+92], this "generation gap" is either ignored altogether or handled in a simplistic way. As a result, few realization components are able to produce high quality texts in a stable way. As indicated in [Hov92], a rather sophisticated architecture for sentence planning is needed to bridge this gap.

This paper reports ongoing development of a microplanner for *PROVERB*, a system that verbalizes machine-found proofs [Hua94a, Hua94b]. Our microplanner is mainly designed to address two goals. First it should order and group clause-sized message elements generated by the macroplanner in a flexible and dynamic way to produce more coherent text. Second its architecture should be an uniform mechanism supporting multilingual verbalization of machine-found proofs. In the current prototype, the microplanner is designed to produce formatted text in two fairly similar languages, namely English and German. We are planning to integrate Chinese, a language with significantly distinct linguistic structures.

The architecture based on the Meaning-Text Theory [IKP91] and the architecture based on the systemic-functional linguist theory [BKMW90, BMNZ91] are probably two of the most explicitly formulated approaches towards multilingual text generation. To support microplanning operations such as aggregation [FH95] and sentence scoping, however, we adopt an intermediate representation called *text structure* first proposed by Meteer [Met92]. We have extended this approach in two ways: first following [Pan94] the ideational part of Meteer's semantic categories are replaced by the *upper model* [BKMW90]. Second instead of treating single nodes in a text structure as atomic elements, we explicitly represent compound constituent structures of upper model objects and specify our aggregation operations in terms of such compound structures. While the macroplanner of *PROVERB* delivers language insensitive messages, both our microplanner and our realizer [KF95] are multilingual, differing from systems that employ multiple realizers [IKKP90].

After a brief sketch of our macroplanner, we will provide an overview of our microplanner in this paper. Emphasis is laid upon different levels of representation within our text structure, as well as their role in multilingual generation. We want to indicate that we are still at the prototype stage and are experimenting with this architecture and the representations in concern.

2 The Macroplanner of *PROVERB*

The macroplanner of *PROVERB* combines *hierarchical planning* with *local organization* in a uniform planning framework [Hua94a]. The hierarchical planning is realized by so-called top-down presentation operators that split the task of presenting a particular proof into subtasks of presenting subproofs. While the overall planning mechanism is similar to the RST-based planning approach, the planning operators resemble the schemata in *schema-based* planning. The output of the macroplanner is an ordered sequence of *proof communicative acts* (PCAs).

PCAs are the primitive actions planned during the macroplanning to achieve communicative goals. Like speech acts, PCAs can be defined in terms of the communicative goals they fulfill as well as in terms of their possible verbalizations. Based on an analysis of proofs in mathematical textbooks, there are mainly two types of goals a PCA is generated to achieve:

- *Conveying a step of derivation:* In terms of rhetorical relations, PCAs in this category represent a variation of the rhetorical relation *derive*. Below is an example of the simplest PCA of this sort called `Derive`.

  (`Derive Reasons:` $(a \in F,\ F \subseteq G)$
  `Method: def-subset`
  `Conclusion:` $a \in G$)

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

  > "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."

- *Updating the global attentional structure:* These PCAs either convey a partial plan for the forthcoming discourse or signal the end of a subproof. PCAs of this sort are also called *meta-comments* [Zuk91].

The PCA

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

produces the verbalization:

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

Because the sublanguage of our mathematical domain is relatively limited and English and German are closely related, PCAs are currently kept language insensitive.

3 Text Structure in *PROVERB*

In order to bridge the generation gap between the representation in the application program and the linguistic resources provided by the language under consideration, Meteer proposed a new intermediate level of representation for text planning [Met91, Met92]. This so-called *text structure* reflects linguistic constraints, while abstracting away from syntactic detail. Text structure is organized as a tree, in which each node represents a constituent of the text. It is defined in terms of three types of abstract linguistic resources, namely

- *Constituency:* The elements of a text are hierarchically organized into constituents, which may range in size from a text or a paragraph to a single word.
- *Structural relations among constituents:* The nodes of a text structure are labeled with the semantic category the constituent expresses and with the structural relations to its parent and its children, respectively.
- *Semantic category the constituent expresses:* It reflects combinations of linguistic features with the meaning of the constituent.

Depending on the structural relations, there are two types of basic subtrees (see Figure 1):

- An atomic *kernel subtree* with a head at the root and arguments as children, representing basically a predicate/argument structure.
- A *composite subtree* that allows incremental extension, that is new children can be added. 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, whereas the second has two or more *coordinated* children and stands for parataxis.

Panaget argued that Meteer's hierarchy of semantic categories mixes ideational and textual constraints according to the systemic linguistic theory [Pan94]. Furthermore he found that the textual part of the hierarchy is too abstract to treat phenomena that need syntactic information. Therefore, he split Meteer's hierarchy of semantic categories into two orthogonal hierarchies: an ideational semantic hierarchy based on the *upper model* defined in [BKMW90], and a textual semantic hierarchy which he calls *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. A concept may

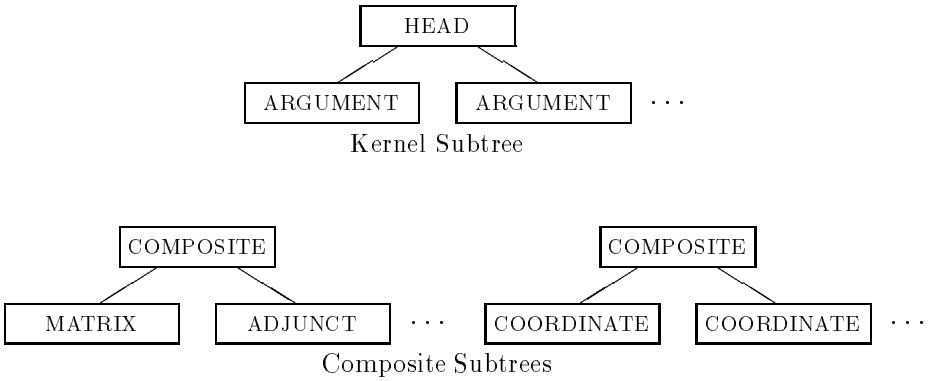


Figure 1: Subtree types

have required and optional arguments. Objects that may fill these slots are restricted in terms of upper model concepts. For instance, the concept *ascription*[1] has an attribute and an attribuent as arguments. The attribute must be a *quality* or an *object* and the attribuent must be an *object*. Clearly, there are differences in different languages, but closely related languages may share significant parts in their upper models.

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 textual-object[2] is realized as a noun phrase, and its descendents unidentifiable-object and identifiable-object are realized as indefinite or definite noun phrases, respectively. The hierarchies of textual semantic categories, as the upper model, are language sensitive, albeit different languages may share common parts.

Our text structure is mainly adopted from Meteer, together with the splitting of semantic categories proposed by Panaget. A domain concept is inserted to the upper model for every concept used in the PCAs, primarily including the rhetorical relations, predicates, and functions symbols in predicate logic.

Concretely now, text structure in our application represents the following information:

- constituency,
- structural relations among constituents,
- upper model concept the constituent expresses,
- textual semantic category, and
- layout decisions for the constituent.

While Meteer treats a single node in a text structure as an atomic element, we explicitly represent compound constituent structures of upper model objects in order to support aggregation operations to be carried out before the corresponding upper model objects are expanded. Therefore, the content of our text structure nodes can be one of the following: an upper model object, an application object (such as PCAs and formulas), or a compound upper model object with application objects as leaves (compare section 5).

[1] Upper model concepts are noted in *slanted* text.

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

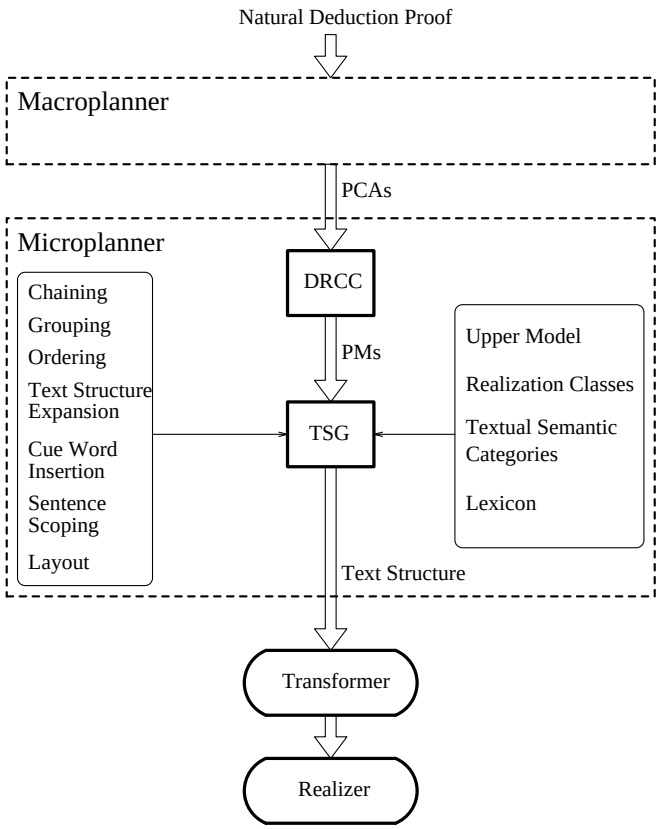


Figure 2: The architecture of the microplanner.

The output of our microplanner is a text structure with all decisions already made. That is the textual semantic category of every node is chosen, and a lexical item is selected for every node dominating a single word constituent. Such text structures are linguistic specifications that can be handed over to the realization component.

4 The Architecture of the Microplanner

Intuitively speaking, while the macroplanner determines what to say, the microplanner elaborates on how to say it. The task of the latter includes aggregation to remove redundancies, insertion of cue words to increase coherence, making reference choices, as well as lexical choices. Apart from that, the microplanner handles also sentence scoping and layout. An overview of the microplanner's architecture is provided in Figure 2.

Our microplanner takes as input an ordered sequence of PCAs. The first module, the *derivation reference choice component* (DRCC), suggests which parts of a PCA are to be verbalized. PCAs with these decisions annotated are called *preverbal messages* (PMs).

Starting from a list of preverbal messages as the initial text structure, the microplanner progressively maps concepts in preverbal messages into text structure objects of some textual semantic type via upper model objects as an intermediate level. The text structure evolves by expanding leaves top-down and from left to right. The main module of our microplanner is the

text structure generator (TSG), which builds up the text structure by applying appropriate rules:

- First a list of preverbal messages is always mapped to a list of the corresponding domain concepts in a one-to-one manner. Note that the main work is done in the placement of a domain concept in the upper model, which already defines the concept along the ideational dimension according to the systemic linguistic theory. This justifies such a simple one-to-one mapping.
- whenever applicable, ordering and aggregation rules are applied to arguments of a list (an upper model concept itself), in order to produce more concise and more coherent text. The application of an aggregating rule before the expansion of a leaf node may trigger the insertion of cue words.
- A text structure node containing a compound domain concept is then expanded to a text structure of appropriate textual semantic type via so-called *realization classes*. Each realization class represents the multiple realization possibilities in form of so-called *resource trees*.
- A fully expanded text structure will be traversed again in order to:
  - choose a lexical item for every upper model concept from the lexicon.
  - to make sentence scoping decisions by singling out one candidate textual semantic category for each constituent. This in turn may trigger the execution of a cue word rule. For instance, the choice of the category *sentence* for a constituent may lead to the insertion of the cue word "furthermore" in the next sentence.
  - to determine the layout parameters, which will be realized later as LaTeX-commands in the final output text.

A text structure constructed in this way is the output of our microplanner, and will be transformed into the input formalism of TAG-GEN [KF95], our linguistic realizer. TAG-GEN is designed to handle both German and English.

While we have kept the architecture and its operations general enough to support multilingual generation, only the upper model, the realization classes, and the lexicon are language sensitive. Since English and German are fairly similar and since our domain is strictly limited, we are experimenting with a uniform upper model and a uniform textual semantic hierarchy. However, the realization classes and the lexicon are separated entirely for the first prototype, with the hope that they will be limited in size for our application.

5 A Walk through an Example

This section tries to elucidate the most important operations by walking through a sample execution. Let us start with a text structure with the PM

`(Derive :reason (`$a \in F,\ F \subset G$`) :conclusion` $a \in G$`)`

as the application object in a leaf node, which has `sentence` as textual semantic category, and trace the processing of this leaf. First, the PM is mapped to the text structure object below.

```
(leaf
 :content (derive :reason (a ∈F, F ⊂G) :conclusion a ∈G)
 :tsc (sentence))
```

To expand this leaf, we must choose a resource tree of the realization class *derive* below. Note that we only list the first resource tree in this class explicitly. The realization classes for English and German are identical in this case.

```
(realization-class (derive :reason R :conclusion C)
  (resource-tree (composite-tree
                  :content nil
                  :tsc (sentence clause)
                  :matrix (leaf
                           :content C
                           :tsc (clause))
                  :adjunct (composite-tree
                            :content since
                            :tsc (clause)
                            :matrix (leaf
                                     :content R
                                     :tsc (clause)))))
  ⟨further resource trees ...⟩)
```

Our heuristics recommend here the resource tree shown above. After instantiating R with $(a \in F, F \subset G)$ and C with $a \in G$, this resource tree is used to expand the leaf. Let us concentrate on the processing of the second leaf of the resulting text structure shown below.

```
(leaf
 :content (a ∈F, F ⊂G)
 :tsc (clause))
```

The list of the two formulae $(a \in F, F \subset G)$ is mapped to the following text structure object:

```
(leaf
 :content (conjunction :args ((element-of :element a :set F)
                              (subset-of :subset F :superset G)))
 :tsc (clause))
```

Since the root of the text structure object is the upper model concept *conjunction*, we try to aggregate its arguments. The system applies an aggregation rule called *object grouping*, as defined below:

Rule 1 *(object grouping)*

Let P and Q be domain model concepts subordinated to the concept *object*. Furthermore, let $S_1, \ldots, S_m$ and $T_1, \ldots, T_n$ be application objects with $S_k \in \{T_1, \ldots, T_n\}$ for a $1 \leq k \leq m$.

$$\frac{P\langle S_1, \ldots, S_{k-1}, S_k, S_{k+1}, \ldots, S_m\rangle, Q\langle T_1, \ldots, T_n\rangle}{P\langle S_1, \ldots, S_{k-1}, Q\langle T_1, \ldots, T_n\rangle, S_{k+1}, \ldots, S_m\rangle}$$

This notation means, the structures above the bar are replaced by those beneath the bar. The application objects in angles $\langle\rangle$ are the arguments of the corresponding domain model concepts. Thus, this rule means: replace S_k in $P\langle S_1, \ldots, S_{k-1}, S_k, S_{k+1}, \ldots, S_m\rangle$ by $Q\langle T_1, \ldots, T_n\rangle$.

In our example, P is the domain model concept *element-of* and Q is the concept *subset-of*. Furthermore, $k = m = n = 2$. Applying this rule we get:

```
(leaf
 :content (conjunction :args ((element-of :element a
                                           :set (subset-of :subset F
                                                           :superset G))))
 :tsc (clause))
```

The application of this rule adds further restrictions on the choice of realization classes, which are omitted here. Then the conjunction is deleted since it has only a single argument, which is taken to the outer level. The expansion of the domain model concepts *element* and *subset-of* are the remaining operations to be carried out. The final text structure is displayed in Figure 3, which the realization component verbalizes to

"Since a is an element of the subset F of G, a is an element of G."

For this simple example, the applied resource trees for English and German are identical. Thus, the final text structures for English and German differ only in their lexical items. The German realization of the text structure is

"Da a ein Element von der Teilmenge F von G ist, ist a ein Element von G."

6 Discussion and Future Development

This paper provides a snapshot of the ongoing research of a microplanning component for the generation of multilingual mathematical proofs. Our main claim is that at least for similar languages like German and English, and for a limited domain such as mathematical proofs, macroplanning can be carried out in a language-independent way. Although the text structure contains language specific elements, we claim that the operational part of the microplanner can be organized in a language independent architecture. Language differences are exclusively reflected in delarative linguistic knowledge bases, namely the upper model, the textual semantic categories, the realization classes, and the lexicon.

We are extending our microplanner to cope with Chinese, a third language with significantly distinct linguistic structures. To do this, the following tasks are on the agenda: to enrich our realizer TAG-GEN with a Chinese grammar, to refine the upper model [BMNZ91] and textual semantic categories to incorparate Chinese specific categories, and to set up extra realization classes and an extra lexicon for Chinese. As the realization classes and the lexicon grows, the redundance will soon become apparent. It will be useful to study how much of the resource trees and the lexicon can be shared in a manageable way. Finally, it is an interesting question whether the architecture suggested is also feasible for more complex domains.

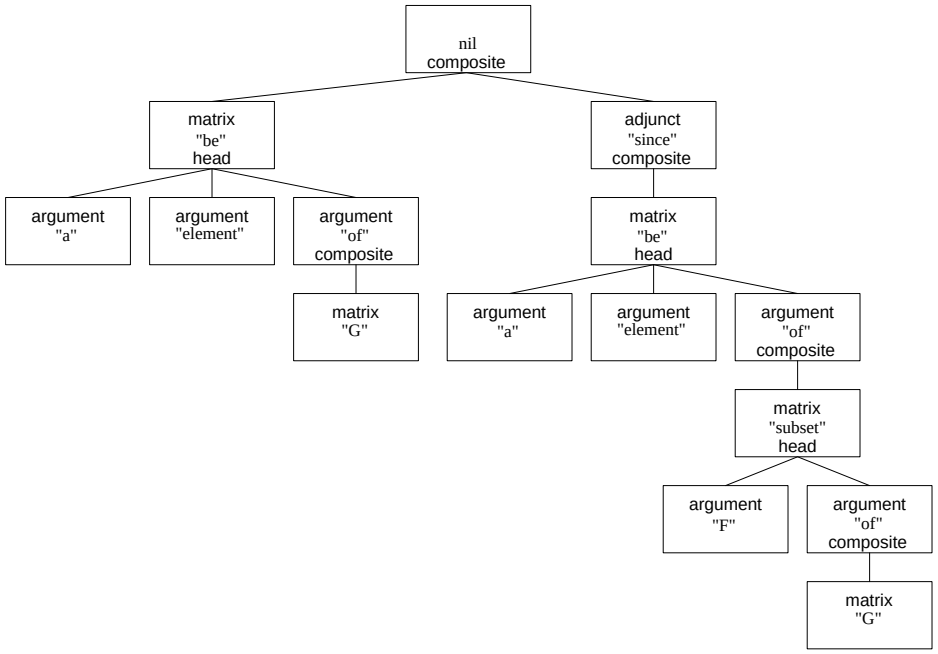


Figure 3: The text structure of the example.

References

[BKMW90] John Bateman, Bob Kasper, Johanna Moore, and Richard Whitney. The penman upper model. Technical Report ISI research report, USC/Information, Science institute, 1990.

[BMNZ91] John Bateman, Christian Matthiessen, Keizo Nanri, and Licheng Zeng. The re-use of linguistic resources across languages in multilingual generation components. In John Mylopoulos and Ray Reiter, editors, *Proc. of IJCAI-91*, pages 966–971, Sydney, 1991. Morgan Kaufmann.

[FH95] Armin Fiedler and Xiaorong Huang. Aggregation for mathematical proofs. In *Proc. of 5th European Workshop on Natural Language Generation*, 1995. forthcoming.

[HLM+92] E. H. Hovy, E. H. Lavid, E. Maier, V. Mittal, and C. L. Paris. Employing knowledge resources in a new text planner architecture. In Robert Dale, Eduard H. Hovy, Dietmar Rösner, and Oliviero Stock, editors, *Aspects of Automated Natural Language Generation*, LNAI 587, pages 57–72. Springer, 1992.

[Hov92] Eduard H. Hovy. Sentence planning requirements for automated explanation generation. Diamod 23, 1992.

[Hua94a] Xiaorong Huang. Planning argumentative texts. In *Proc. of 15th International Conference on Computational Linguistics*, pages 329–333, Kyoto, Japan, 1994.

[Hua94b] Xiaorong Huang. *PROVERB*: A system explaining machine-found proofs. In Ashwin Ram and Kurt Eiselt, editors, *Proc. of 16th Annual Conference of the Cognitive Science Society*, pages 427–432, Atlanta, USA, 1994. Lawrence Erlbaum Associates.

[IKKP90] L. Iordanskaja, M. Kim, R. Kittredge, and A. Polguère. Generation of extended bilingual statistical reports. In Hans Karlgren, editor, *Proc. of 13th International Conference on Computational Linguistics*, pages 329–333, Helsinki, 1990.

[IKP91] Lidija Iordanskaja, Richard Kittredge, and Alain Polguère. Lexical selection and paraphrase in a meaning-text generation model. In Cécile L. Paris, William R. Swartout, and William C. Mann, editors, *Natural Language Generation in Artificial Intelligence and Computational Linguistics*, pages 293–312. Kluwer Academic Publishers, 1991.

[KF95] Anne Kilger and Wolfgang Finkler. TAG-based incremental generation. *Computational Linguistics*, 1995. forthcoming.

[Met91] Marie W. Meteer. Bridging the generation gap between text planning linguistic realization. *Computational Intelligence*, 7(4), 1991.

[Met92] Marie W. Meteer. *Expressibility and the Problem of Efficient Text Planning*. Pinter Publishes, London, 1992.

[Pan94] Franck Panaget. Using a textual representational level component in the context of discourse or dialogue generation. In *Proc. of 7th International Workshop on Natural Language Generation*, pages 127–136, Kennebunkport, Maine, USA, 1994.

[Zuk91] Ingrid Zukerman. Using meta-comments to generate fluent text in a technical domain. *Computational Intelligence*, 7:276–295, 1991.