Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

Research
Report
RR-92-15

Constraint-Based Graphical Layout of Multimodal Presentations

Winfried Graf

January 1992

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH

Postfach 20 80
D-6750 Kaiserslautern, FRG
Tel.: (+49 631) 205-3211/13
Fax: (+49 631) 205-3210

Stuhlsatzenhausweg 3
D-6600 Saarbrücken 11, FRG
Tel.: (+49 681) 302-5252
Fax: (+49 681) 302-5341

Deutsches Forschungszentrum für Künstliche Intelligenz

Constraint-Based Graphical Layout of Multimodal Presentations

Winfried Graf

DFKI-RR-92-15

Submitted to AVI'92 (Advanced Visual Interfaces), Roma, Italy, May, 1992

This work has been supported by a grant from The Federal Ministry for Research and Technology (FKZ ITW-89018).

Constraint-Based Graphical Layout of Multimodal Presentations

Winfried Graf

German Research Center for Artificial Intelligence (DFKI)
Stuhlsatzenhausweg 3, W-6600 Saarbrücken 11, Germany
Phone: (+49 681) 302-5264
Fax: (+49 681) 302-5341
E-mail: graf@dfki.uni-sb.de

Abstract

When developing advanced multimodal interfaces, combining the characteristics of different modalities such as natural language, graphics, animation, virtual realities, etc., the question of automatically designing the graphical layout of such presentations in an appropriate format becomes increasingly important. So, to communicate information to the user in an expressive and effective way, a knowledge-based layout component has to be integrated into the architecture of an intelligent presentation system. In order to achieve a coherent output, it must be able to reflect certain semantic and pragmatic relations specified by a presentation planner to arrange the visual appearance of a mixture of textual and graphic fragments delivered by mode-specific generators.

In this paper we will illustrate by the example of *LayLab*, the layout manager of the multimodal presentation system WIP, how the complex positioning problem for multimodal information can be treated as a constraint satisfaction problem. The design of an aesthetically pleasing layout is characterized as a combination of a general search problem in a finite discrete search space and an optimization problem. Therefore, we have integrated two dedicated constraint solvers, an incremental hierarchy solver and a finite domain solver, in a layered constraint solver model *CLAY*, which is triggered from a common metalevel by rules and defaults. The underlying constraint language is able to encode graphical design knowledge expressed by semantic/pragmatic, geometrical/topological, and temporal relations. Furthermore, this mechanism allows one to prioritize the constraints as well as to handle constraint solving over finite domains. As graphical constraints frequently have only local effects, they are incrementally generated by the system on the fly. Ultimately, we will illustrate the funcionality of LayLab by some snapshots of an example run.

Contents

1 Introduction

Due to the growing complexity of information that has to be communicated by current AI systems, there comes an increasing need for building sophisticated intelligent user interfaces that take advantage of a coordinated combination of different modalities, e.g., natural language, graphics, animation, and virtual realities to produce a flexible and efficient information presentation (cf. also [ST91,WAB+92,May92]).

When developing advanced visual user interfaces composing coordinated text and graphics as in the system *WIP* (Knowledge-based Presentation of Information), the question of laying out such multimodal presentations in an appropriate format becomes increasingly important. Here, the graphical communication of multimodal information plays a crucial role.

As with many other interesting AI design problems, the determination of a complex layout can be viewed as a discrete combinatorial problem, i.e., finding in a finite discrete search space a point satisfying set of constraints. In contrast to other configuration tasks, e.g., hardware design, an optimal layout of a multimodal presentation has to satisfy certain additional aesthetic criteria. We treat layout as a *constraint satisfaction problem* (CSP) that can be solved efficiently by consistency techniques to prune the search space a priori.

Therefore, this paper details a constraint-based approach for processing design-relevant graphical knowledge for automatic layout, either as a formalism for specifying basic design principles such as gridding, alignment, symmetry, etc., or as a mechanism for propagating obligatory, optional, and default constraints to position individual layout fragments on a graphic design grid (cf. [Gra91,GM91]).

So, the arrangement of an aesthetically pleasing layout is characterized as a combination of a general search problem in a finite discrete search space and an optimization problem. Both problems are addressed by an advanced constraint solver model comprising two dedicated solvers for handling different kinds of graphical constraints defined on constraint hierarchies and finite domains.

We will illustrate the use of these constraint techniques by the example of *LayLab*, WIP's experimental layout manager. 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. WIP generates illustrated texts that are customized for the intended audience and situation (see also [WAGR91,WAB+92,AFG+92]).

A fundamental goal of our work is to construct a universal framework for automatic layout management, as an integrated component of a multimodal presentation system, that makes intelligent use of human visual abilities and generation parameters, such as user stereotypes, output devices, etc., whenever arranging multimodal objects in any kind of presentation. Thus, from the functional viewpoint the main task of a knowledge-based layout manager is to convey certain semantic and pragmatic relations specified by a presentation planner by the arrangement of graphics and text fragments delivered by mode-specific generators, i.e., to determine the size of the layout elements and the exact coordinates for positioning them on the document page.

2 Related Research

As graphics hardware becomes more and more sophisticated, computer-based graphical communication achieves a crucial role in intelligent user interfaces. While much research in this area has focused on the automatic synthesis of graphics for either presenting relational information (cf. [Mac85]) and realistic depictions of 3D objects as in [SF91,RA92], the automatic layout design of graphical presentations is relatively unexplored.

Some interesting early efforts in automating layout enclose Eastman's work on a *General Space Planner* that addressed the task of arranging objects (e.g., furniture) in a space subject to given constraints (cf. [BF81]), chap. III). Feiner's *GRIDS* (GRaphical Interface Design System) was constructed as an experimental system to investigate approaches in the automatic display layout of text, illustrations, and later virtual input devices (cf. [Fei88]). In contrast to, e.g., WIP, the generation of the contents is separated from that of the style of a presentation. The current version of the testbed system has been implemented using an OPS5-like production language. The layout process is guided by the concept of a graphical design grid. Other approaches using computer-based grids, modeled by a human designer, can be found in the system *VIEW* (cf. [Fri84]) for synthesizing graphical object depictions from high-level specifications and by [Bea85] for low-level table layout, whose high-level topology was specified by the user as a matrix.

Since constraint-satisfaction techniques have become more sophisticated during the last decade, and with the growing availability of advanced graphics hardware, there has been an upward trend in applying constraint techniques to user interface design. Thus, most of the related work on applications of constraint languages and systems has been done in the area of graphical interfaces, especially geometric layout.

A pioneering system in both constraint satisfaction and interactive graphics was *Sketchpad* (cf. [Sut63]) written by Sutherland at MIT in 1963. Using the Sketchpad system allowed a user to create complex objects by sketching primitive graphical entities and specifying constraints on them. Many of these ideas have been explored by Borning in the *ThingLab* system at Xerox PARC [Bor81], a graphical constraint-oriented simulation laboratory implemented in Smalltalk-80. Later versions of ThingLab were concerned with extensions supporting constraint hierarchies, incremental compilation, and graphical facilities for defining new kinds of constraints (e.g. [BDFB+87,FBMB90]). Both systems exploit numeric techniques such as relaxation for solving constraint networks containing cycles, in contrast to symbolic techniques, e.g., used in Steele's constraint language (cf. [SS80]). Other research activities in constraint-based graphics include the systems *Juno* [Nel85], *IDEAL* [vW82], *Magritte* [Gos83], *Bertrand* (cf. [Lel88]), and the work of Cohen et al. on constraint-based tiled windows (cf. [CSI86]).

An increasing number of interface-design systems mostly based on a graphical editor have been developed during the last years to make the interface design process more efficient and comfortable than with conventional techniques. Here, constraints provide a means of stating layout requirements, e.g., the *Peridot* system deduces constraints automatically as the user demonstrates the desired behaviour (cf. [Mye91]).

Other representative research related to in this paper has concentrated more on the theoretical background of constraint languages (see also [Lel88]) including *constraint logic programming* (cf. [JJL87]). One example is *CHIP* (cf. [DHS+88,Hen89]), a new logic language developed at the ECRC in Munich, combining the declarative aspects of logic programming with the efficiency of consistency techniques. A special feature is its facility of solving constraints defined over finite domains that enables CHIP to solve a variety of discrete combinatorial search problems.

3 Multimodal Layout

Design is a central skill in many human tasks such as layout of multimodal presentations. In general, we have to distinguish functional versus art design. In our context, design means the arrangement of an efficient and expressive presentation subject to given restrictions, e.g., regarding filled versus empty space and non-overlapping, that further satisfies some aesthetic and functional criteria, such as transparency, legibility, objectivity, or credibility.

Layout design is essentially influenced by ideas and approaches known from general graphics design (e.g., [Arn66,KM89]). So, we will transfer the graphical presentation problem as it is defined in [Mac85] to *multimodal objects* (i.e., textual, graphic, or virtual objects) and tackle it with graphical techniques. Thus, the *multimodal presentation problem* is to synthesize a multimodal design that effectively expresses a set of semantic and pragmatic relations and their structural properties. As we will not treat content issues of automatic presentations here, presentation design means the arrangement of the outer form of a document. According to (cf. [Mac85]), a design can be judged to be effective when it can be interpreted accurately or quickly, when it has visual impact, or when it can be realized in a cost-effective manner.

Beach (cf. [Bea85]) has shown that the general layout problem formalized as a random packing problem, i.e., determining whether an unordered set of non-overlapping rectangular table entries can be arranged into a minimum space, is strongly *NP-complete* and thus, there is no general and efficient algorithm for solving it. So even the question of finding an aesthetically pleasing layout for multimodal documents under certain outward restrictions seems to be intractable. The problem becomes linear in time only if we use additonal constraints to reduce the design space for arrangement, e.g., sufficient grid structures.

3.1 The Grid-based Approach

Basic formal design aspects have been addressed by several graphical techniques such as typographic grids (see [MB81,Rüe89]). The grid partitions a 2D plane into smaller rectangular units of mostly equal size, so-called 'universal' grid fields, using horizontal and vertical lines. The fields are separated by an intermediate space depending on the size of the type characters and the illustrations. A grid-based approach builds a logical and constant basis for the designer, but does not limit his creativity. So, a grid can be treated as an ordering system for efficiently designing functional (i.e., uniform, coherent, and consistent) layouts. This method is also used in Feiner's expert system for automatically laying out displays containing text and pictures.

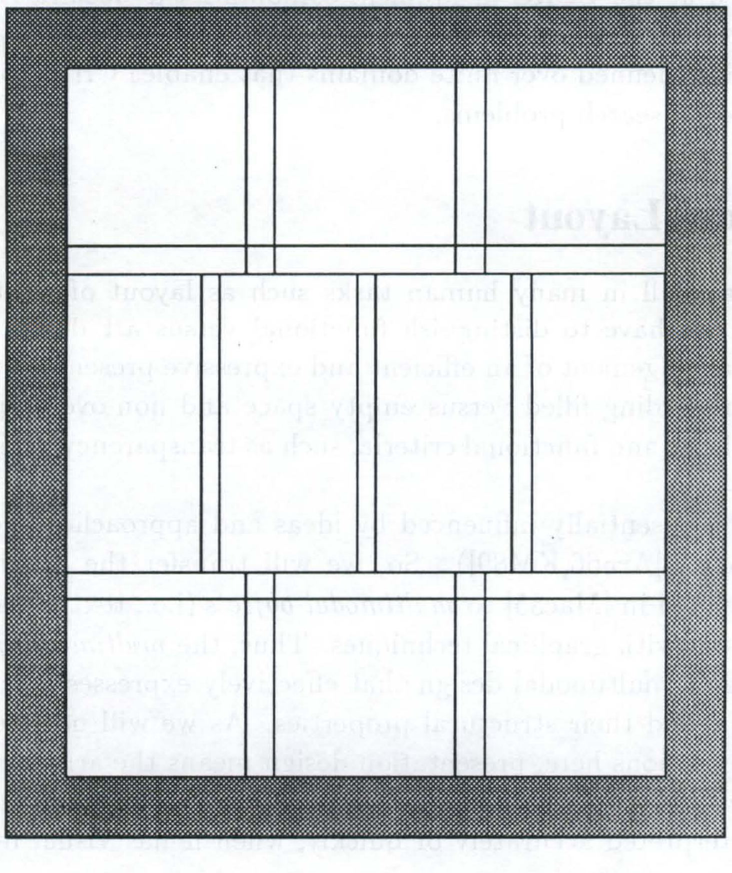


Figure 1: Structure of a Superimposed Design Grid

The granularity of the grid can be determined by the number of grid fields. To gain more flexibility the concept of *superimposed grids*, i.e., a multiple set of grids comprising different type areas (e.g., combining an odd and even number of columns) for one document page based on the information to be presented, has become useful for major newspapers (cf. Fig. 1). The design of an optimal grid for a specific presentation is a difficult task, including decisions on the dimensions of the type area, its divisions, and its margins. For a variation of the selected grid, a priori knowledge about the typographic elements is a necessary prerequiste.

3.2 Classifying Multimodal Relationships

The logical structure of a multimodal document is essentially characterized by functional dependencies between the different document parts, e.g., a mixture of text and graphics, which may describe actions themselves. In empiric psychological studies [WH87] have observed that there is large variability of layout patterns, such as pictures sandwiched within text, surrounding the text, superimposed over the text, and interspersed throughout the text, when placing illustrations with respect to corresponding text.

Semantic and pragmatic relationships between graphic and textual fragments are represented by WIP's presentation planner (cf. [AR90,AR92]) by means of so-called *rhetorical relations* based on the *RST-theory* proposed by [MT88] for text planning. Examples of RST-relations are 'sequence', 'contrast', 'elaboration', 'organization', 'motivation', etc. Some of these semantic-pragmatic coherence relations between major document parts can be easily reflected by graphic constraints representing topological arrangement facilities for visualization.

Relationships among multimodal layout objects on a document can be classified in, *local* relations between objects that are semantically connected and *global* relations regarding the whole document. Local relationships correspond to the topology while global ones are related to the geometric process. These relationships have to be considered in order to achieve a coherent and consistent presentation.

So, the general layout process follows the strategy of building larger units from groups of objects connected by local topological relations first and to ultimately arrange these units following global relations. This approach favors the idea of arranging identical structures in a similar way. By that, we can express the basic design principle, that the same layout decisions should be employed in different parts of a presentation. This concept also addresses the question of rhythm and visual balance.

As a well-designed layout is determined by its visual clarity, semantic dependencies and contradictions in the information to be presented are reflected by building larger units of locally connected layout elements. In contrast to local relationships, global relationships handle larger units of topological arranged objects by basic design strategies.

Graphical layout of multimodal presentations comprises most features of visual interface design, including the overall display layout, use of color and grey levels, as well as the placement of the individual layout elements with respect to their internal semantic relations. A well-designed expressive and effective layout is determined essentially by its clarity, consistency, and attractive aesthetic appearence. Visual clarity of a presentation can be achieved by using a visual organization of information that emphasizes the underlying logical structure. Therefore, a few global design rules regarding similarity, proximity, closure, and good continuation have been stated by different *Gestalt psychologists* (e.g., [Arn66]). For placing the designed presentation elements in the overall context, basic layout principles that concern gridding, proportion, and balance have been established.

4 Encoding Graphical Knowledge via Constraints

A central idea underlying automatic layout is the incorporation of causal geometric knowledge into the layout design process in order to achieve a deep encoding of perceptual criteria, i.e., design strategies can be used for generating the layout of every kind of multimodal presentation independent of the current domain.

We will address the problem of encoding this very heterogeneous types of design relevant knowledge, e.g., graphical and psychological information about perceptual criteria expressing horizontal versus vertical alignment, grouping, symmetry, similarity, and ordering by using constraint formalisms.

To influence the interface design and to determine the graphical style of the multimodal presentation, application domain-specific knowledge as well as common-sense knowledge about basic design heuristics have to be encoded. But some of the layout issues can only be settled by basic design heuristics or individual preferences.

4.1 Constraint Representation

Layout as a configuration task can be viewed as a combinatorial problem, in a finite discrete search space. The main problem consists of finding any solution that satisfies all topological and geometrical restrictions with regard to certain aesthetic criteria. So, the design of an optimal layout can be treated as a combination of a general search problem with an optimization problem.

In the following, we will view layout as a Boolean CSP. That is, one has a set V of variables $v_1, ..., v_n$ each to be instantiated in an associated domain D_i and a set of Boolean constraints limiting the set of allowed values for specified subsets of the variables. There are various forms of definitions for the terminus 'constraint'. Informally, a constraint expresses desired relations among one or more objects. In this paper, we will stay to a general definition for CSP given by [Mac77,Mac88] as a formula in first-order predicate logic restricted to unary and binary predicates:

$$\begin{aligned}(\exists x_1)(\exists x_2)...(\exists x_n) \quad & (x_1 \in D_1)(x_2 \in D_2)...(x_n \in D_n) \\ & P_1(x_1) \wedge P_2(x_2) \wedge ... \wedge P_n(x_n) \wedge \\ & P_{12}(x_1, x_2) \wedge P_{13}(x_1, x_3) \wedge ... \wedge P_{n-1,n}(x_{n-1}, x_n).\end{aligned}$$

Here P_{ij} is only included in the wff if $i < j$ since it is assumed that $P_{ji}(v_j, v_i) = P_{ij}(v_i, v_j)$.

With regard to this definition, layout planning as a CSP can be formalized in the following manner: the position of each layout item is a variable, with an associated domain that contains an infinite number of n-tuples of real values. Those domains can be described intensionally by constraints, e.g., specifying the boundaries of the connected subspaces permitted for that item. Constraints, e.g. ",The header has to be placed in the upper left corner," must also be expressed intensionally by algebraic equations and inequalities on the values of the constraint variables. Moreover, one might have p-ary relations such as "Text-1 must be between Graphics-2 and Graphics-3".

Intuitively, a constraint definition comprises a declarative representation of the relation and a set of procedures that can be invoked by the underlying constraint solver to fufill it. We will treat these procedures as methods in an object-oriented sense that can alternately try to satisfy the constraint. A constraint network may be *underconstrained*, in which case there are additional degrees of freedom, or *overconstrained*, in which case some of the constraints may not be satisfied.

Constraint networks provide an elegant mechanism to declaratively state design-relevant knowledge about geometrical and topological relationships, characterizing properties among graphical objects that can be maintained by the underlying system (e.g., [BD86,KPKY91]). Constraints can represent very heterogenious relationships between different kinds of multimodal objects. They are used extensively, e.g., in computer graphics and graphical interfaces, and in the specification of relationships between graphical objects. Constraints can be easily used to specify layout requirements in graphical environments in order to guarantee local circumscriptions of the presentation like format restrictions, margins, distances, and non-overlapping, or to maintain consistency among objects on the whole document.

The declarative semantics of constraint languages allow one to specify graphical objects while avoiding extraneous concerns about the realization of the drawing algorithms. Another major advantage is their ability to describe complex objects simply and naturally.

Our approach uses constraint programming techniques to declaratively represent aesthetic knowledge, e.g., basic design principles expressing perceptual criteria, as well as to compute the precise position and size of a set of multimodal elements.

Layout constraints can be classified as *semantic/pragmatic*, *geometrical/topological*, and *temporal*. Semantic and pragmatic constraints essentially correspond to coherence relations like the RST-relations described above. They can be compiled into graphic constraints that specify perceptual criteria concerning the organization of the visual elements, such as the sequential ordering (horizontal versus vertical layout), alignment, grouping, symmetry, or similarity. Geometrical and topological constraints refer to absolute and relative constraints. Furthermore, we can classify constraints in absolute ones that fix geometric parameters to constant values (e.g., absolute coordinates) and in those that relate a geometric parameter of one object to another (relative distance). Temporal relations are used in the case of animated presentations to represent temporal, spatial information.

Elementary constraints of the constraint language are defined by the arithmetic functions of the underlying Symbolics Lisp system. *Primitive constraints* represent elementary local relations, e.g., 'beside', 'connect', or 'under', expressing basic geometric relations. These constraints are specified by sets of mathematical equations (e.g., two objects that are constrained to touch at specific points) or by sets of inequalities (e.g., one object is constrained to lie inside another). The primitive constraints can be aggregated to complexer *compound constraints*, specifying the visualization of semantic-pragmatic relations such as 'contrast' or 'sequence'.

To give an example of a typical compound constraint in a predicate logic-like notation, let's have a look at a section of the representation of the 'contrast' constraint. The corresponding constraint network is illustrated in Fig. 2.

$$
\begin{aligned}
\textbf{CONTRAST}\ (\mathrm{G}_1, \mathrm{G}_2) \leftrightarrow\ & \\
& \mathrm{G}_1 \equiv \mathrm{pkt}(\mathrm{x}_{\mathrm{G}_1}, \mathrm{y}_{\mathrm{G}_1}, \mathrm{wi}_{\mathrm{G}_1}, \mathrm{he}_{\mathrm{G}_1}) \wedge \\
& \mathrm{G}_2 \equiv \mathrm{pkt}(\mathrm{x}_{\mathrm{G}_2}, \mathrm{y}_{\mathrm{G}_2}, \mathrm{wi}_{\mathrm{G}_2}, \mathrm{he}_{\mathrm{G}_2}) \wedge \\
& [\mathrm{EQUAL}(y_{G_1}, y_{G_2}) \wedge \mathrm{BESIDE}(x_{G_1}, \mathrm{wi}_{G_1}, x_{G_2}) \\
& \vee \\
& \mathrm{EQUAL}(x_{G_1}, x_{G_2}) \wedge \mathrm{UNDER}(y_{G_1}, \mathrm{he}_{G_1}, y_{G_2})]
\end{aligned}
$$

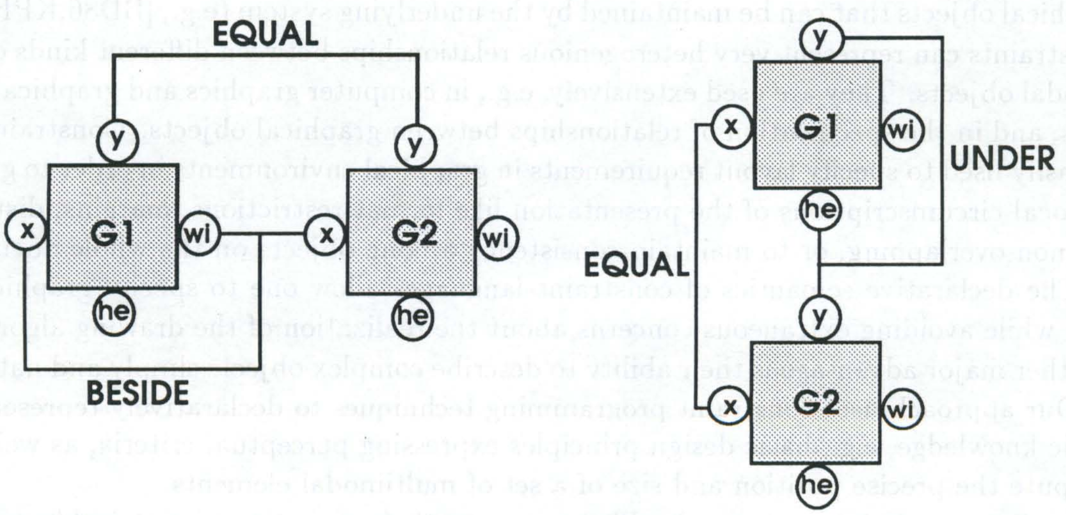


Figure 2: Constraint Network of the Definition Above

Since the ordering of the constraints in the definition is significant for their ranking, the stronger constraints have to preceed the weaker ones. E.g., according to the definition above, the layout manager will use a horizontal alignment in preference to a vertical one if a contrast-constraint has to be satisfied.

In graphical synthesis tasks like layout, constraints frequently have only local effects, i.e., the set of variables that are relevant to a solution and must be assigned values, changes dynamically as layout fragments are incrementally generated by a presentation system during problem solving. So, we have to distinguish *static constraints* that are related to a fixed set of variables and *dynamic constraints* that are generated on the fly (cf. [MF90]).

Another form of dynamic constraints concerns those in which the number of layout elements belonging to one relation is not known a priori (e.g., the 'sequence'). The definiton of a dynamic constraint for two sequentially related graphics-text blocks B_1, B_2 is given by the following logical formula:

$$
\begin{aligned}
&\mathrm{SEQUENCE}(\mathrm{B}_1, \mathrm{B}_2) \leftrightarrow \\
&\qquad \mathrm{B}_1 \equiv \mathrm{pkt}(\mathrm{x}_{\mathrm{B}_1}, \mathrm{y}_{\mathrm{B}_1}, \mathrm{wi}_{\mathrm{B}_1}, \mathrm{he}_{\mathrm{B}_1}) \wedge \\
&\qquad \mathrm{B}_2 \equiv \mathrm{pkt}(\mathrm{x}_{\mathrm{B}_2}, \mathrm{y}_{\mathrm{B}_2}, \mathrm{wi}_{\mathrm{B}_2}, \mathrm{he}_{\mathrm{B}_2}) \wedge \\
&\qquad [\mathrm{EQUAL}(x_{B_1}, x_{B_2}) \wedge \\
&\qquad \mathrm{UNDER}(y_{B_1}, \mathrm{he}_{B_1}, y_{B_2}) \wedge \\
&\qquad \mathrm{EQUAL}(y_{B_1}, ?\text{succ-y}) \wedge \\
&\qquad \mathrm{BESIDE}(?\text{succ-x}, \mathrm{wi}_{B_1})] \\
&\qquad \vee \\
&\qquad [\mathrm{EQUAL}(y_{B_1}, y_{B_2}) \wedge \\
&\qquad \mathrm{BESIDE}(x_{B_1}, \mathrm{wi}_{B_1}, x_{B_2}) \wedge \\
&\qquad \mathrm{EQUAL}(x_{B_1}, ?\text{succ-x}) \wedge \\
&\qquad \mathrm{UNDER}(?\text{succ-y}, \mathrm{he}_{B_1})],
\end{aligned}
$$

where ?succ-x, ?succ-y denote the x- and y-variables of the respective successor object.

The constraints regarded so far were concerned with local relationships between layout objects. Other global constraints refer to aggregated units of objects and represent global design strategies, e.g., that a unit should be preferably placed 'RightOf' another one. This is represented in the following way:

RIGHT-OF $(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8) \leftrightarrow$

$[\mathrm{FDADD}(\mathrm{FDMIN}(x_1), x_4) \wedge$;;; dx2- min
$\mathrm{FDMAXRIGHT}(\mathrm{MIN}(x_8, x_2)) \wedge$;;; dx2-max
$\mathrm{FDMIN}(x_2) \wedge$;;; dy2-min
$\mathrm{FDDIFF}(\mathrm{FDMAX}(x_2), \mathrm{FDDIFF}(x_7, x_3))] \wedge$;;;dy2-max
$\vee$
$[\mathrm{FDMIN}(x_1) \wedge$;;; dx1-min
$\mathrm{FDDIFF}(\mathrm{FDMAX}(x_5), x_4)) \wedge$;;; dx1-max
$\mathrm{FDMIN}(x_2) \wedge$;;; dy1-min
$\mathrm{FDDIFF}(\mathrm{FDMAX}(x_6), \mathrm{FDDIFF}(x_3, x_7))] \wedge$;;; dy1-max

Another problem addresses the handling of default constraints representing default assumptions, i.e., assigning a default value to a variable unless another value is known. To avoid trivial solutions to a constraint problem one has to express negative defaults explicitly.

4.2 Constraint Satisfaction

The constraint solving process ensures that all constraints will be satisfied when the individual layout elements are assembled into a complete layout. In problem solving, one has to decide generality versus efficiency. Since efficient constraint satisfaction is crucial when using constraints for graphical tasks, currently available universal constraint systems seem to be incovenient because of their increasing run time. Therefore, we have focused on tailoring specific solving algorithms for handling the large variety of design constraints to our needs, especially by extending or modifying approaches from recent work on constraint logic programming cf. [JJL87]).

4.2.1 Handling of Constraint Hierarchies

Besides the semantic classification of local constraints outlined above, one often wants to prioritize the constraints in those which must be required and others which are preferably held and could be relaxed in the worst case.[1] If the various constraints are given a priority,

[1] We call a set of constraints, each labelled with a certain strength a *constraint hierarchy*. A theory of constraint hierarchies is described in [BDFB+87].

the most important and restrictive constraints can be satisfied first. A powerful way of expressing this layout feature is to organize the constraints in a hierarchy by assigning a preference 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.

A typical example of using a constraint hierarchy in geometric layout is the problem of specifying the relative distance between two graphic elements, e.g., communicating a contrast relation. The adequate aesthetic criteria can be represented by the following constraints of different strength:

- An obligatory constraint that specifies that the distance between two graphics G_1, G_2 has to be greater than zero.
- A disjunction of two optional constraints states that the graphics G_1, G_2 are preferably to be aligned side by side or else one below the other.

For solving local constraints that are compiled from semantic/pragmatic RST-relations between a set of layout objects, *SIVAS* (**S**ingle **VA**lue **S**olver), an incremental constraint solver that can handle constraint hierarchies, has been constructed.

The processing of constraint hierarchies is based on the *DeltaBlue algorithm* by Freeman-Benson (cf. [FBMB90]), an incremental algorithm that maintains an evolving solution to the constraint hierarchy as constraints are added and removed dynamically. DeltaBlue, as one instance of a spectrum of incremental hierarchical algorithms, has been constructed as an extension for *flat* constraint solvers that satisfy required constraints only. It is restricted to solving hierarchies of unique, non-cyclic constraints using a so-called *locally-predicate-better comparator* for finding an optimal solution.

The flat solver underlying SIVAS employs a *value inference* technique for constraint propagation of numeric constraints similar to Sketchpad and ThingLab. Using this technique, variables are labeled with constant values, and values of uninstantiated variables are determined from instantiated ones using constraints.

In contrast to hypermedia-based approaches to adaptive information presentation, in WIP the information to be presented is partially generated on the fly. Therefore, incrementality is also required in the automatic layout design. Therefore, SIVAS is able to dynamically generate layout constraints, i.e., to add constraints to and remove constraints from the network during runtime. That way, it achieves a high performance of the presentation design process by minimizing the cost of finding a new solution. It does this by exploiting knowledge of the previous solution.

4.2.2 Handling of Finite Domains

The high-level arrangement of larger units of layout elements on the design grid is accomplished by satisfying the global constraints that represent general geometric design strategies.

When processing global relations, we can impose the further restriction that the variable domains each consist of a finite number of discrete values, as we can treat the grid coordinates in the design space as finite domains. In this case, the constraining relations can be specified extensionally as the set of all p-tuples that fulfill the constraint.

FIDOS (**FI**nite **Do**main **S**olver) is an incremental solver for handling global graphical constraints that is based on a so-called *label inference* technique. Here, variables range over a set of values, and constraints are used to restrict these sets.

The complex space optimization problem can be efficiently solved by using consistency techniques based on the idea of a priori pruning. As the efficiency of standard backtracking algorithms suffers from its so-called thrashing behaviour, various intelligent inference techniques based on tree search procedures such as *delay* mechanisms (cf. [JJL87,DHS+88]), *check rules* and *looking ahead* (cf. [Hen89]), *guarded rules* and *residuation* (cf. [Smo91]) are surveyed to reduce or eliminate thrashing.

According to [Hen89] search procedures based on consistency techniques are best viewed as the iteration of the following steps:

1. Propagate the constraints as much as possible

2. Make an assumption on the values of some variables, until the problem is solved

We have employed the forward checking technique known from the CHIP system (cf. [Hen89]) based on the idea of a priori pruning, that is using constraints to reduce the search space before the discovery of a failure in contrast to conventional search techniques. Forward checking makes sure that each not-yet-assigned variable has at least one consistent value with the already assigned variables. This technique is well suited for early pruning of layout alternatives in order to efficiently satisfy global geometric constraints.

4.3 A Layered Constraint Solver Model

We have integrated these two special-purpose constraint solvers, SIVAS and FIDOS, for satisfying local topological and global geometrical relationships in a new architecture model that is organized hierarchically. Because it is very difficult and inefficient to represent design heuristics via constraints, we use rules and defaults on a metalevel to trigger the inference process. This model is proposed as a general framework for processing most of the layout-relevant graphical knowledge that can be encoded by means of constraints. Fig. 3 sketches the architecture of the layered constraint solver model. For a more detailed description refer to [Maa92].

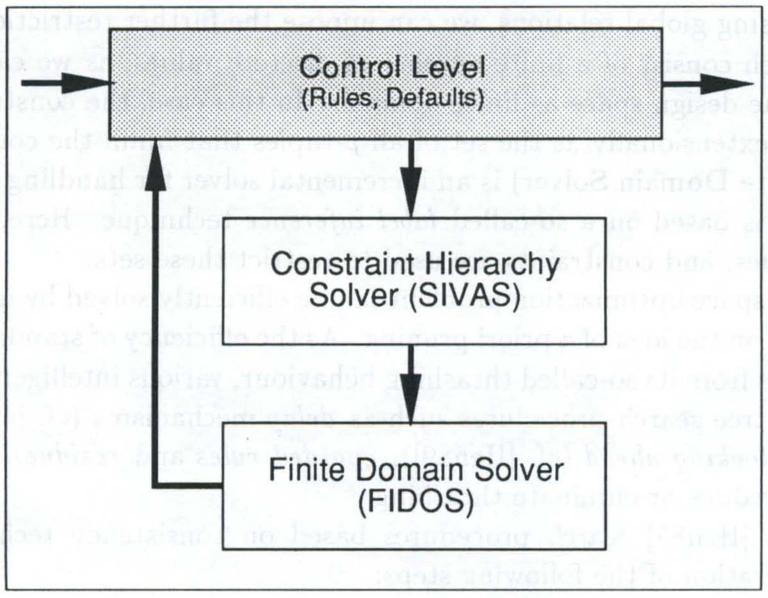


Figure 3: Model of a layered constraint solver

5 The Architecture of the *LayLab* System

In the following, by the example of *LayLab* WIP's layout manager, we propose a conceptual architecture for a knowledge-based layout component that automatically designs the graphical layout of multimodal presentations.

Various modules are embedded in the archicture of LayLab (see Fig. 4): a positioning component, an intelligent typographer, a document rendering component (beautifier), an interaction handler, and a knowledge-base. Let's have a closer look at its major components.

To communicate information to the user in an effective way, a knowledge-based presentation system must be able to reflect certain semantic and pragmatic relations specified by a presentation planner to automatically determine the outward appearance of the visualization and verbalization results delivered by the graphics and text generator. That is the main task of the constraint-based positioning component *CLAY*. CLAY exploits the two constraint solvers SIVAS and FIDOS outlined in section 4.2. to arrange the multimodal layout elements in the design space (see also [Gra91,GM91,Maa92]).

Since the placement process is essentially based on a graphic design grid as a framework for effective layout, we have constructed a rule-based *grid generation component* that automatically generates a set of superimposed grids, depending on different generation parameters such as output medium and user. Type area and text width are determined by legibility rules using default values for point size, font, distance, etc. Then, a uniform grid field is computed regarding the smallest picture or text block that holds an integral number of lines and scaling/cropping factors. If the grid is constructed, the individual design elements are adjusted to the size of the universal grid fields and fitted precisely

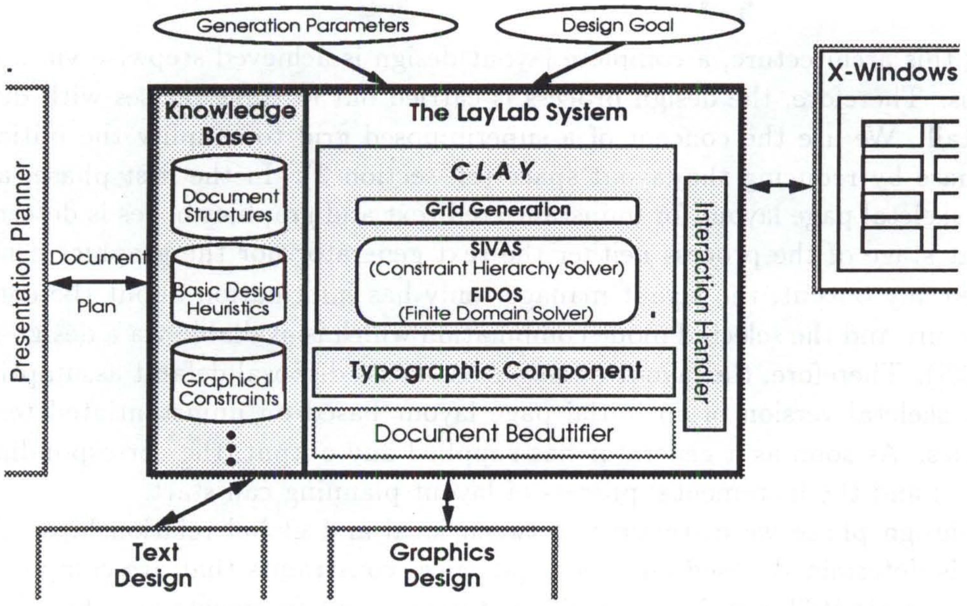


Figure 4: A Zoom into the Architecture of WIP's Layout Manager Showing the Various Submodules

into an integer number of fields, considering that the smallest grid field corresponds to the smallest text or graphic element to be presented (cf. also [MSS91]).

The *typographic component* addresses the problem of laying out text. It must decide not only how to organize the text, but also how to shape it. With [HA91] who treat text layout as a planning task, we distinguish different textual devices such as plain text, itemized lists, indented paragraphs, enumerations, sidebars, footnotes, italicizations, quotation, inserts, etc. Relations between these textual devices are treated analogously to text-picture combinations via constraints.

The determination of the graphical style of the individual layout objects, including the choice of colors or grey levels, and the beautifying of the entire document (e.g., design of the background) is performed by a *document rendering component* (cf. [PW85,Bea85]).

An *interaction handler* builds the interface to the hardware output medium (e.g., display) and presents the visualization and verbalization results of the mode-specific generators using special communication software (e.g., X-Windows).

LayLab's *knowledge base* contains information about document structures, rule-based representations of basic design heuristics as well as the graphical design constraints extensively illustrated before.

6 The Overall Layout Process

Considering this architecture, a complete layout design is achieved stepwise via a refinement process. Therefore, the design process is carried out in three phases with different levels of detail. We use the concept of a superimposed grid to simplify the initial construction phase by reducing the layout space (cf. section 3). In the first phase, a draft version of a skeletal page layout for uninstantiated text and graphics boxes is determined. Since at that stage of the process neither the text generator nor the graphics generator has produced any output, the layout manager only has information about the contents, the act structure and the selected mode combination which is available via a design record (cf. [AFG+92]). Therefore, the layout manager has at its disposal default assumptions to determine a skeletal version of an initial page layout based on uninstantiated text and graphics boxes. As soon as a generator has supplied any output, the corresponding box is instantiated and the incremental process of layout planning can start.

In that design phase we distinguish between local and global relationships. A low-level layout is determined based on local topological constraints that are compiled from semantic/pragmatic RST-relations specified between layout fragments and describe their relative topology. Finally, the high-level layout is computed as a global geometrical arrangement. In this step, aggregated units of locally connected objects are placed on the grid using global relations to determine the explicit coordinates.

Frequently, a draft layout has to be revised because of design constraints or visual imbalances in the output presentation. When a partially instantiated layout entered in the design record is evaluated by the layout manager with a negative result, a dependency-based layout revision process is initiated, i.e., revising the skeletal layout after contents planning, or in the worst case, revising the planned contents due to graphical constraints. A larger example showing the temporal coordination of contents planning and layout design is reported in [WAB+92].

Fig. 5 shows a snapshot of an example system run. In Fig. 6 CLAY's facility for automatically generating a graphical trace of the constraint satisfaction process is demonstrated. A larger example of a system run is outlined in [Maa92].

7 Integration and Implementation Aspects

The layout manager outlined above is integrated in the hierarchically organized architecture of the overall WIP system. This includes two parallel processing cascades for the incremental generation of text (cf. [HFS91]) and pictures including depictions of 3D objects (cf. [RA92]) and is moderated by a presentation planner (cf. [AR90,AR92]) and the layout manager. So, there is a coordination of the contents generation and the format design process. A so-called *design record* is employed to exchange information about intermediate results of the current presentation generation between the various components. To achieve a coherent output with an optimal media mix, the single components of an intelligent user interface have to be interleaved. Therefore, the layout manager has

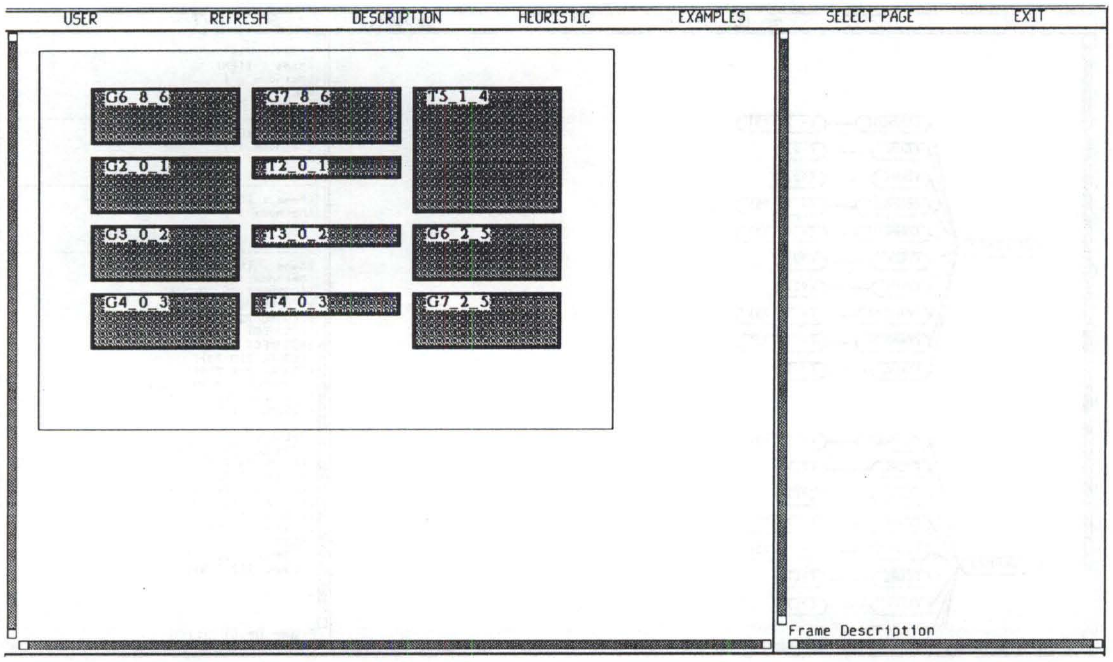


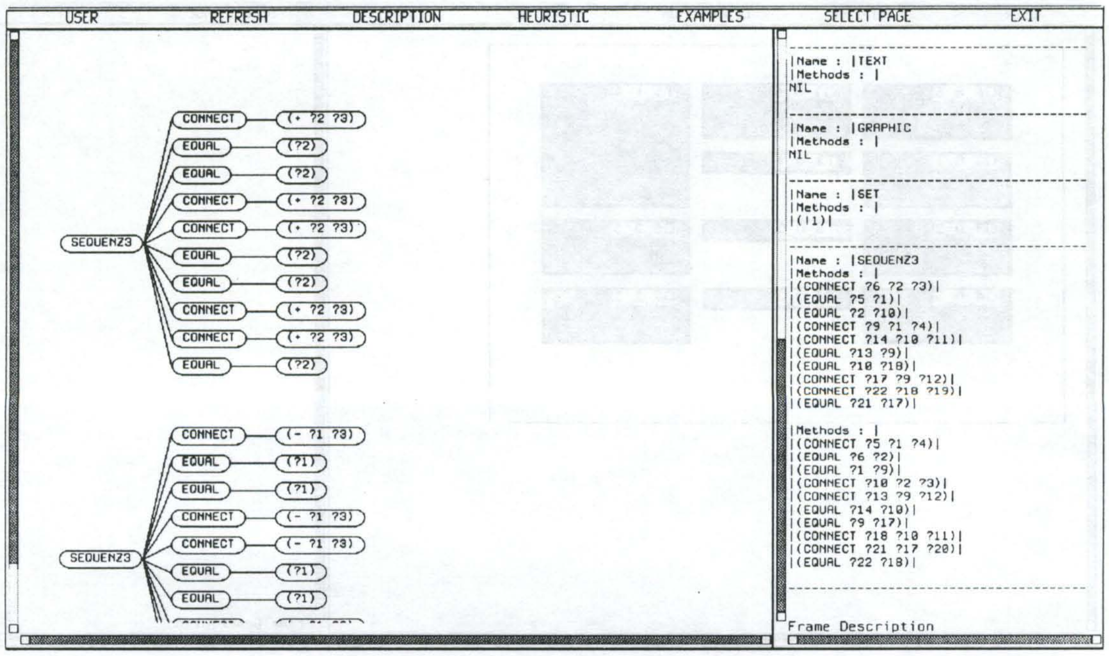
Figure 5: An Example Layout

to be integrated into the presentation planning process at an early stage to allow for an incremental refinement of the initial layout. The interplay of the various components is illustrated in a companion paper (see [WAB+92]).

The layout design process is influenced by most of WIP's generation parameters as design objectives, resource limitations, output modes (incremental vs. complete only), and layout formats (e.g., hardcopy, screen display, slides), the costs of different modalities' activations, the user's individual preferences, and more.

LayLab, a stand-alone prototype version of WIP's integrated layout manager, including CLAY, its constraint-based positioning component, has been implemented on a Symbolics XL 1200 Lisp machine and several MacIvory workstations under Genera 8.0 using Symbolics Common Lisp/CLOS and Flavors for object-oriented interface programming. The interaction handler and the typographer exploit features of the interface programming tools included in the Symbolics Lisp environment. These comprise window management utilities, which are compatible to the X-Windows system, and low-level text formatting routines. The constraint solvers SIVAS and FIDOS integrated in CLAY have been implemented by Wolfgang Maaß (for further information see [Maa92]).

First evaluations of the constraint solving process achieved a high runtime efficiency. Currently, CLAY is transported to UNIX workstations and is going to be tested in other domains and environments (e.g., configuration of technical devices). The work on the intelligent typographic and document rendering component is still in progress. Preliminary demonstrators have already been implemented.



```
----------------------------------------
|Name : |TEXT
|Methods : |
NIL
----------------------------------------
|Name : |GRAPHIC
|Methods : |
NIL
----------------------------------------
|Name : |SET
|Methods : |
|(|1)|
----------------------------------------
|Name : |SEQUENZ3
|Methods : |
|(CONNECT ?6 ?2 ?3)|
|(EQUAL ?5 ?1)|
|(EQUAL ?2 ?10)|
|(CONNECT ?9 ?1 ?4)|
|(CONNECT ?14 ?10 ?11)|
|(EQUAL ?13 ?9)|
|(EQUAL ?10 ?18)|
|(CONNECT ?17 ?9 ?12)|
|(CONNECT ?22 ?18 ?19)|
|(EQUAL ?21 ?17)|

|Methods : |
|(CONNECT ?5 ?1 ?4)|
|(EQUAL ?6 ?2)|
|(EQUAL ?1 ?9)|
|(CONNECT ?10 ?2 ?3)|
|(CONNECT ?13 ?9 ?12)|
|(EQUAL ?14 ?10)|
|(EQUAL ?9 ?17)|
|(CONNECT ?18 ?10 ?11)|
|(CONNECT ?21 ?17 ?20)|
|(EQUAL ?22 ?18)|
----------------------------------------
```

Figure 6: A Graphical Trace of the Constraints Used in the Layout Process

8 Conclusions and Future Work

The work presented in this paper reflects the problems surrounding intelligent layout design of multimodal presentations as well as the current interdisciplinary investigations in the area of graphics design and psychology. LAYLAB, a testbed system for laying out multimodal documents containing text and graphics elements with which it is provided by mode-specific generators has been discussed in terms of the WIP architecture. We hypothesized that increased flexibility of constraints, the semantic expressiveness of constraint hierarchies, and the efficiency gained from using the domain concept in combination with forward checking aim at a powerful problem solver for synthesis tasks like automatic layout.

Most of our current research is concerned with knowledge representation. We have to maintain the graphical constraints knowledge base in order to augment the primitive and non-primitive design constraints. Furthermore, we focus on a universal constraint-based representation of document structures and a declarative representation of design basics.

As our approach only allows a rudimentary treatment of design compromises and alternatives, we have to regard this feature by extending the underlying constraint language to complexer structures.

Since WIP will be enriched by further dynamic modalities such as animation in the future, the layout manager has to cope with that problem too.

Frequently, it is easier for experts in a specific domain (e.g., graphics designers) to express their expertises such as positive or negative constraints by graphical examples. Therefore, the construction of an automatic acquisition component that infers design constraints from graphical sketches will be required in order to allow a maintenance of the knowledge-base by the experts themselves (see also [KF91]). In that case, the system could learn new layout patterns by interactively criticizing old ones through the user.

Acknowledgements

The research reported in this paper has been carried out in the WIP project which is supported by the German Ministry for Research and Technology under contract ITW 8901 8. This work profited from a cooperation between the DFKI and the Hochschule für Bildende Künste Saar. I would like to thank Wolfgang Wahlster, Steve Feiner, Elisabeth André and Thomas Rist for stimulating discussions about automating layout and valuable comments on this paper. The implementation has benefited from the contributions of Wolfgang Maaß (*CLAY*) and Dudung Soetopo (interaction handler, typography component).

References

[AFG+92] E. André, W. Finkler, W. Graf, T. Rist, A. Schauder, and W. Wahlster. WIP: The automatic synthesis of multimodal presentations. German Research Center for Artificial Intelligence, DFKI Research Report, January 1992.

[AR90] E. André and T. Rist. Synthesizing illustrated documents: A plan-based approach. In *Proceedings of InfoJapan '90, Vol. 2*, pages 163–170, Tokyo, 1990. Also DFKI Research Report RR-91-06.

[AR92] E. André and T. Rist. The design of illustrated documents as a planning task. German Research Center for Artificial Intelligence, DFKI Research Report, January 1992.

[Arn66] R. Arnheim, editor. *Towards a Psychology of Art.* University of California Press, Berkeley, 1966.

[BD86] A. Borning and R. Duisberg. Constraint-based tools for building user interfaces. *ACM Transactions on Graphics*, 5(4):345–374, October 1986.

[BDFB+87] A. Borning, R. Duisberg, B. Freeman-Benson, A. Kramer, and M. Woolf. Constraint hierarchies. In *Proceedings of OOPSLA '87*, pages 48–60, October 1987.

[Bea85] R. Beach. *Setting Tables and Illustrations with Style.* PhD thesis, Dept. of Computer Science, University of Waterloo, Ontario, 1985.

[BF81] A. Barr and E. Feigenbaum, editors. *The Handbook of Artificial Intelligence, Vol. 1.* Pitman, London, England, 1981.

[Bor81] A. Borning. The programming language aspects of ThingLab, a constraint-oriented simulation laboratory. *ACM Transactions on Programming Languages and Systems*, 3(4):353–387, October 1981.

[CSI86] E. Cohen, E. Smith, and L. Iverson. Constraint-based tiled windows. *IEEE Computer Graphics and Applications*, 6(5):35–45, 1986.

[DHS+88] M. Dincbas, P. Van Hentenryck, H. Simonis, A. Aggoun, T. Graf, and F. Berthier. The constraint logic programming language chip. In *Proceedings of the International Conference on Fifth Generation Computer Systems FGCS-88*, pages 693–702, Tokyo, Japan, December 1988.

[FBMB90] B. Freeman-Benson, J. Maloney, and A. Borning. An incremental constraint solver. *Communications of the ACM*, 33(1):54–63, 1990.

[Fei88] S. Feiner. A grid-based approach to automating display layout. In *Proceedings of the Graphics Interface '88*, pages 192–197. Morgan Kaufmann, Los Altos, CA, June 1988.

[Fri84] M. Friedell. Automatic synthesis of graphical object descriptions. *Computer Graphics*, 18(3):53–62, 1984.

[GM91] W. Graf and W. Maaß. Constraint-basierte Verarbeitung graphischen Wissens. In W. Brauer and D. Hernández, editors, *Proceedings 4. Internationaler GI-Kongreß Wissensbasierte Systeme - Verteilte KI und kooperatives Arbeiten*, pages 243–253. Springer-Verlag, Berlin, Germany, October 1991. Also DFKI Research Report RR-91-35.

[Gos83] J. Gosling. *Algebraic Constraints.* PhD thesis, Dept. of Computer Science, Carnegie Mellon University, 1983.

[Gra91] W. Graf. Constraint-based processing of design knowledge. In *Proceedings of the AAAI-91 Workshop on Intelligent Multimedia Interfaces*, Anaheim, CA, July 1991.

[HA91] E. Hovy and Y. Arens. Automatic generation of formatted text. In *Proceedings of the 9th National Conference of the American Association for Artificial Intelligence*, pages 92–97, Anaheim, CA, July 1991.

[Hen89] P. Van Hentenryck, editor. *Constraint Satisfaction in Logic Programming.* MIT Press, Cambridge, MA, 1989. Revision of Ph.D. thesis, University of Namur, 1987.

[HFS91] K. Harbusch, W. Finkler, and A. Schauder. Incremental syntax generation with tree adjoining grammars. In W. Brauer and D. Hernández, editors, *Proceedings 4. Internationaler GI-Kongreß Wissensbasierte Systeme - Verteilte KI und kooperatives Arbeiten*, pages 363–374. Springer-Verlag, Berlin, Germany, October 1991.

[JJL87] J. Jaffar and J.-L.Lassez. Constraint logic programming. In *Proceedings of the 14th ACM Symposium on Principles of Programming Languages*, pages 111–119, Munich, Germany, January 1987.

[KF91] D. Kurlander and S. Feiner. Inferring constraints from multiple snapshots. Technical report, Department of Computer Science, Columbia University, New York, NY, 1991.

[KM89] L. Koren and W. Meckler, editors. *Graphics Design Cookbook: Mix and Match Recipes for Faster, Better Layouts.* Chronicle Books, San Francisco, CA, 1989.

[KPKY91] G. Kramer, J. Pabon, W. Keirouz, and R. Young. Geometric constraint satisfaction problems. In *Working Notes AAAI Spring Symposium 'Constraint-Based Reasoning'*, pages 242–251, Stanford University, CA, March 1991.

[Lel88] W. Leler, editor. *Constraint Programming Languages: Their Specification and Generation.* Addison-Wesley, Reading, MA, 1988.

[Maa92] W. Maaß. Constraint-basierte Plazierung in multimodalen Dokumenten am Beispiel des Layout-Managers in WIP. Master's thesis, Dept. of Computer Science, University of Saarbrücken, January 1992.

[Mac77] A. Mackworth. Consistency in networks of relations. *Artificial Intelligence*, 8(1):99–118, 1977.

[Mac85] J.D. Mackinlay. *Automatic Design of Graphical Presentations.* PhD thesis, Dept. of Computer Science, Stanford University, Stanford, CA, 1985.

[Mac88] A. Mackworth. Constraint satisfaction. In S. Shapiro, editor, *Encyclopedia of Artificial Intelligence*, volume 1, pages 205–211. Wiley, Chichester, England, 1988.

[May92] M. Maybury, editor. *Intelligent Multimedia Interfaces.* AAAI Press, Menlo Park, CA, 1992. forthcoming.

[MB81] J. Müller-Brockmann, editor. *Grid Systems in Graphic Design.* Verlag Arthur Niggli, Niederteufen, Switzerland, 1981.

[MF90] S. Mittal and B. Falkenhainer. Dynamic constraint satisfaction problems. In *Proceedings of the 8th National Conference of the American Association for Artificial Intelligence*, pages 25–32, Boston, MA, July 1990.

[MSS91] W. Maaß, T. Schiffmann, and D. Soetopo. LAYLAB: Ein System zur automatischen Plazierung in multimodalen Dokumenten. Fortgeschrittenenpraktikum 'Wissensbasierte Graphikgenerierung', Dept. of Computer Science, University of Saarbrücken, 1991.

[MT88] W. Mann and S. Thompson. Rhetorical structure theory: Towards a functional theory of text organization. *TEXT*, 8(3), 1988.

[Mye91] B. Myers. Using AI techniques to create user interfaces by example. In Sullivan and Tyler [ST91], pages 385–402. Peridot.

[Nel85] G. Nelson. Juno, a constraint-based graphics system. *Proceedings of the SIGGRAPH '85*, 19(3):235–243, 1985.

[OSS92] A. Ortony, J. Slack, and O. Stock, editors. *Computational Theories of Communication and their Application.* Springer-Verlag, Berlin, Germany, 1992.

[PW85] T. Pavlidis and C. Van Wyk. An automatic beautifier for drawings and illustrations. *Computer Graphics*, 19(3):225–234, 1985.

[RA92] T. Rist and E. André. From presentation tasks to pictures including depictions of 3D objects: Towards an approach to automatic graphics design. German Research Center for Artificial Intelligence, DFKI Research Report, January 1992.

[Rüe89] R. Rüegg, editor. *Basic Typograhy: Design with Letters.* ABC-Verlag, Zürich, Switzerland, 1989.

[SF91] D. Seligmann and S. Feiner. Automated generation of intent-based 3d illustrations. *Computer Graphics*, 25(3), July 1991.

[Smo91] G. Smolka. Residuation and guarded rules for constraint logic programming. Technical report, German Research Center for Artificial Intelligence, Saarbrücken, Germany, March 1991.

[SS80] G.J. Sussman and G.L. Steele. Constraints – a language for expressing almost-hierachical descriptions. *Artificial Intelligence*, 14(1):1–39, 1980.

[ST91] J. Sullivan and S. Tyler, editors. *Intelligent User Interfaces.* Frontier Series. ACM Press, New York, NY, 1991.

[Sut63] I. Sutherland. Sktechpad: A man-machine graphical communication system. In *IFIPS Proceedings of the Spring Joint Computer Conference*, pages 329–345, 1963.

[vW82] C.J. van Wyk. A high-level language for specifying pictures. *ACM Transactions on Graphics*, 1(2):163–182, 1982.

[WAB+92] 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. [OSS92]. Also DFKI Research Report RR-91-08.

[WAGR91] 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*, pages 8–14. Springer-Verlag, Berlin, Germany, April 1991. Also DFKI Research Report RR-91-05.

[WH87] D. Willows and H. Houghton, editors. *The Psychology of Illustration, Vol. 1, 2.* Springer-Verlag, Berlin, Germany, 1987.

DFKI Documents

D-91-01
Werner Stein , Michael Sintek: Relfun/X - An Experimental Prolog Implementation of Relfun
48 pages

D-91-02
Jörg P. Müller: Design and Implementation of a Finite Domain Constraint Logic Programming System based on PROLOG with Coroutining
127 pages

D-91-03
Harold Boley, Klaus Elsbernd, Hans-Günther Hein, Thomas Krause: RFM Manual: Compiling RELFUN into the Relational/Functional Machine
43 pages

D-91-04
DFKI Wissenschaftlich-Technischer Jahresbericht 1990
93 Seiten

D-91-06
Gerd Kamp: Entwurf, vergleichende Beschreibung und Integration eines Arbeitsplanerstellungssystems für Drehteile
130 Seiten

D-91-07
Ansgar Bernardi, Christoph Klauck, Ralf Legleitner
TEC-REP: Repräsentation von Geometrie- und Technologieinformationen
70 Seiten

D-91-08
Thomas Krause: Globale Datenflußanalyse und horizontale Compilation der relational-funktionalen Sprache RELFUN
137 Seiten

D-91-09
David Powers, Lary Reeker (Eds.):
Proceedings MLNLO'91 - Machine Learning of Natural Language and Ontology
211 pages
Note: This document is available only for a nominal charge of 25 DM (or 15 US-$).

D-91-10
Donald R. Steiner, Jürgen Müller (Eds.):
MAAMAW'91: Pre-Proceedings of the 3rd European Workshop on „Modeling Autonomous Agents and Multi-Agent Worlds"
246 pages
Note: This document is available only for a nominal charge of 25 DM (or 15 US-$).

D-91-11
*Thilo C. Horstmann:*Distributed Truth Maintenance
61 pages

D-91-12
Bernd Bachmann:
$\mathrm{H}^{\mathrm{iera}}\mathrm{C}_{\mathrm{on}}$ - a Knowledge Representation System with Typed Hierarchies and Constraints
75 pages

D-91-13
International Workshop on Terminological Logics
Organizers: Bernhard Nebel, Christof Peltason, Kai von Luck
131 pages

D-91-14
Erich Achilles, Bernhard Hollunder, Armin Laux, Jörg-Peter Mohren: *KRIS* : *K*nowledge *R*epresentation and *I*nference *S*ystem
- Benutzerhandbuch -
28 Seiten

D-91-15
Harold Boley, Philipp Hanschke, Martin Harm, Knut Hinkelmann, Thomas Labisch, Manfred Meyer, Jörg Müller, Thomas Oltzen, Michael Sintek, Werner Stein, Frank Steinle:
μCAD2NC: A Declarative Lathe-Worplanning Model Transforming CAD-like Geometries into Abstract NC Programs
100 pages

D-91-16
Jörg Thoben, Franz Schmalhofer, Thomas Reinartz:
Wiederholungs-, Varianten- und Neuplanung bei der Fertigung rotationssymmetrischer Drehteile
134 Seiten

D-91-17
Andreas Becker:
Analyse der Planungsverfahren der KI im Hinblick auf ihre Eignung für die Abeitsplanung
86 Seiten

D-91-18
Thomas Reinartz: Definition von Problemklassen im Maschinenbau als eine Begriffsbildungsaufgabe
107 Seiten

D-91-19
Peter Wazinski: Objektlokalisation in graphischen Darstellungen
110 Seiten

Constraint-Based Graphical Layout of Multimodal Presentations

Winfried Graf

RR-92-15

Research Report