The Textplanning Component PIT of the LILOG System

J. Kreyß and H.-J. Novak
IBM Deutschland GmbH, Wissenschaftliches Zentrum
Institut für Wissensbasierte Systeme
Postfach 80 08 80, 7000 Stuttgart 80, West-Germany
Phone: (+49) 711-6695-447/540, e-mail: NOVAK at DS0LILOG.BITNET

Abstract

In this paper we describe the construction and implementation of PIT (Presenting Information by Textplanning), a subsystem of the LILOG textunderstanding system. PIT is used for planning answers of paragraph length to questions of the kind *What do you know about X?* We concentrated on a simple, easy to implement mechanism that can be further extended. Experiences with this planning component, especially concerning the integration of new plans and further extensions are discussed [1].

1. Introduction

As PIT is wholly integrated into the LILOG system, first some general remarks about LILOG.

In the LILOG project (Linguistic and logic methods for the automatic understanding of German) we aim primarily at constructing a text understanding system. For the analysis part we use an HPSG-based (*Pollard and Sag 87*) syntax and semantics that is further developed for German. For the representation of world knowledge and the knowledge extracted from the texts we have devised the representation language L_{LILOG} (*Pletat and von Luck 89*). L_{LILOG} is an order sorted first-order predicate logic that allows to define and further describe sorts by using a KL-ONE like sort description language, i.e. sorts can be described by supersort and subsort relations as well as by roles (relations) and features (functions). The sorts themselves can be either primitive (atoms) or complex e.g. defined as union, intersection, or complement of other sorts with constraints on roles and features. The sorts form the conceptual entities of the system (they build an ontology) and they are organized as a lattice. The semantics of a word in the lexicon is given by a pointer into this sort lattice.

In order to find out what the text-understanding system has really understood we can ask questions about the texts. In the first prototype (*Bollinger et al. 89*) we could only ask yes/no and constituent questions. In the present scenario the system is to understand and combine the information of several paragraph length texts about places of interest in the city of Düsseldorf and we also want to be able to ask questions of the kind *What do you know about the Lambertus cathedral?* Questions of this type necessitate a textplanning component that decides first, which entities and second, in which order they should be verbalized such that a coherent descriptive paragraph is generated.

There have been several approaches to the generation of coherent texts that can be coarsely divided into two kinds: the schema based approach and the plan based approach.

The first is described in detail in *McKeown 85*. Schemata are representational structures for stereotypical paragraphs that describe objects. A variant of this approach, somewhere between the schema and the plan based approach is described in *Novak 86* and *Novak 87*. Here the structure of the whole text is based on a schema whereas the single paragraphs are constructed using domain restrictions and a technique called anticipated visualization. The aim is to describe the movement of an object such that the hearer can visualize it as it has been seen by the system.

The plan based approach has been put forth, among others, by *Meehan 76*, *Cohen 78*, *Appelt 85*, and *Hovy 85*. *Mann and Thompson 88* propose a set of about 20 relations sufficient to represent the relations that hold within texts occurring in English. These relations, called RST (Rhetorical Structure Theory), have been operationalized and used as plans (*Hovy 88*) in a top-down hierarchical expansion planner. The planner takes as input one or more communicative goals along with a set of clause-sized inputs to be generated as an English paragraph. It assembles the input entities into a tree that embodies the paragraph structure. Nonterminal nodes in the tree are RST relations and terminal elements contain the inputs.

In our approach the same kind of planner as described in *Hovy 88* is used to *find* the entities in the knowledge base that should be generated.

In the following we first describe the overall architecture of the planner and then its implementation.

[1] Both authors are indebted to Eduard Hovy who devised the planning component while staying as a guest scientist in the LILOG project. The refinements, the implementation, and the experiences reported here have been made by the authors. The views expressed in this paper are our sole responsibility.

2. Architecture

Our textplanner basically decides **what** to say and gives as output a linear list of the conceptual entities that should be verbalized as answer to general questions of the kind *What do you know about X?* or *What can you tell me about X?*

The planner takes as input a communicative goal, e.g. *describe(x)*, and needs access to all knowledge sources of the system, namely to the user model, the ontology, the background knowledge and the textknowledge. As the knowledge of the system is represented in L_{LILOG} we use the inference engine for lookup and inferences. The user model currently only contains the facts that are already known to the hearer. The ontology is given by the sort hierarchy of the system, the background knowledge contains world knowledge in the form of facts and inference rules and finally the textknowledge results from the analysis of seven short paragraphs describing places of interest in the city of Düsseldorf.

The output is a list of the entities and their attributes that should to be verbalized in this order. This list is passed on to the generator that determines sentence boundaries and decides on the syntactic realization of the entities. The result of the generator is a formal description of the output sentence. This description is then taken by the formulator that constructs a correctly inflected German sentence. The formulator is a system similar to SUTRA (*Busemann 88*) or MUMBLE-86 (*McDonald and Meteer 88*).

2.1 Implementation

In general, our implementation of the planner is along the same lines as described in *Hovy 88* except that we incorporate not only RST relations but also domainspecific relations like ACCESS (how does one get to an object) and OPEN (when are the opening hours of an object). *Moore and Swartout 89* and *Moore and Paris 89* use the same planning algorithm and they have added plans like e.g. PERSUADE to the RST plans. This enables them to answer follow-up questions in advisory dialogues or in the explanation facility of an expert system. Most of the questions they can answer are *Why* questions except two *What* questions: *What is a* concept? and *What is the difference between two* concepts? The general idea of their approach too is to gather the information that should be communicated but using their plans we could not answer the kind of general question we have in mind.

Like RST plans our plans consist of a nucleus and a satellite each associated with requirements and growth points. The nuclei contain the information that has to be verbalized obligatorily which is either done by recursively invoking other subplans or by an explicit verbalization command *say(x)*. All plans are recursively expanded until they lead to a verbalization command. In contrast to nuclei satellites, using the same notation, contain the same kind of information that can be optionally verbalized. The growth points allow for the inclusion of further information into the list of entities that is finally passed on to the generator. They again contain plans. Finally, the requirements for nucleus and satellite contain inquiries to the inference engine about e.g. the validity of certain subsort relations and about beliefs of the hearer. An example of a plan, *interesting_feature*, is given below (the planner is implemented in PROLOG so the atoms with capital letters are variables):

```
plan(interesting_feature(Object),
     nucleus:  [say(Object)],
     satellite:  [say(Feature)],
     nucleus_requirement:
       and([subsort(Object,object)]),
     satellite_requirement:  [],
     nucleus_and_satellite_requirement:
       and([
         attribute(Object, remarkability:  Feature),
         not(bel(hearer, attribute(Object,
         remarkability:  Feature)))]),
     nucleus_growth_point:
       [interesting_feature(Feature)],
     satellite_growth_point:  [])
```

Among the 12 plans used by PIT are domain dependent ones as well as domain independent ones. The latter are formalizations of RST relations that lead to small text structures. The domain dependent plans lead to larger structures, e.g. whole paragraphs. Each plan, even if it can be seen as domain independent, contains a domain specific part, namely the requirements for nucleus and satellite which are inquiries to the inference engine that have to heed the names of entities, roles, and features of the knowledge representation.

The planning algorithm uses four data structures: the plans, a tree, a stack, and a used list. The text structure tree is binary. The root contains the communicative goal that initiates the planning process. The nodes represent the executed plans. Each node has successive nucleus and satellite edges whose corresponding nodes may be either empty or contain an explicit verbalization command or further plans.

The stack is used as an agenda. Its elements are tuples consisting of the plan to be executed next and a pointer to that leaf of the tree where the subtree stemming from the execution of the plan should be added.

The used list is a bookkeeping device representing which plan has been used for which entity.

The planning algorithm consists of three phases: first, the text structure tree is built by a top-down hierarchical planner (*Sacerdoti 75*) using recursive descent. Second, the verbalization commands are collected by traversing the tree depth-first, left-to-right. Third, the entities to be verbalized are expanded by their attributes contained in the knowledge base and

are passed on to the generator in a suitable form.

At the start of the planning process, i.e. when the communicative goal comes in, the tree, the stack, and the used-list are empty. If the plan library offers an appropriate plan to achieve the goal it is tested whether this plan has already been executed for the entity in question. If so, the execution is aborted, otherwise the plan is put on the used list.

Next the requirements of the plan are checked, first, the ones common to both nucleus and satellite and then the nucleus requirements. If they cannot be met, execution of the plan aborted, otherwise the requirements of the satellite are checked. If they cannot be met the corresponding plans of the satellite and the satellite growth points are skipped. Are all requirements met, the new plans together with their pointers to that leaf of the tree where the subtrees should be added are pushed onto the stack in the following order: satellite growth points, nucleus growth points, satellite and nucleus.

The second plan that is to be executed is popped from the stack and dealt with as described above with the addition that the agenda has to be updated when the tree has been expanded. The pointers of all plans to that leaf of the tree where a subtree has been added have to be changed in order to point to the nucleus of the new subtree.

Planning stops when the agenda is empty.

3. Shortcomings and possible extensions

The original plans like the one shown above are based on an extensive analysis of seven paragraphs describing places of interest in Düsseldorf. Hence, they capture the typical structure of such descriptions and act as more flexible schemas that can be adapted to a user's needs by incorporating more communicative goals. Nevertheless, problems arise when new plans are added or when old ones are changed. It proved to be difficult to say in advance which text structure will be the outcome of the planning process. Through the top-down expansion of the text structure tree new plans may be inserted into the tree at places where they do not have the desired effect on the text structure. E.g. the plan *features(X)* may be the nucleus of the initiating plan *description(X)* and also satellite of a more fine grained plan. As those plans that have been pushed last onto the stack are executed first and no plan is executed twice the features may be verbalized at the wrong place in the text.

Generally speaking, these problems point to the need to strictly separate the planning of the propositional and the rhetorical. Although our hierarchical planner can be used successfully to plan the content of the descriptive paragraphs we feel that a nonlinear planning algorithm might be better suited for the planning of the propositional content followed by a hierarchical planner for the rhetorical structure. Another problem is the domain dependence of the propositional planner which always sneeks in through the requirements placed on nucleus and satellite. The requirements are stated in terms of the knowledge representation language. The only partial solution to this problem is to use general terms in the planner and a separate mapping of these general terms onto the knowledge representation language. Our further research is directed in this direction.

References

Appelt 85 Appelt, D.E.: *Planning English Sentences.* Cambridge University Press, 1985.

Bollinger et al. 89 Bollinger, T., Hedtstück, U., Rollinger, C.-R.: *Reasoning for Text Understanding - Knowledge Processing in the 1st LILOG-Prototype.* In: Metzing, D. (Ed.): GWAI-89, 13th German Workshop on Artificial Intelligence. Springer, 1989, 203-212.

Busemann 88 Busemann, S.: *Surface Transformations during the Generation of Written German Sentences.* In: Bolc, L. (Hg.), Natural Language Generation Systems. Springer, Berlin, 1988, 98-165.

Cohen 78 Cohen, P.: *On Knowing What to Say: Planning Speech Acts.* Techn. Report No. 118, University of Toronto, 1978.

Hovy 85 Hovy, E.H.: *Integrating Text Planning and Production in Generation.* IJCAI-85, 848-851.

Hovy 88 Hovy, E.H.: *Planning Coherent Multisentential Text.* Proc. of the 26th Annual Meeting of the ACL, 1988, 163-169.

Mann and Thompson 88 Mann, W.C., Thompson, S.A.: *Rhetorical Structure Theory: Toward a functional theory of text organization.* In: Text 8 (3), 1988, 243-281.

McDonald and Meteer 88 McDonald, D.D., Meteer, M.W.: *From Water to Wine: Generating natural Language Text from Today's Application Programs.* Proc. of the 2nd ACL Conference on Applied Natural Language Processing, 1988, 41-48.

McKeown 85 McKeown, K.R.: *Text Generation: Using Discours Strategies and Focus Constraints to Generate Natural Language Text.* Cambridge, Cambridge University Press, 1985.

Meehan 76 Meehan, F.: *The Metanovel: Writing Stories by Computer.* Ph.D. dissertation, Yale University, 1976.

Moore and Paris 89 Moore, J.D., Paris, C.L.: *Planning Text for Advisory Dialogues.* ACL-89, 203-211.

Moore and Swartout 89 Moore, J.D., Swartout, W.R.: *A Reactive Approach to Explanation.* IJCAI-89, Detroit.

Novak 86 Novak, H.-J.: *Generating a Coherent Text Describing a Traffic Scene.* COLING-86, 570-575.

Novak 87 Novak, H.-J.: *Textgenerierung aus visuellen Daten: Beschreibungen von Strassenszenen.* Berlin/Heidelberg/New York, Springer, 1987.

Pletat and von Luck 89 Pletat, U., von Luck, K.: *Knowledge Representation in LILOG.* In: Bläsius, K.-H., Hedtstück, U., Rollinger, C.-R. (eds.): Sorts and Types in Artificial Intelligence. Springer, 1989, (to appear).

Pollard and Sag 87
Pollard, C., Sag, I.A.: *An Information-Based Syntax and Semantics. Vol. 1, Fundamentals.* CSLI Lecture Notes 13, Stanford, 1987.

Sacerdoti 75 Sacerdoti, E.D.: *A structure for plans and behaviour.* North-Holland Publishing Company, Amsterdam, 1975.