In: Maybury, M.: Intelligent Multimedia Interfaces, AAAI Press, Menlo Park, pp. 75-93.

WIP: The Automatic Synthesis of Multimodal Presentations

Elisabeth André, Wolfgang Finkler, Winfried Graf, Thomas Rist, Anne Schauder, and Wolfgang Wahlster

German Research Center for Artificial Intelligence (DFKI)
Stuhlsatzenhausweg 3, W-6600 Saarbrücken 11, Germany
Phone: (+49 681) 302-5252
Fax: (+49 681) 302-5341
Email: {andre, finkler, graf, rist, schauder, wahlster}@dfki.uni-sb.de

Abstract

Due to the growing complexity of information that has to be communicated by current AI systems, there comes an increasing need for building advanced intelligent user interfaces that take advantage of a coordinated combination of different modalities, e.g., natural language, graphics, and animation, to produce situated and user-adaptive presentations. A deeper understanding of the basic principles underlying multimodal communication requires theoretical work on computational models as well as practical work on concrete systems. In this article, we describe the system WIP, an implemented prototype of a knowledge-based presentation system that generates illustrated texts that are customized for the intended audience and situation. We present the architecture of WIP and introduce as its major components the presentation planner, the layout manager, and the generators for text and graphics. To achieve a coherent output with an optimal media mix, the single components have to be interleaved. The interplay of the presentation planner, the text and the graphics generator will be demonstrated by means of a system run. In particular, we show how a text-picture combination containing a crossmodal referring expression is generated by the system.

Contents

1 Introduction

With increases in the amount and sophistication of information that must be communicated to the users of complex technical systems comes a corresponding need to find new ways to present that information flexibly and efficiently. Intelligent presentation systems are important building blocks for the next generation of user interfaces, because they translate from the narrow output channels provided by most of the current application systems into high-bandwidth communications tailored to the individual user. Since, in many situations, information is only presented efficiently through a particular combination of communication modes[1], the automatic generation of multi-modal presentations is one of the tasks of such presentation systems. Multimodal interfaces combining, e.g., natural language and graphics take advantage of both the individual strength of each communication mode and the fact that several modes can be employed in parallel, e.g., in the text-picture combinations of illustrated documents (see also [Sullivan & Tyler 91,Ortony et al. 92,Roth & Heffley 92]).

It is an important goal of this research not simply to merge the verbalization results of a natural language generator and the visualization results of a knowledge-based graphics generator, but to carefully coordinate graphics and text in such a way that they complement each other (see also [Wahlster et al. 91,Wahlster et al. 92a]).

The automatic design of multimodal presentations has only recently received significant attention in artificial intelligence research. Most systems generate written text and graphics including bar charts (see the system *SAGE* [Roth et al. 91]), network diagrams (cf. [Marks *h* Reiter 90]), weather maps (cf. [Kerpedjiev 92]) and depictions of 3D objects (see the systems *COMET* [Feiner & McKeown 92] and *WIP* [Wahlster et al. 92a, Wahlster et al. 92b]). [Maybury 92] is concerned with the planning of multimodal directions in a cartographic information system. Badler and colleagues (cf. [Badler et al. 91b]) focus on the generation of animation from instructions. Further work concentrates on the analysis and representation of relevant design knowledge (cf. [Arens et al. 92]) as an important prerequisite for the automatic design of multimodal presentations.

The work closest to our own is done in the COMET project (cf. [Feiner & McKeown 92]). Both projects share a strong research interest in the coordination of text and graphics. COMET generates directions for the maintenance and repair of a portable radio using text coordinated with 3D graphics. In spite of many similarities, there are major differences between COMET and WIP, e.g. in the systems' architectures, representation languages and processing strategies. While during one of the final processing steps of COMET the media layout component is supposed to combine text and graphics fragments produced by media-specific generators, in WIP layout considerations can influence the early stages of the planning process and constrain the media-specific generators. In WIP, we view layout as an important carrier of meaning. COMET uses a schema-based content planner while WIP uses an operator-based approach to planning. Other distinguishing features of WIP's architecture are that it supports incremental output and that mode selection is done not after, but during content planning.

[1] Since one of the generation parameters of WIP is the specification of the output device, we use the term 'medium' in the sense of a physical carrier of information. In contrast, the term 'mode' is used throughout this paper to refer to the particular sign system. We are aware of the fact that other authors use these terms differently.

2 A Functional View on WIP

The task of the knowledge-based presentation system WIP is the generation of a variety of multimodal documents from an input consisting of a formal description of the communicative intent of a planned presentation. The generation process is controlled by a set of generation parameters such as target group, presentation objective, resource limitations, and target language.

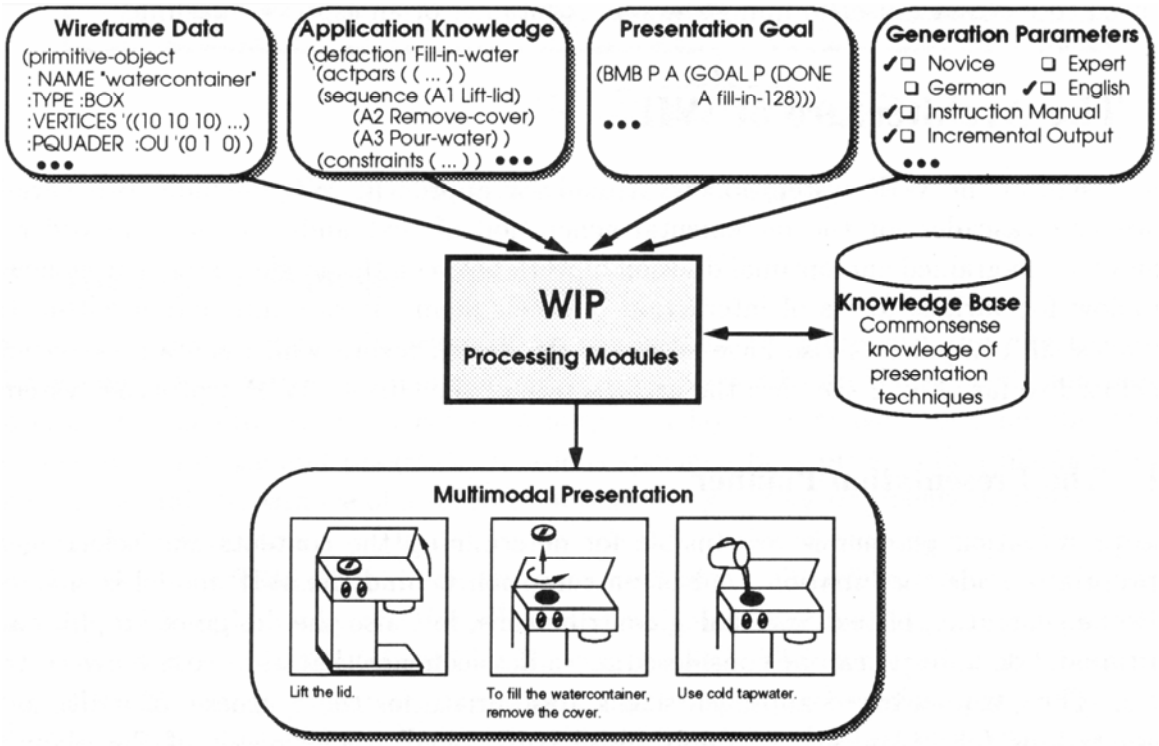


Figure 1: A Functional View on WIP

The example of a presentation goal in Fig. 1 represents the system's assumption about the mutual belief (BMB) of the presenter P and the addressee A that it is P's goal that A carries out a plan denoted by the constant fill-in-128. This is a concrete domain plan specified as part of WIP's application knowledge. In this case, the plan is a fully instantiated sequence of actions represented in the assertional part of the hybrid knowledge representation system RAT (Representation of Actions in Terminological Logics, cf. [Heinsohn et al. 92]). The terminological part of RAT is used to represent the ontology and abstract plans for a particular application domain (see Fig. 1).

In addition to this propositional representation, which includes the relevant information about the structure, function, behavior, and use of the technical device, WIP has access to an analogical representation of the geometry of the machine in the form of a wireframe model (see Fig. 1).

WIP is a transportable interface based on processing schemes that are independent of any particular back-end system and thus requires only a limited effort to adapt to a new application. Obviously, for a new domain the application knowledge and the wireframe model must be transformed into WIP's representation schemes. Currently all input for the development and testing of the system is, however, created manually.

A good example of the use of a system like WIP is the generation of user-friendly multimodal instructions for technical devices. As a first domain, we have chosen instructions for the use of espresso-machines. Fig. 1 shows a typical text-picture sequence that may be used to instruct a user in filling the watercontainer of an espresso-machine.

3 The Architecture of WIP

The design of the WIP system follows a modular approach. WIP includes two parallel processing cascades for the incremental generation of text and graphics. In order to achieve a fine-grained and optimal division of work between the single system components, we allow for various forms of interaction between them. Beside interaction within the cascades, all components also have access to the design record which contains all results generated so far. Fig. 2 sketches the architecture of the current WIP prototype system.

3.1 The Presentation Planner

The presentation planner is responsible for determining the contents and selecting an appropriate mode combination. A basic assumption behind the WIP model is that not only the generation of text and dialog contributions, but also the design of graphics and multimodal documents can be considered as an act sequence that aims to achieve certain goals. Thus, a plan-based approach seems appropriate for the synthesis of multimodal presentations (cf. [André & Rist 90b, André & Rist 90a]). The result of the planning process is a hierarchically structured plan of the document to be generated in the form of a directed acyclic graph (DAG). This plan reflects the propositional contents of the potential document parts, the intentional goals behind the parts, as well as rhetorical relationships between them. While the top of the presentation plan is a more or less complex presentation goal (e.g., explaining how to make coffee), the lowest level is formed by specifications of elementary presentation tasks (e.g., formulating a request or depicting an object) that are passed directly to the mode-specific design components. A detailed description of the presentation planner is given in [André & Rist 92].

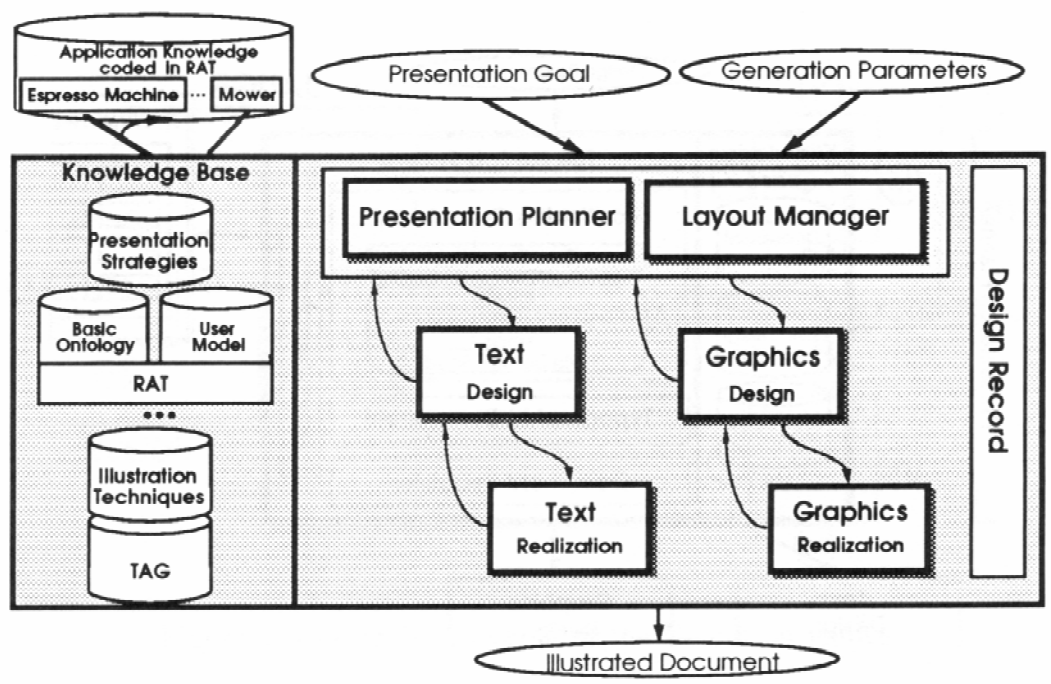


Figure 2: The Architecture of the WIP System

3.2 The Layout Manager

To communicate generated information to the user in an expressive and effective way, a knowledge-based layout component has to be integrated into the presentation design process. In order to achieve a coherent output, a layout manager must be able to reflect certain semantic and pragmatic relations specified by a presentation planner to arrange the visual appearance of a mixture of text and graphics fragments delivered by the mode-specific generators, i.e., to determine the size of the layout objects and the exact coordinates for positioning them on the document page. Therefore, we use a grid-based approach as an ordering system for efficiently designing functional (i.e., uniform, coherent, and consistent) layouts (cf. [Müller-Brockmann 81]). This concept has also been used in the GRIDS system for automatically laying out displays containing text and pictures (cf. [Feiner 88]) and by Beach for low-level table layout (cf. [Beach 85]). Fig. 3 sketches the architecture of WIP's layout manager including a constraint-based positioning component (CLAY), an intelligent typographer, a document rendering component, and an interaction handler. For the rest of this paragraph we will only talk about the positioning component (see also [Graf & Maaß 91, Maaß 92]).

The automatic placement of layout objects in a design space can be viewed as a combinatorial problem. Therefore, we treat layout as a constraint satisfaction problem *(CSP)* in a finite discrete search space (cf. [Mackworth 77]). We encode graphical design knowledge via constraints expressing semantic/pragmatic and geometrical/topological relations. Semantic and pragmatic constraints essentially correspond to coherence relations, such as the rhetorical relations 'sequence' and 'contrast' specified in the RST theory by

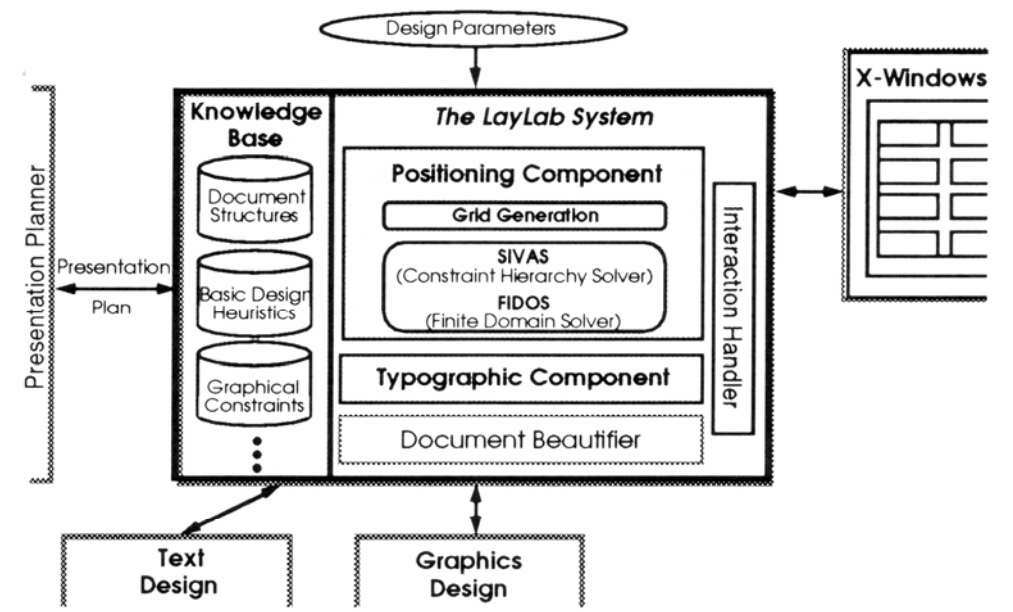


Figure 3: The Various Modules of the Layout Manager

[Mann & Thompson 88], and can be easily reflected through specific design constraints. They describe perceptual criteria concerning the organization of the visual elements, such as the sequential ordering (horizontal vs. vertical layout), alignment, grouping, symmetry, or similarity. Geometrical resp. topological constraints refer to absolute and relative constraints. Absolute constraints fix geometric parameters to constant values (e.g., coordinates). Relative constraints relate a geometric parameter of one object to another.

To give an example of a typical compound constraint in a predicate logic like notation, let's have a look at the representation of the 'contrast'-constraint (cf. Fig. 4) and the illustration through the corresponding constraint network in Fig. 5.

$$
\begin{aligned}
&\mathbf{CONTRAST}\ (G_1, G_2) \leftrightarrow \\
&\qquad G_1 \equiv \mathrm{pkt}(x_{G_1}, y_{G_1}, wi_{G_1}, he_{G_1}) \wedge \\
&\qquad G_2 \equiv \mathrm{pkt}(x_{G_2}, y_{G_2}, wi_{G_2}, he_{G_2}) \wedge \\
&\qquad [\mathbf{EQUAL}(y_{G_1}, y_{G_2}) \wedge \mathbf{BESIDE}(x_{G_1}, wi_{G_1}, x_{G_2}) \\
&\qquad \vee \\
&\qquad \mathbf{EQUAL}(x_{G_1}, x_{G_2}) \wedge \mathbf{UNDER}(y_{G_1}, he_{G_1}, y_{G_2})]
\end{aligned}
$$

Figure 4: Representation of the Compound-Constraint *CONTRAST*

When using constraints to represent layout knowledge, one often wants to prioritize the constraints in those which must be required and others which are preferably held (cf. [Borning et al. 87]). A powerful way of expressing this layout feature is to organize the constraints in a hierarchy by assigning a preference

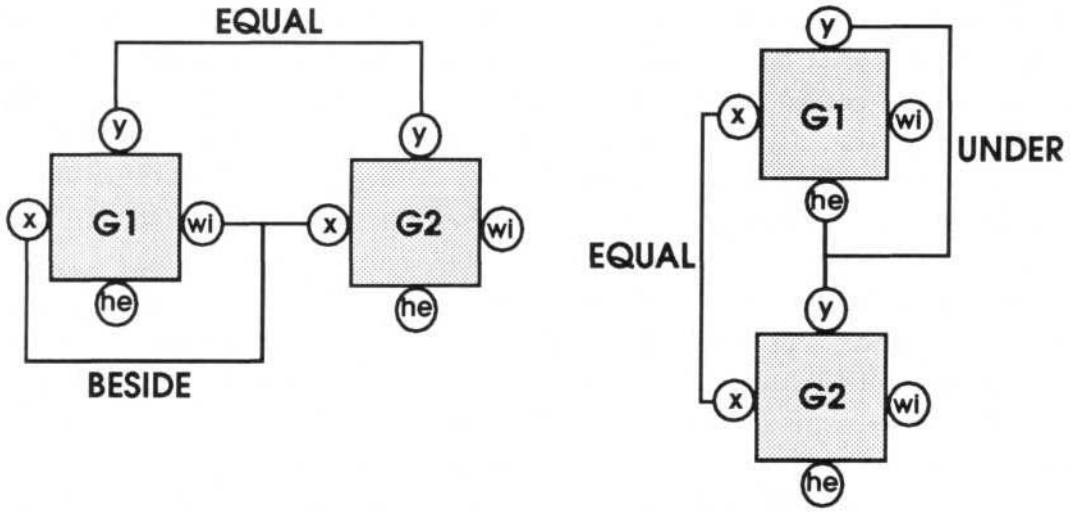


Figure5: Constraint Network of the Definition Above

scale to the constraint network. We distinguish between *obligatory*, *optional* and *default constraints*. The latter state default values, which remain fixed unless the corresponding constraint is removed by a stronger one. As graphical constraints frequently have only local effects, they are generated by the system on the fly.

For constraint solving, two dedicated incremental solvers are organized hierarchically in CLAY. An incremental constraint hierarchy solver based on the DeltaBlue algorithm by Freeman-Benson (cf. [Freeman-Benson et al. 90]) and a domain solver that handles finite domains like the approach in CHIP (cf. [Hentenryck 89]) are integrated in a layered model. These solvers are triggered from a common met a level by rules and defaults. For a more detailed description of the layout manager see [Graf 92].

3.3 The Text Generator

WIP's text generator consists of the Text Design and the Text Realization component, which form a cascade. The Text Design component receives as input from the Presentation Planner exactly that piece of knowledge that was chosen to be presented as text. It determines in which order the given input elements shall be realized in the text. The structure of a text is worked out at several levels. This comprises, for example, the partition of a paragraph into sentences, the assignment of a perspective or the use of anaphora to obtain a coherent text. Therefore, this component is comparable with the so-called 'Micro-planner', a part of the What-to-Say component - while the Presentation Planner can be seen as 'Macro-planner' (cf. [Levelt 89]).

The resulting preverbal message is grammatically encoded, linearized and inflected in the Text Realization component (How-to-Say component). Thereby, the Generation Parameters direct the choice of syntactic structures. One difficulty is in defining the boundary between the What-to-Say and the How-to-Say component.

We decided to associate the process of lexical choice with the Text Design component. The valency information of the chosen lemmas form syntactic constraints for the Text Realization component, e.g., a transitive verb must be combined with two complements, subject and object, before the sentence is syntactically correct and complete. Since these lemmas are chosen relatively independent from each other, they can cause conflicts during verbalization in the Text Realization component. To be able to report these problems to the Text Design component we propose a model with feedback between the two modules.

Our main emphasis for the text generator is on an incremental style of processing. Both components of the cascade work incrementally and information is handed over in a piecemeal fashion. This leads to greater flexibility and efficiency, since the realization component can start working before the design component has completed its results. One of the prerequisites for incremental processing in the realization component is the use of a grammar formalism which allows the specification of entities of an adequately small size (see below).

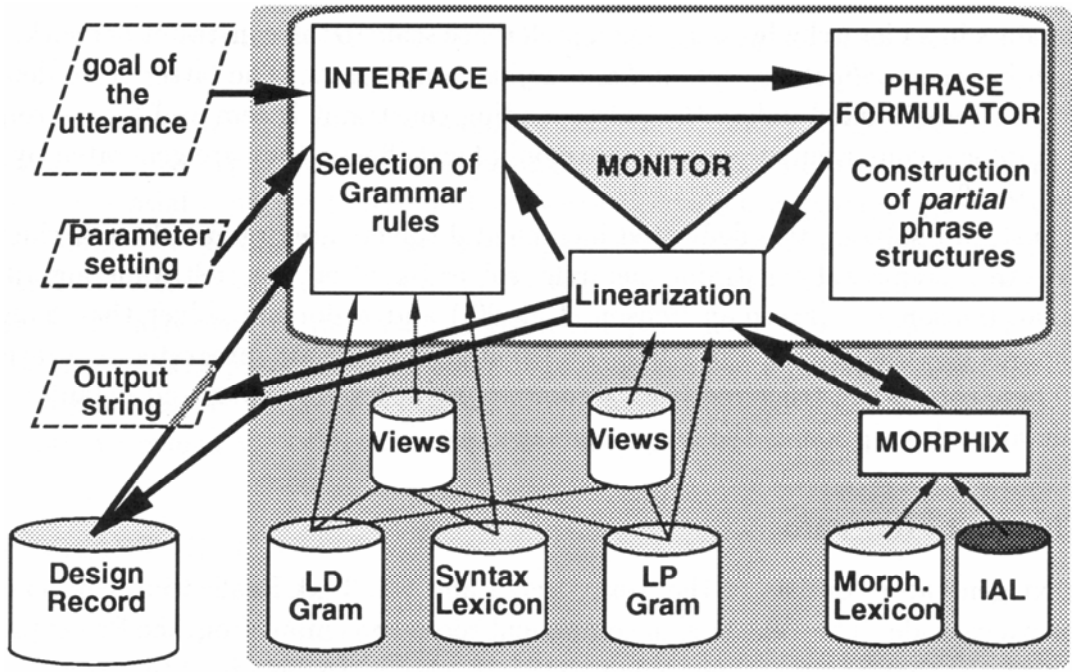


Figure 6: The Architecture of the Syntactic Generator

Figure 6 shows the architecture of the Text Realization component. The input from the Text Design component is called 'goal of the utterance'. It comprises the content words and indicates their semantic relation. On this basis the 'Interface' component chooses the respective grammar rules. The Text Realization component is based on the formalism of *Tree Adjoining Grammars* (TAGs, cf. [Harbusch et al. 91]). Reasons for the use of TAGs are, e.g., its adequate power (see [Joshi 85]) and its combination operations that allow the flexible expansion of syntactic trees (see [Schauder 92]). For each lemma of the input a grammar rule (a tree of the TAG) is chosen that represents its subcategorization

frame. In order to perfom this task, we use lexicalized Tree Adjoining Grammars. This formalism restricts the size of trees, so that they describe exactly one lemma as the head of the represented structure which is used as anchor in the lexicon.

The interface is associated with two boxes: 'LD Gram' contains the descriptions of the pure hierarchical structures (without ordering constraints), the 'Syntax Lexicon' relates lemmas with tree families of 'LD Gram'. We use LD/LP TAGs (local dominance / linear precedence TAGs) to divide the grammar rules into a set of mobiles and a set of order restricting constraints. This leads to a more compact representation, because all grammar rules representing the same hierarchical structure with different word order can be expressed within one local dominance structure (a mobile). The 'Views' on the grammar are used, if there are alternative structures. They guide the choice among these alternatives with respect to the given generation parameters (e.g., the style).

The processing inside the Text Realization component takes place within two modules: In the 'Phrase Formulator', syntactic structures are composed without consideration of word order rules. Thereby, the chosen elementary trees are combined by means of the TAG operations adjunction and substitution. The specified semantic relations are mapped to functional relations and guide these combinations. In the 'Linearization' module, the resulting trees are traversed and an adequate and syntactically correct word order is computed.

Incremental processing is supported by parallelism because the expansion of existing structures can be performed simultaneously at several branches of syntactic trees. Therefore, the basis for the Text Realization component is a distributed parallel model with active cooperating objects. For each given lemma one object is created which is responsible for the further processing of the respective syntactic structure. The dependency relations between the various lemmas define a hierarchy of objects. In the Phrase Formulator the objects try to build the complete syntactic tree by communicating with one another and exchanging information. If an object fulfills specific completeness constraints, it moves to the linearization level. There it tries to compute its local word order and - again by communication with other objects - to correctly integrate its partial terminal string into the whole sentence. After the computation of word position, the lemmas are inflected using the module *MORPHIX* (see [Finkler & Neumann 88]). Then, complete parts are uttered incrementally.

In addition to the incremental inner working, the production of incremental output is one important feature of our system. While there are several other approaches to incremental generation, most of them do not deal with incremental output. If output is delayed until it is complete, the utterances become less natural because of the long initial delay. If the output is generated incrementally, the uttered prefix forms an additional constraint for the further processing because each change becomes visible to the dialogue partner (see [Finkler & Schauder 92]).

3.4 The Graphics Generator

WIP generates illustrated instructions explaining how to operate technical devices, such as an espresso machine or a lawn mower. Thus, WIP often has to graphically communicate information about physical objects, i.e., information about object properties (such as shape, material surface characteristics, constituent structure and function), static relations to other objects (such as the location, the orientation and the distance of the objects) and dynamic relations that represent changes of object attributes and static relations (e.g., a change in the spatial position of an object).

As with text generation, we distinguish between components for graphics design and graphics realization (cf. also [Rist & André 92b, Rist & André 92a]). The realization component can be considered as an extension of a object-oriented graphics editor that handles both 2D concepts and 3D models of objects and object configurations. Thus, the component has to support three kinds of operations: First, there are operators to create and manipulate 3D object configurations. Examples are: adding an object to a configuration, spatially separating object parts to construct exploded views and cutting away object faces to make opaque parts visible. The second kind of operators constrains mapping functions and viewing specifications and performs the instantiation of images from 3D models. Finally, there are several operators defined on the picture level. E.g., an object depiction may be annotated with a label, or picture parts may be colored in order to emphasize them. Beside these achievement operators that effect either models, mappings or pictures, the functionality of the realization component also encompasses evaluation operators (e.g., to check whether an object as part of an object configuration is visible from a given viewing specification, or to check whether a picture part can be discriminated from other picture components). These evaluators are necessary in order to recognize whether the effect of an achievement operator has been destroyed by the application of subsequent achievement operators. The major modules of the realization component are: a 3D studio, a mapping controller, a 2D clipboard and a module for handling data structures for images and pictures (cf. Fig. 7).

The task of the graphics design component is to transform presentation tasks received from the presentation planner into a sequence of operators to be executed by the graphics realization component. Basic knowledge about how to accomplish this transformation is represented by so-called design strategies. Each design strategy consists of a header, an applicability condition and a body. The header may be a presentation task or a graphical constraint. The applicability condition specifies when a strategy may be used and constrains the variables to be instantiated. The body contains a set of graphical constraints that have to be achieved in order to accomplish the goal indicated in the header. Whereas some graphical constaints are directly related to achievement operators, others lead to the application of further design strategies. In addition, graphical constraints are associated with evaluation operators to check whether constraints are already satisfied and whether they are still satisfied after having executed further achievement operators

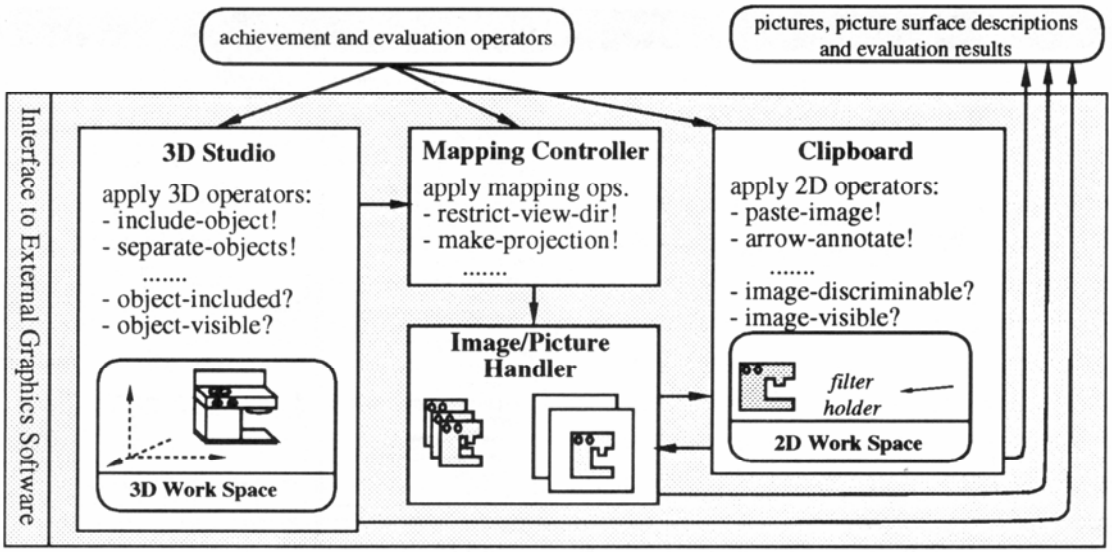


Figure 7: The Graphics Realization Component

4 Interplay of the Various Components

In the following, the interplay of the presentation planner, the text generator, and the graphics generator will be demonstrated by means of an example.

Fig. 8 and Fig. 9 show snapshots of a system run. The discourse structure built up by the planner is shown in the upper left window of the WIP frame. The results of the text and graphics generators appear in the two lower left windows. The right window displays the verbal trace messages of all system components.

In our example, the presentation planner has decided to explain to the user where the on/off switch of the espresso machine is located. To accomplish this task, one can activate a representation of the concept switch-2 that identifies switch-2 as an on/off switch, b) activate a representation of switch-2 that contains information to localize it and c) ensure that a coreferential link between the representations activated in a) and can be established. A presentation strategy for achieving this is to name switch-2 as 'on/off switch,' show the location of the switch with respect to a landmark object in a picture and relate the generated name with the corresponding picture object via graphical annotation. Since the switch is part of the espresso machine, the espresso machine is considered a suitable landmark object. After the expansion of this strategy, the graphics generator receives the task of creating a picture with the espresso machine and the switch whereas the text generation component has to find a natural language expression for the switch. In the example, the expression 'On/off switch' is generated and passed as label onto the graphics generator (see the TAG Results Window in Fig. 8). Complex nominal phrases for titles or labels are handled in exactly the same way as 'normal' sentences with a verbal predicate. The highest object in the hierarchy realizes the head of the sentence (a verb, a noun, ...) and decides on its completion.

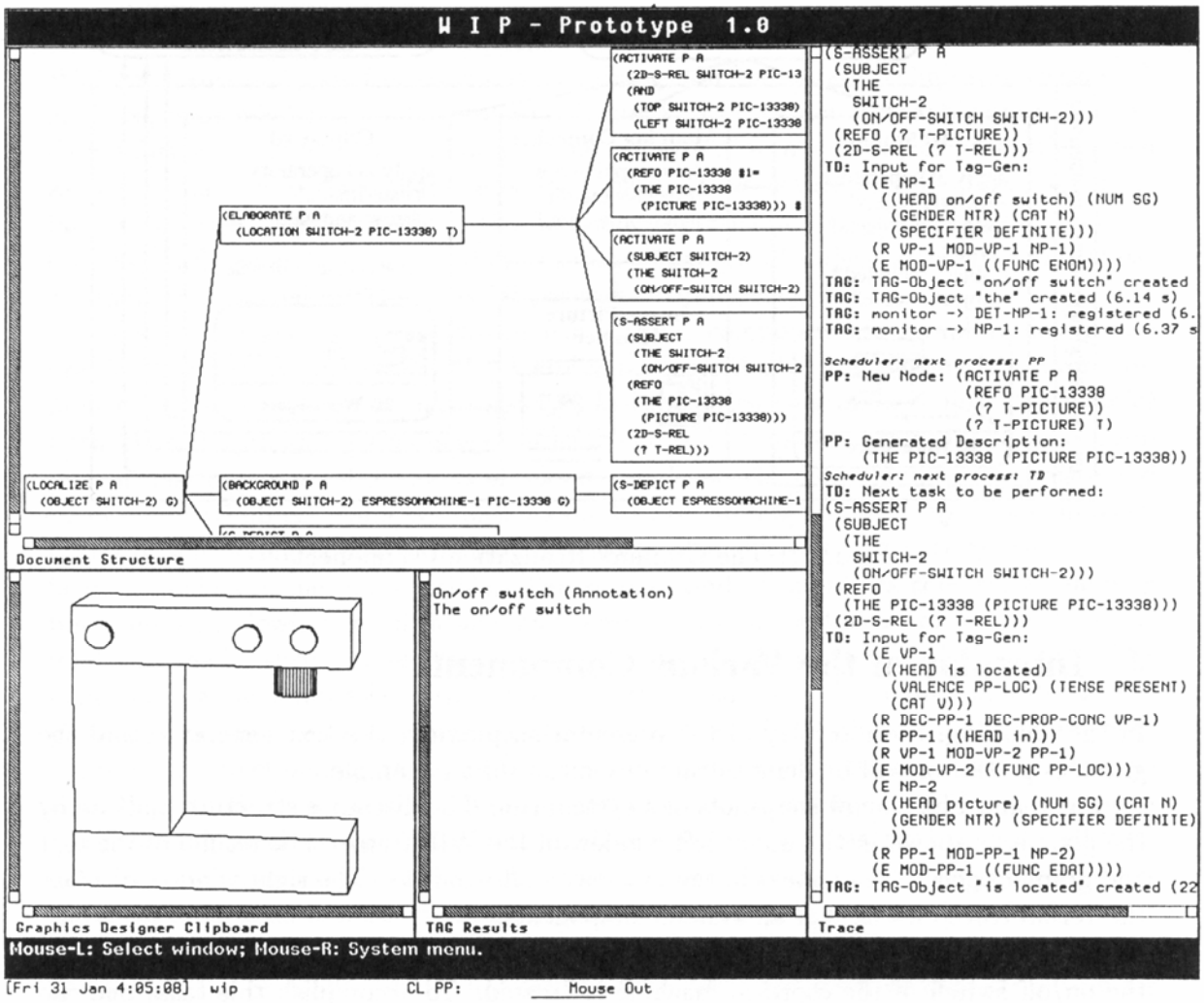


Figure 8: Snapshot 1

When trying to place this text string within the picture, it turns out that there is not enough space. It is neither possible to place the string inside the corresponding picture object nor close to it. Therefore, the graphics generator sends a message to the presentation planner that it could not accomplish the annotation task. The presentation planner then has to backtrack and try another presentation strategy. This time, it tries to achieve c) by unambiguously describing the location of the picture element corresponding to switch-2.

The top left pane in Fig. 8, labeled 'Document Structure', shows the DAG that has been produced by the presentation planner. The presentation goal (Localize P A (Object switch-2) G) has been decomposed into three subgoals: (Elaborate P A ...), (Background P A . . .) and (S-Depict P A ...). After the refinement of (Elaborate P A ...), four acts have been posted as new subgoals: three referential acts for specifying the spatial relation, the reference object and the subject, and an elementary speech act (S-Assert P A ...) which is passed onto the text designer.

As mentioned before, the planner passes a certain piece of information onto the respective generator as soon as it has decided which component should encode it. In the example, (S-Assert P A ...) is sent to the text designer although the contents of the assertion are not yet fully specified. Currently, this component is only rudimentarily realized.

The computation of the output is done by a simple transformation. The input for the text realization component consists of entities containing content words and functional relations between the entities as can be seen in the trace window of the example of WIP (see Fig. 8). These entities and relations may be specified in a piecemeal way. Their order is independent from the word order in the resulting utterance.

The resulting structures generated by the text designer at this state in our example are not sufficient for the TAG generator to start with the utterance. All it can do now is to prepare a small package of syntactic information, which will later be associated with the head of the sentence and lead to the realization of the respective style (assertion clause, …)

Some time later, the planner has determined the contents for the description of the switch. Thus, the incomplete task specification that has been sent to the text designer is supplemented accordingly. The text design component is now able to provide new input for the TAG generator. The TAG generator then starts computing and generates two objects which together form the noun phrase 'the on/off switch'. Since the verb of the sentence is not yet chosen and the relation between the noun 'on/off switch' and the verb is not yet specified, its position in the sentence cannot be computed immediately (unless we use default values as 'first noun is subject'). So the utterance is delayed until the verb is realized. Fig. 8 shows a snapshot of the system run shortly after the first part of the sentence has been uttered.

At this time, the planner has also determined the contents of a referring expression for the picture. Since there are no other pictures with which it can be confused, it is sufficient to include the concept 'picture' in the description. After the corresponding structures have been transformed by the text designer, the TAG generator has enough information to build a syntactically correct and complete sentence. Since the utterance should start as soon as possible, all words are uttered that can be added to the right of the previously uttered prefix according to the linearization rules. Since no further input information is known, the TAG generator assumes that the sentence is complete and generates: *The on/off switch is located in the picture.*

After the sentence has been completed, new information about the exact position of the switch in the picture is provided by WIP's localization component (cf. [Wazinski 92]). Since it is not possible to incorporate this information after transformations by the text designer into the already generated sentence in a syntactically correct way, the TAG generator has to revise the utterance. Up to now we have not realized sophisticated strategies for the integration of revised parts in the sentence on the output screen, so the easiest way to make revision visible is to repeat the whole sentence: *The switch is in the upper left corner of the picture* (see the TAG Results Window in Fig. 9).

Now all components have finished their task and no new goals have been posted. Thus, the layout manager is activated that is responsible for the arrangement of the text and the pictures (cf. Fig. 10).

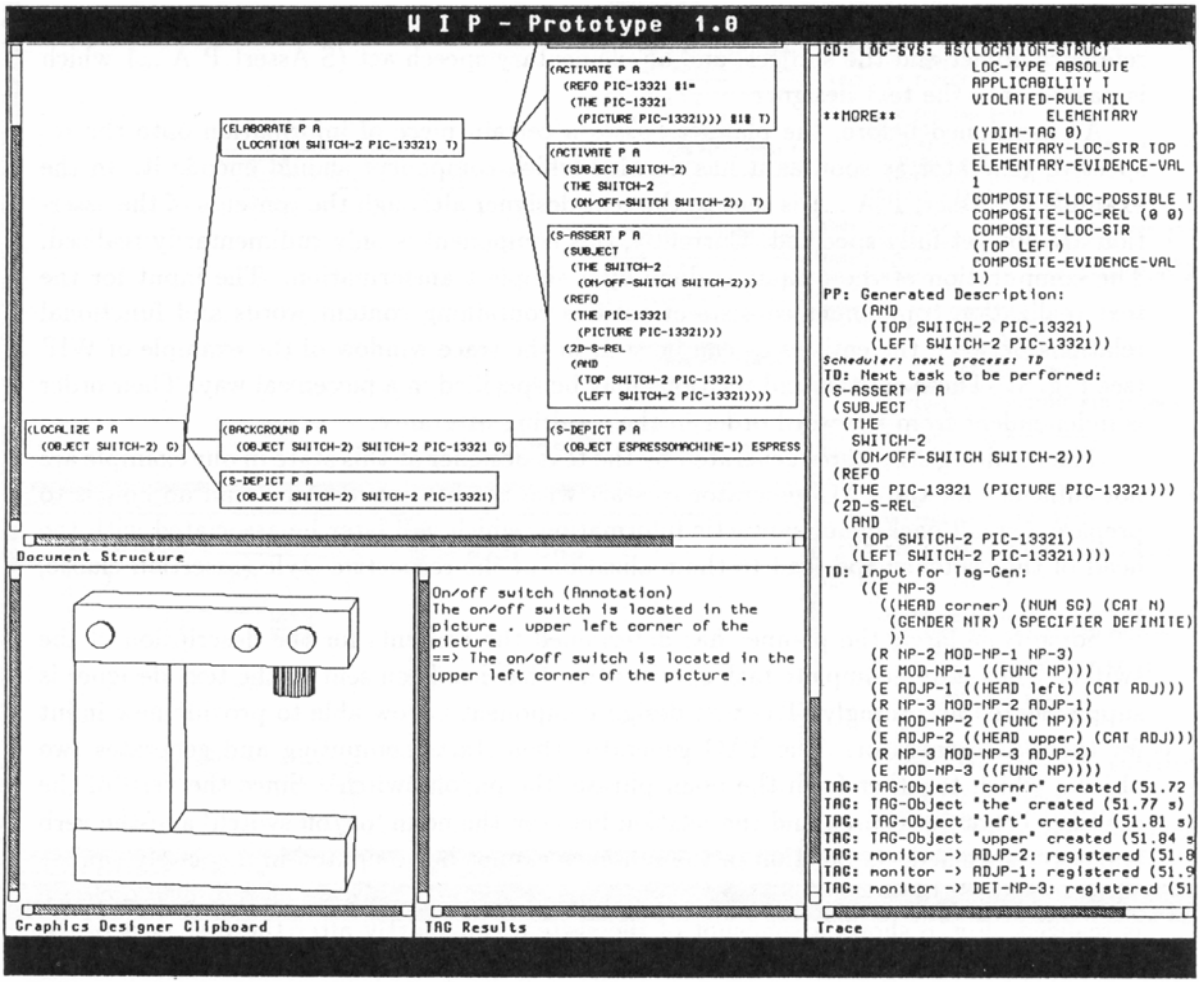
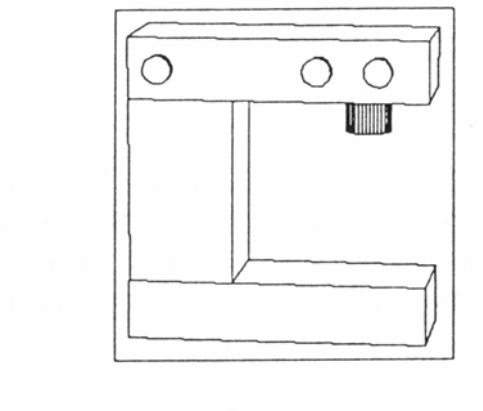


Figure 9: Snapshot 2

5 Conclusion

As a first step towards a computational mode for the generation of multimodal presentations we have conceived the knowledge-based presentation system WIP. In this paper, we described the architecture and the main components of the first implemented WIP prototype. The system can be considered as a testbed to examine various forms of interactions which are necessary to tailor textual and pictorial output to each other. The experience we gained from this prototype provides a good basis for a deeper understanding of the interdependencies between text and graphics. In the future, we will not only concentrate on conceptual extensions but also evaluate the performance of the WIP system by adapting it to other domains. WIP is currently able to generate simple German or English explanations for using an espresso machine, assembling a lawn-mower, or installing a modem, demonstrating our claim of language and application independence.



The on/off switch is located in the upper left corner of the picture.

Figure10: Final Results

6 Implementation of the Prototype

The WIP system has been developed on a Symbolics XL 1200 Lisp machine and several MacIvory workstations. The modules are implemented in Symbolics Common Lisp using CLOS and Flavors for object-oriented programming.

The constraint-based positioning component CLAY of the layout manager has been implemented using ideas from the DeltaBlue algorithm and the forward checking mechanism from the CHIP system. First evaluations of a standalone prototype gained a high runtime efficiency.

The text realization component was conceived as a distributed parallel model in the framework of object-oriented concurrent programming. We use the Ivory boards in our local area network to run the processes in parallel.

The graphics realization component utilizes both facilities of the symbolics window system and the commercial software packages S-Geometry and S-Render. Representations of domain objects comprise wire-frame models which are based on the modelling primitives provided by S-Geometry.

Acknowledgements

The development of WIP is an ongoing group effort and has benefited from the contributions of our colleagues Karin Harbusch, Jochen Heinsohn, Bernhard Nebel, and Hans-Jürgen Profitlich as well as our students Andreas Butz, Bernd Herrmann, Antonio Krüger, Daniel Kudenko, Wolfgang Maaß, Peter Poller, Georg Schneider, Frank Schneiderlöchner, Christoph Schommer, Dudung Soetopo, and Detlev Zimmermann.

References

[André & Rist 90a] E. **André** and **T. Rist.** *Synthesizing Illustrated Documents: A Plan-Based Approach.* In: Proceedings of InfoJapan '90, Vol. 2, pp. 163-170, Tokyo, 1990. Also DFKI Research Report RR-91-06.

[André & Rist 90b] E. **André** and T. **Rist**. *Towards a Plan-Based Synthesis of Illustrated Documents*. In: Proceedings of the 9th European Conference on Artificial Intelligence, pp. 25-30, Stockholm, Sweden, July 1990. Also DFKI Research Report RR-90-11.

[André & Rist 92] E. **André** and T. **Rist**. *The Design of Illustrated Documents as a Planning Task*. In present volume, 1992.

[Arens et al. 92] Y. **Arens**, E. H. **Hovy**, and M. **Vossers**. *The Knowledge Underlying Multimedia Presentations*. In present volume, 1992.

[Badler et al. 91a] N. **Badler**, B. **Barsky**, and D. **Zeltzer** (eds.). *Making Them Move: Mechanics, Control, and Animation of Articulated Figures*. Los Altos, CA: Morgan Kaufmann, 1991.

[Badler et al. 91b] N. **Badler**, B. **Webber**, J. **Kalita**, and J. **Esakov**. *Animation from Instructions*. In: Badler et al. [Badler et al. 91a], pp. 51-93.

[Beach 85] R. **Beach**. *Setting Tables and Illustrations with Style*. PhD thesis, Dept. of Computer Science, University of Waterloo, Ontario, 1985.

[Borning et al. 87] A. **Borning**, R. **Duisberg**, B. **Freeman-Benson**, A. **Kramer**, and M. **Woolf**. *Constraint Hierarchies*. In: Proceedings of OOPSLA '87, pp. 48-60, October 1987.

[Feiner & McKeown 92] S. K. **Feiner** and K. R. **McKeown**. *Automating the Generation of Coordinated Multimedia Explanations*. In present volume, 1992.

[Feiner 88] S. **Feiner**. *A Grid-Based Approach to Automating Display Layout*. In: Proceedings of the Graphics Interface '88, pp. 192-197. Los Altos, CA: Morgan Kaufmann, June 1988.

[Finkler & Neumann 88] W. **Finkler** and G. **Neumann**. *MORPHIX: A Fast Realization of a Classification-Based Approach to Morphology*. In: Proceedings of the Workshop 'Wissensbasierte Sprachverarbeitung', Berlin, Germany, 1988. Springer-Verlag.

[Finkler & Schauder 92] W. **Finkler** and A. **Schauder**. *Effects of Incremental Output on Incremental Natural Language Generation*. In: Proceedings of the 10th European Conference on Artificial Intelligence, Vienna, Austria, August 1992.

[Freeman-Benson et al. 90] B. **Freeman-Benson**, J. **Maloney**, and A. **Borning**. *An Incremental Constraint Solver*. Communications of the ACM, 33(l):54-63, 1990.

[Graf & Maaß 91] W. **Graf** and W. **Maaß**. *Constraint-basierte Verarbeitung graphischen Wissens*. In: W. Brauer and D. Hernandez (eds.), Proceedings of 4. Internationaler GI-Kongreß Wissensbasierte Systeme - Verteilte KI und kooperatives Arbeiten, pp. 243-253. Berlin, Germany: Springer-Verlag, October 1991. Also DFKI Research Report RR-91-35.

[Graf 92] W. **Graf**. *Constraint-Based Graphical Layout of Multimodal Presentations*. In: Proceedings of AVT92 (Advanced Visual Interfaces), Rome, Italy, May 1992. Also DFKI Research Report RR-92-15.

[Harbusch et al. 91] K. **Harbusch**, W. **Finkler**, and A. **Schauder**. *Incremental Syntax Generation with Tree Adjoining Grammars*. In: W. Brauer and D. Hernandez (eds.), Proceedings of 4. Internationaler GI-Kongreß Wissensbasierte Systeme -Verteilte KI und kooperatives Arbeiten, pp. 363-374. Berlin, Germany: Springer-Verlag, October 1991.

[Heinsohn et al. 92] J. **Heinsohn**, D. **Kudenko**, B. **Nebel**, and H.-J. **Profitlich**. *RAT-Representation of Actions in Terminoligical Logics*. In: J. Heinsohn and B. Hollunder (eds.), Proceedings of the DFKI Workshop on Taxonomic Reasoning, DFKI Document D-92-08, pp. 16-22. Saarbrücken, Germany, May 1992. February.

[Hentenryck 89] P. Van **Hentenryck** (ed.). *Constraint Satisfaction in Logic Programming*. Cambridge, MA: MIT Press, 1989. Revision of Ph.D. thesis, University of Namur, 1987.

[Joshi 85] A. **Joshi**. *An Introduction to TAGs*. Technical report, MS-CIS-86-64, LINC-LAB-31, Dept. of Computer and Information Science, Moore School, University of Pennsylvania, 1985.

[Kerpedjiev 92] S. M. **Kerpedjiev**. *Automatic Generation of Multimodal Weather Reports from Datasets*. In: Proceedings of the 3rd ACL Conference on Applied Natural Language Processing (ANLP-92), Trento, Italy, pp. 48-55, 1992.

[Levelt 89] W. **Levelt** (ed.). *Speaking: From Intention to Articulation*. Cambridge, MA: MIT Press, 1989.

[Maaß 92] W. **Maaß**. *Constraint-basierte Platzierung in multimodalen Dokumenten am Beispiel des Layout-Managers in WIP*. Master's thesis, Dept. of Computer Science, University of Saarbrücken, January 1992.

[Mackworth 77] A. **Mackworth**. *Consistency in Networks of Relations*. Artificial Intelligence, 8(1):99-l 18, 1977.

[Mann & Thompson 88] W. **Mann** and S. **Thompson**. *Rhetorical Structure Theory: Towards a Functional Theory of Text Organization*. TEXT, 8(3), 1988.

[Marks & Reiter 90] J. **Marks** and E. **Reiter**. *Avoiding Unwanted Conversational Implicatures in Text and Graphics*. In: Proceedings of the 8th National Conference of the American Association for Artificial Intelligence, pp. 450-456, Boston, MA, July 1990.

[Maybury 92] M. **Maybury**. *Planning Multimedia Explanations Using Communicative Acts*. In present volume, 1992.

[Müller-Brockmann 81] J. **Müller-Brockmann** (ed.). *Grid Systems in Graphic Design*. Niederteufen, Switzerland: Verlag Arthur Niggli, 1981.

[Ortony et al. 92] A. **Ortony**, J. **Slack**, and O. **Stock** (eds.). *Communication from an Artificial Intelligence Perspective: Theoretical and Applied Issues*. Berlin, Germany: Springer-Verlag, 1992. In press.

[Rist & André 92a] T. **Rist** and E. **André**. *From Presentation Tasks to Pictures: Towards a Computational Approach to Automatic Graphics Design*. In: Proceedings of the 10th European Conference on Artificial Intelligence, Vienna, Austria, August 1992.

[Rist & André 92b] T. **Rist** and E. **André**. *Incorporating Graphics Design and Realization into the Multimodal Presentation System WIP*. In: Proceedings of AVI'92 (Advanced Visual Interfaces), Rome, Italy, May 1992.

[Roth & Heffley 92] S. **Roth** and W. **Heffley**. *Intelligent Multimedia Presentation Systems: Research and Principles*. In present volume, 1992.

[Roth et al. 91] S. **Roth**, J. **Mattis**, and X. **Mesnard**. *Graphics and Natural Language as Components of Automatic Explanation*. In: Sullivan and Tyler [Sullivan & Tyler 91], pp. 207-240. SAGE.

[Schauder 92] A. **Schauder**. *Incremental Syntactic Generation of Natural Language with Tree Adjoining Grammars*. Technical Memo, German Research Center of Artificial Intelligence (DFKI), 1992.

[Sullivan & Tyler 91] J. **Sullivan** and S. **Tyler** (eds.). *Intelligent User Interfaces*. Frontier Series. New York, NY: ACM Press, 1991.

[Wahlster et al. 91] W. **Wahlster**, E. **André**, W. **Graf**, and T. **Rist**. *Designing Illustrated Texts: How Language Production Is Influenced by Graphics Generation*. In: Proceedings of the 5th Conference of the European Chapter of the Association for Computational Linguistics, pp. 8-14. Berlin, Germany: Springer-Verlag, April 1991. Also DFKI Research Report RR-91-05.

[Wahlster et al. 92a] W. **Wahlster**, E. **André**, S. **Bandyopadhyay**, W. **Graf**, and T. **Rist**. *WIP: The Coordinated Generation of Multimodal Presentations from a Common Representation*. In: Ortony et al. [Ortony et al. 92], pp. 121-144. Also DFKI Research Report RR-91-08.

[Wahlster et al. 92b] W. **Wahlster**, E. **André**, W. **Finkler**, H.-J. **Profitlich**, and T. **Rist**. *Plan-based Integration of Natural Language and Graphics Generation*. German Research Center for Artificial Intelligence, DFKI Research Report, 1992.

[Wazinski 92] P. **Wazinski**. *Generating Spatial Descriptions for Cross-modal References*. In: Proceedings of the 3rd Conference on Applied Natural Language Processing (ANLP-92), Trento, Italy, pp. 56-63, 1992.