Relational Grammars for Interactive Design

Louis Weitzman
Visible Language Workshop
MIT Media Laboratory
20 Ames Street, E15-443
Cambridge, Massachusetts 02139-4307, USA
E-mail: weitzman@media-lab.mit.edu

Kent Wittenburg
Computer Graphics & Interactive Media Group
Bellcore
445 South Street, MRE 2A-347
Morristown, New Jersey 07962-1910, USA
E-mail: kentw@bellcore.com

1: Abstract

This paper describes an approach to interactive design tasks based on Relational Grammars. Relational Grammars extend traditional one-dimensional string languages to higher dimensions through user-supplied domain relations. Design support takes the form of graphic inferences on partial input during design interaction with the user. Because of the nature of the rule definitions, design elements can be roughly "sketched" without concern for their specific details. During the interaction process, the system installs constraints, adds default attributes, and/or builds higher-level composite structures out of the original input. An example scenario of the use of this formalism is presented supporting the task of document design. A general algorithm for the integration of grammar-based parsing in an interactive design environment is presented.

Keywords: Interface design tools and techniques, grammar-directed design, visual languages, relational grammars, parsing, graphic inferencing, constraints

2: Introduction

Computers have provided access to tools for doing tasks that have traditionally only been done by design professionals. We should not expect that users of these tools be designers or have the necessary design expertise. Therefore, as design moves from traditional mediums to the electronic studio, representation of design knowledge becomes crucial in order to support a dialog between designer and machine. This work applies a grammar formalism to the design domain with the ultimate goal of supporting the user with the general task of interactive computer-based design.

Recently, we have seen the emergence of computational systems that embody languages of design to perform these tasks automatically [4]. Languages to support design generation is only one possible use of these formalisms. Supporting the general design process includes many other tasks such as parsing of design input [15, 17], supporting designers with critiques [5], prediction of design decisions and the exploration of design alternatives [10]. The emphasis here is on interaction. As a designer builds up design alternatives, Relational Grammars can be used in various ways to support the process. Such a system can assist in the implementation of details during design exploration as well as suggest or enforce design requirements.

There are a number of requirements that interactive visual design domains place on conventional rule representation and parsing techniques. In graphic design tasks, there are many layout constraints related to the use of two-dimensional space, such as symmetry and alignment, that a rule language should be able to state. Traditional one-dimensional string languages are not adequate for this task and must be extended to higher dimensions. Also, the rule language must be expressive enough to allow arbitrary domain-specific constraints, not necessarily restricted to those that are just visual in nature. In traditional string parsing, the assumption that input is in a strict left to right sequence leads naturally to parsers that can, for instance, offer completion facilities in command interfaces. In visual domains, however, the question of ordering the input is a thorny one. Forcing the user to conform to some narrow temporal ordering requirements in any design task seems particularly misplaced. On the other hand, with no ordering restrictions, design could continue in a natural way, the user building up elements incrementally while individual design problems could be recognized and addressed (or postponed). Non-monotonic changes to the input should also be handled with minimal disruption so users can move, resize, and delete elements of their input. Our approach to these problems is to use Relational Grammars [23-25] for our rule language and the incremental, bottom-up parsing algorithm of [23,25] in our top-level interaction loop.

In the following sections, the general applicability of Relational Grammars for supporting the design process is discussed. First, a scenario of the use of Relational Grammars to support interactive layout design is presented followed by a generalized algorithm for incorporating this parsing paradigm into an interactive loop. The grammar used in the scenario is then described in detail. Related work in rule-based design systems is then presented. Finally, thoughts on problems this approach raises and future directions this research may take are described.

3: Page design scenario

This section presents an example interaction with the system in supporting the design of the layout of a page. The rules of this grammar may be inferred from the example page illustrated in Figure 1. The rules in this language capture this style and embody various layout articulation techniques, such as: graphic separators (or graphic rule bars) above chapter titles and section headings, relative font sizes, and the use of margins for images. This grammar will form the basis for this scenario and the following discussion.

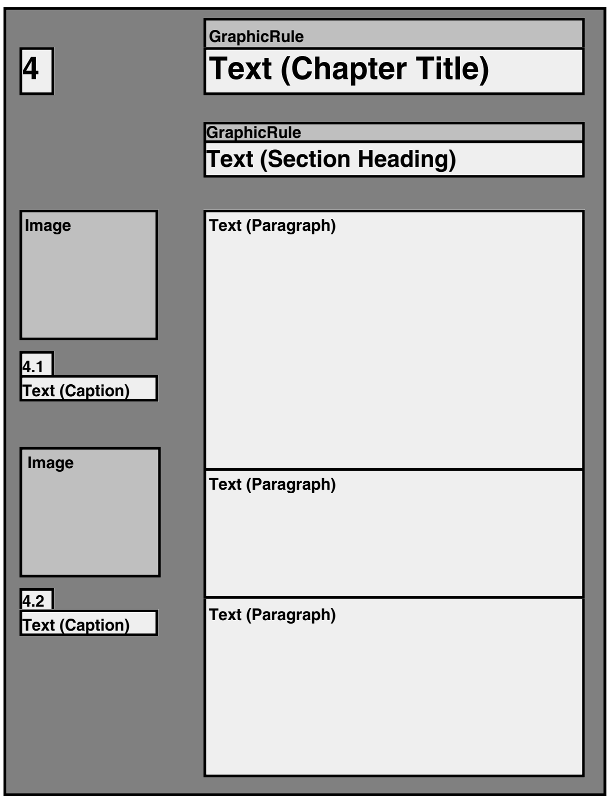

Figure 1. An example page layout based on the graphic design of [14] that is the model for the grammar described in this paper.

The interaction sequence begins by selecting a primitive lexical element from a menu and adding it to the layout. In this example, there are four basic categories of input which include: text, number, image and graphic rule. As design input proceeds, the system parses the input and automatically creates new composite structures. The creation of these composite structures is a form of graphic inferencing that installs constraints on the original input elements. The constraint system is based on [1] and is easily extended to provide new graphic constraints. The graphic attributes of the participating elements are constrained to act together as defined by the behavior of the new composite object. Typically, multiple graphical constraints are used to enforce the position and size relations between elements. Constraints may also make individual changes to elements (e.g., changing their color or font specification). Relationships can be defined so that the elements involved only roughly match the desired requirements. In this way, input can be loosely *sketched* and the application of the rules will *clean up* the input.

In this scenario, the user begins to create a `Figure` by adding a `Text` item and placing a `Number` roughly above and left aligned to it. This interaction automatically creates a `Caption` composite structure and in the process, the elements are left aligned exactly and placed immediately adjacent to one another. By continuing to add an `Image` element above the newly formed `Caption`, a `Figure` is then created and the `Image` is left aligned and placed above the `Number` element.

We have not made any assumptions as to the appropriate interaction paradigm and allow the user to select between multiple interaction styles. The user can request automatic rule selection and invocation, as described above, or the user can postpone rule firing using an interactive graphical agenda. The interactive agenda is motivated by the desire to bring the user back into the toplevel loop to deal with ambiguous derivations in a nonobtrusive manner. In the interactive mode, unfired actions will be accumulated on the agenda. The agenda is visualized as a set of buttons, each illustrating before and after states of the elements that participate in the pending inference. The user is free to ignore the suggestions offered by the system, or click on the most appropriate action. By selecting a button, the inference is performed and button is removed from the agenda. Figure 2 illustrates a sequence of three interactions using the agenda to build the same `Figure` composite described above. In Figure 2A, the user has created and placed three items, an image, a number and a text item and the agenda (on the right side of the figure) reflects one pending action, to create a `Caption`. After clicking on this "Make Caption" button, Figure 2B shows the resulting `Caption` element and a new agenda item to create a `Figure` composite element. Figure 2C shows the final `Figure` element with all constraints (position, size, and font specification) applied.

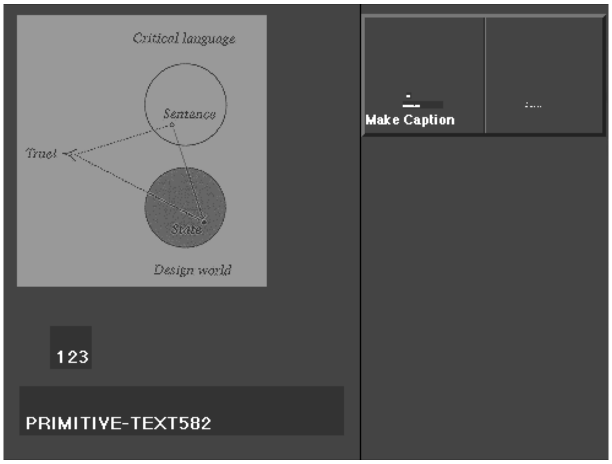

Figure 2A.

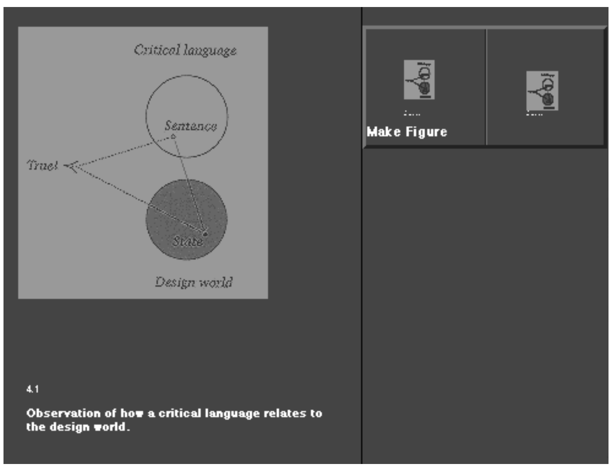

Figure 2B.

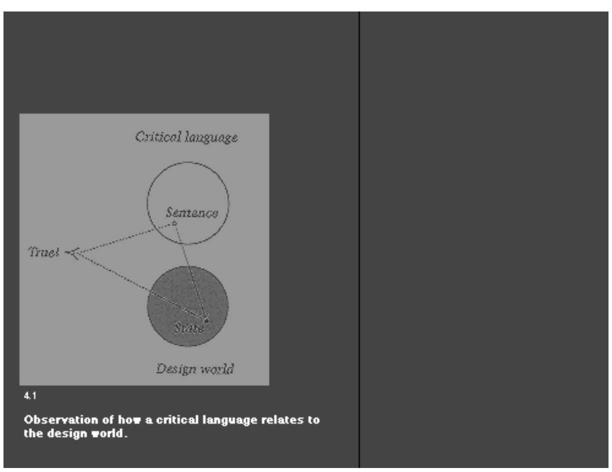

Figure 2C.

Figure 2. Sequence of interactions with the agenda depicting the process of creating a `Figure` composite element.

As described above, an interactive design environment must also support nonmonotonic changes to the input. The user may move, resize, or delete any lexical input. This input may also be a component of a composite element. The parse table reflects these changes and pending derivations that are based on the deleted element are removed from the agenda.

4: The top-level loop

The significant requirement for our top-level interaction loop is to maintain a working set of valid system actions discovered through parsing. There are actually three data structures which need to be maintained: the current input state (simple or composite objects that the user or system has created along with a data structure to maintain the object relationships); the parse table (the set of partially and completely matched rules indexed by input elements); and the system action agenda (actions stemming from the derivations admitted by the grammar that are headed by one of the root symbols of the grammar and that terminate in a subset of the input).

Successful derivations can produce one or more system actions. Typically, such actions will involve building a composite object--for example, a `Figure` or `Heading`. The individual elements of the composite are typically "grouped," precisely aligned, and defaults such as font styles added. The introduction of a composite object in place of its parts entails the following bookkeeping with respect to the three data structures mentioned previously. First, the input elements making up the composite must be deleted from the current input set; second, any complete or partially matched rules in the parse table that dominate these now deleted input elements must be removed from the parse table; third, any derivations with their associated system actions that depend on these input elements must be removed from the action agenda as well. The effect of these operations may be to invalidate previously valid actions on the agenda. For example, if two different derivations each use some of the same input elements but lead to different composites, the selection of one of the actions to produce one composite will eliminate the possibility of ever choosing the competing alternative. Finally, the new composite object is added to the current input set so that it now can be considered as input to further rule matches, derivations, and actions. Rules are advanced and the system agenda is updated. Currently, there are two modes of interaction with the agenda, automatically firing the first item on the agenda or interactively selecting an item from a graphic representation of the agenda. Sorting and thresholding of the pending inferences can provide additional support dealing with the agenda. If no action is selected, control is returned to the user who may once again add, move, delete, or resize objects. This is all summarized in Figure 3.

The bottom-up, incremental parsing algorithm of [23, 25] is used here with only one change. Derivations are considered successful even if they do not cover all the input elements. This change has no effect on the internals of the parsing algorithm.[1] It is typical for grammars used in one of these design applications to have many root symbols. The grammar for page layout that follows in section 5 uses root symbols such as `Figure`, `Heading`, `Body`, etc., an indication that these are the goal categories that the parser will be seeking. If a larger grain size of interaction is desired, the grammar would use fewer root categories, giving the user the option to build a whole title page in one fell swoop rather than each of its structural parts in turn.

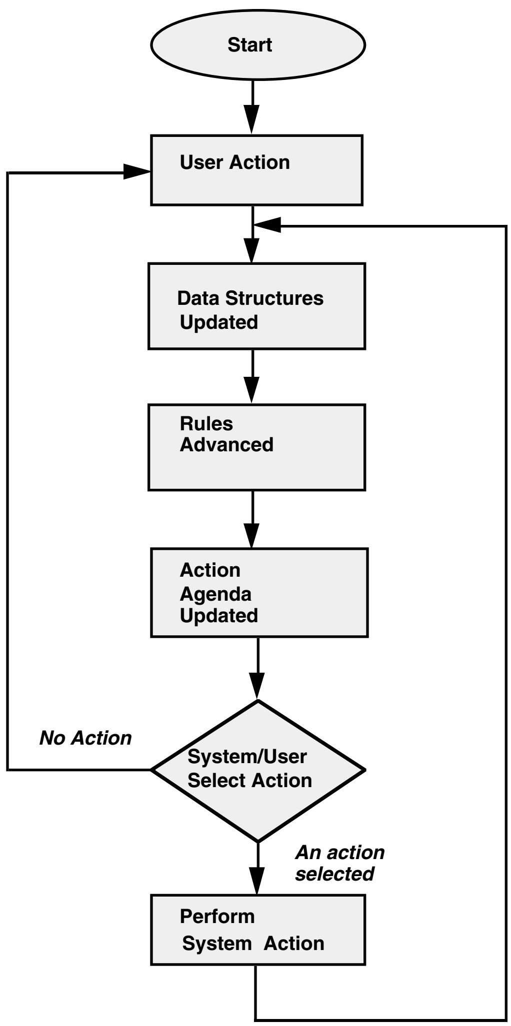

Figure 3. Flowchart of top level algorithm.

[1] Note, however, that such a change would affect predictive parsing algorithms such as [21].

5: A Relational Grammar for layout design

The basic components of the relational grammar utilized in the scenario in section 3 are described below. They include the lexical and compositional categories of the language, the relations between the design elements, and the rule specifications described in terms of these categories and relations. The supporting data structures are also described.

Lexical and compositional categories
The first step in defining a relational language is to identify the primitives of the language and the corresponding lexical mappings for those primitives in the grammar. In an interactive environment, the primitive lexical categories are available to the user from an input menu. They include:

Text	Text element for titles, headings, captions, and paragraphs.
Number	Number element for figures and chapter numbers.
GraphicRule	Graphic element for separating regions.
Image	Bitmap image.

As the user selects a primitive category from an interface menu, an instance of the class is created. The user sizes and locates the newly created instances. These instances maintain local state information, such as size, position, color, and font specification.

Primitive elements combine to form composite structures which may in turn be used to define other new structures. The exact details of the combination of these elements are encoded in the grammar rules and are described below. Composite elements are also treated as instances of classes. Composite icons maintain local information and information of their component elements. Some information, such as size and position, are calculated values based on these component elements.

The compositions for this language are:

TitlePage	= Title	+ Body	+ FigureCol
Title	= GraphicRule	+ Text	+ Number
Body	= Heading	+ TextBody	
Heading	= GraphicRule	+ Text	
Textbody	= Text	+ Text	
Textbody	= Textbody	+ Text	
FigureColumn	= Figure	+ Figure	
Figure	= Image	+ Caption	
Caption	= Text	+ Number	

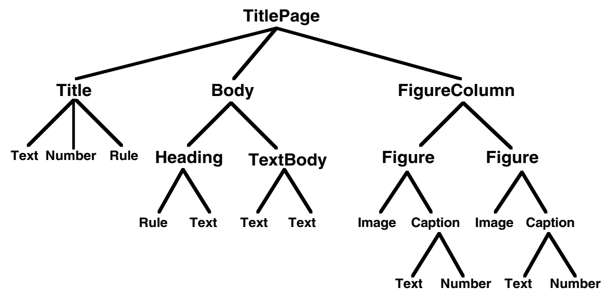

Figure 4. Structural hierarchy of a `TitlePage` corresponding to a derivation in the grammar. A graphic depiction is shown in Figure 1.

A hierarchy of objects representing a `TitlePage` is shown in Figure 4. The primitive lexical items form the leaves of the hierarchical tree. Each subtree of depth 1 corresponds to a rule application. From the bottom-up point of view, a rule application forms a composite element whose parts are its immediate daughters and whose type corresponds to the left-hand-side category of the rule.

What Figure 4 does not show is that there are additional relational requirements necessary for any rule to fire and subsequently introduce a composite element. For example, a `Caption` element will not combine with just any `Image` element to produce a `Figure`--an additional requirement is that the Caption be immediately below the `Figure`. The relational constraints are the key to this parsing algorithm and the requirements for rule firing.

Domain relations and predicates

Domain relations determine rule applicability within the language. The relations generally may have any arity but binary relations play a special role in parsing (see 20, 22). For the domain of graphic design, relations are typically spatial, such as *Above* and *Equal-Width*. Other applications, such as table of contents or graph layout, may describe information more semantic in nature such as *Author-Of* or *Connected-To*, respectively. The standard graphic relations supported in this domain of layout include:

Above	*Below*
Left-Of	*Right-Of*
Top-Aligned	*Bottom-Aligned*
Left-Aligned	*Right-Aligned*
Equal-Width	*Equal-Height*

where directional relations such as *Above* have the additional requirement of adjacency. All relations need to be able to be called as predicates. For the parser to operate, domain and range queries also need to be supported for some relations. A domain query answers questions such as "What is above X?" while a range query answers questions such as, "What is X above?" These queries for the *Above* relation are summarized below.

Category	Example Query
Binary Predicate	"Is X above Y?" `(query 'above X Y)`
Domain Query	"What is above X?" `(query 'above :? X)`
Range Query	"What is X above?" `(query 'above X :?)`

A domain dependent object store is used to answer relational queries and predicates. These object stores differ depending on the application. In the 2D layout domain a KD-tree representation is used in order to optimize queries about spatial proximity. A KD-tree is a representation for storing objects based on K-dimensional attributes [19]. In particular, they are useful for storing objects with rectangular bounding boxes, where each edge of the bounding box is a separate index into the tree, i.e. K=4. Subsequent queries are based on these rectangular regions. Elements are stored as nodes in a binary tree, using the bounding rectangle as keys for deciding whether an object should be stored on one branch or another of any given node. Once the icons are created within the interface, they are added to the KD-tree. Queries of the form "What is above X?" and "Is X above Y?" can then quickly be answered based on their rectangular bounding boxes.

Grammar rules

Grammar rules specify the conditions under which composite elements can be created. During the process of interpretation, rules support the designer by producing constraints on the input elements. Rules use the domain relations to create preconditions, that when satisfied, result in the rule firing and creation of composite objects. An example rule definition for creating the composite element `Figure`, is illustrated below.

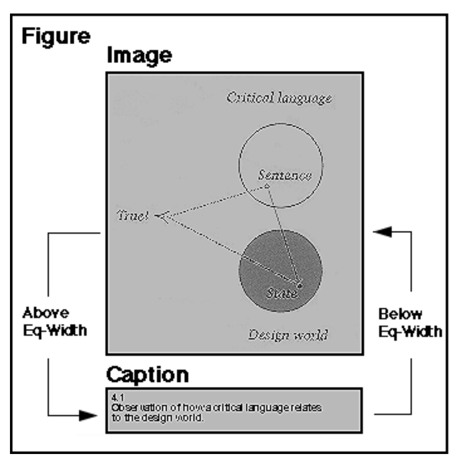

Figure 5A. The graphic illustration of the Make-Figure rule (adapted from [14]).

```
(Defrule (Make-Figure The-Grammar)
   (0 Figure)
   (1 Image)
   (2 Caption (Above    1 2)
              (Eq-Width 1 2))
   :Out
   (Make-Instance 'Figure-Element
      :Image-Element   1
      :Caption-Element 2))
```

Figure 5B. The definition of the Make-Figure rule.

The rules are to be read as follows. Rule elements are numbered 0 to n. Element 0 is considered to be the left-hand-side (composite) category type. Elements 1 to n are the right-hand-side. For the `Make-Figure` rule to be applicable, it must first match an `Image` element. Next it will seek a `Caption` element by trying to satisfy the query *(Above <image-element> :?)*. If these conditions are satisfied, then the predicate *(Eq-width <image-element> <caption-element>)* must return `TRUE`. If all these preconditions are satisfied, then a potential composite category `Figure` is introduced into the parse table. Assuming `Figure` is a root category of the grammar, an action to go ahead and create this composite is made available for the system action agenda. In addition, a rule may contain semantic output forms to be evaluated as part of the rule action. These forms follow the symbol `:Out` in a rule body. In the case of the Make-Figure rule, the `Make-Instance` form creates a Figure-Element when the agenda item is invoked. This object references the input elements which satisfied this rule's preconditions. As part of running the agenda item, additional constraints are then installed on these input elements. These output constraints are different from the rule's preconditions and are the reason input can be roughly *sketched*, satisfying the preconditions, while the output semantics may require more restrictive constraints.

6: Related work

The work described in this paper follows in the tradition of rule-based systems to support the task of design. One characterization of these systems according to [11] is the level of interaction they afford the user. This ranges from completely manual, to constraint-based, to critic-based, to improver-based, and finally, to fully automatic. On this scale of automaticity, our work would be characterized as an *improver*-based system.

Some constraint-based systems make an effort to empower the user by giving them direct feedback and control of the inferencing process. Rockit [8], a system to apply graphic constraints in building an interface, provides not only visual but audio feedback to help the user in deciding the correct constraint application. Hudson and Yeatts [6] explore a similar approach. In both of these systems, the interaction and feedback is direct and immediate, i.e., it occurs on the drawing area. Our apporach also provides control over the inference process, but the interactive agenda allows design to advance without any immediate interruption.

Continuing with this characterization of design systems, critique-based systems [5] provide a commentary on the status of the design without providing the ability to change the design, while improver-based systems may not provide a critique but allow for the modification of an existing design. An example of an early improver-based system is Pavlidis and Van Wyk's *beautifier* system [17] for improving network diagrams. It first infers the relations between graphical objects in network diagrams and then modifies those objects based on the constraints of the domain. No effort was made to involve the user with information or control over the inferencing process. Our agenda mechanism can act as a critiquer, but also provides the ability to modify the design itself.

At the fully automatic end of the scale of interaction, there are many different techniques to create designs without intervention. ANDD [13] is a system that does fully automatic design of network diagrams. It is an effective rule-based system that designs and articulates network diagrams to communicate information represented in arbitrary attributed graphs and is part of a multimedia explanation facility. Two other related rule-based paradigms used for the automatic generation of graphic form include expert systems and shape grammars. Predikt applies expert system technology for the preliminary synthesis and critique of kitchen designs [16]. The system is able to convert between semantic and graphic representations to articulate the designs. Shape grammars [20] are interesting in that they use the language of the designer, i.e. shapes, as the building blocks of domain rules. There have been shape grammars developed to explore a set of conventions [9] and others to extend an existing corpus of work such as, Palladian villas [21], Mughul gardens [22], and Wright prairie homes [12].

We take the generation problem to be characterized as follows. Given a semantic representation of the data to be communicated through some design, create one or more instances of a fully articulated design. This may, in fact, be done through parsing and we are currently using this *generation as parsing* paradigm to automatically create designs of dynamic documents.

Interactive design systems that use Relational Grammars in more structured design tasks have been created at Bellcore [2]. One system uses a top-down algorithm (instead of bottom-up) to support engineering design tasks in telephony networks. The artifact being designed is represented as a network diagram where some of the nodes correspond to grammar nonterminals, representing an abstraction of parts of the design not yet filled in, and others correspond to terminals, representing completed decisions. By clicking on nonterminals, the user can explore choice points in the design represented in the grammar as alternative expansions of the nonterminal. The process is complete when all elements in the design have been replaced by terminal elements in the grammar. The top-down approach is well suited to this more structured domain, enforcing conformance to known design configurations. The bottom-up approach we have adopted, however, is well suited to the more unpredictable task of exploratory design.

We believe that our main contribution is to suggest an innovative interaction paradigm that incorporates an independently justified grammar formalism and parsing algorithm. Much of the work mentioned above that is related to design does not fully articulate exactly how grammar rules are used in algorithms that are not exactly parsing but in some sense related. Here we hope to have made this point clear: our proposal is to use a "standard" bottom-up parsing algorithm with one change--namely, that derivations need not cover all the input. We allow user interaction into the process by declaring the parser's goals to be finer-grained than is typical in most standard parsing applications. Outside the parsing loop, we suggest methods for maintaining nonmonotonic editing changes that still allow the parse table to act as a cache for rule matches that are unaffected by user actions.

7: Future directions

There are a number of issues this system raises in attempting to support the interactive design process. Among these include better integration of grammars to support generation as well as interpretation, maintenance of the design space and searching through the set of possible alternatives, the use of multiple grammars to support the creation of different layout styles, the construction of grammars by demonstration, and the extension of this paradigm to include other media types and manage their design in the temporal, as well as spatial, domain.

Extensions of this work will include the creation of a more integrated environment where interpretation and generation are part of the same design process. [10] is cooperative CAD system that demonstrates the integration of parsing and generation in the iterative cycle of design. The user expresses initial design decisions and requirements. From this input, the system generates partial design solutions. The user can then select the most promising designs and interactively elaborate them, after which, the system continues automatic refinement. Techniques to support browsing and filtering of design alternatives helps the user focus on finding a desired solution.

The ability to search the design space and maintain multiple design alternatives is essential for any successful design system. A more interesting and rich design environment would track design decisions and provide support for backtracking of decisions found to be unproductive. The visual agenda described earlier begins to address this issue. The support of ambiguity in early design stages has parallels in natural language [7] and is an important research direction.

Multiple grammars will be used to investigate the applicability of this approach to support multiple users and their goals. For example, in the creation of interactive training manuals, separate grammars may characterize the different users and their tasks that the manual supports. These goals may range from general overviews to detailed repair procedures. They may include the purpose, installation, use, maintenance, calibration, troubleshooting, and repair of a piece of equipment. By utilizing different grammars for these different tasks, the system can present the information relevant to the user and situation.

Another area for investigation is the construction of grammars and rules *by-example*. There is a growing body of work [3] that is concerned with the construction of artifacts by example. It would be very interesting to see how grammars could support the specification of new rules interactively. In this way the designer can immediately affect the design environment. One approach would use a 2-tiered grammar. The first grammar would be based on graphic design principles and would parse the designer's actions resulting in new rules that could then be incorporated in an application-based grammar. This would help to create a design environment in which nonprogrammers can modify the existing rule sets.

Finally, we are interested in the application of this paradigm to the domain of multimedia and the presentation of information over time. [18] highlights a number of issues of sending interactive documents over a network of limited bandwidth. In addition to the problems of temporal layout, there are also dynamic factors based on network traffic as well as the capabilities of the hardware receiving the document. An approach such as the one described here might be used in authoring systems so as to anticipate the range of variables that might have to be accommodated in the actual presentation. It is easy to see that with the much greated complexity involved in laying out multimedia documents, having more system support for design decisions will be that much more critical.

8: Acknowledgements

We would like to thank both the Media Lab's Visible Language Workshop, directed by Professor Muriel Cooper, and Bellcore's Computer Graphics & Interactive Media Group, directed by Jim Hollan, for their support in this research. We would especially like to thank Professor Bill Mitchell, Ron MacNeil, Mark Rosenstein and Anil Chakravarthy who provided useful comments in the preparation of earlier versions of this paper. In addition, the first author's work is sponsored in part by the Alenia Corporation.

9: References

[1] Abelson, H., Sussman, G., Sussman, J. *Structure and Interpretation of Computer Programs*, MIT Press, Cambridge, Ma., 1985.

[2] Collier, G. and Karlin, P., 1993. Grammar of Design: Applying Grammatical Techniques to Phone Engineering, IEEE International Conference on Communications '93, Geneva, Switzerland.

[3] Cypher, A., (ed), Watch What I Do: Programming by Demonstration, MIT Press, Cambridge, Ma., 1993 (in press).

[4] Feiner, S., Mackinlay, J., Marks, J., Automating the Design of Effective Graphics for Intelligent User Interfaces, Tutorial Notes 30, CHI '92 Monterey, Ca., May 3-7, 1992.

[5] Fischer, G., Lemke, A, Mastaglio, T., Morch, A., Critics: An Emerging Approach to Knowledge-Based Human Computer Interaction, University of Colorado, Bolder, Colorado.

[6] Hudson, S. and Yeatts, A., 1991. Smoothly Integrating Rule-Based Techniques into a Direct Manipulation Interface Builder, Symposium on User Interface Software and Technology (UIST'91), November 11-13, Hilton Head, South Carolina.

[7] Gazdar, G. Mellish, C., *Natural Language Processing in Lisp*, Addison-Wesley, 1989.

[8]. Karsenty, S., J. Landay, and C. Weikart. *Inferring Graphical Constraints with Rockit*. in *Human Computer Interaction: Proceedings of HCI'92*. 1992. University of York, U.K.:

[9] Knight, T. 1989. Color grammars: designing with lines and colors, Environment and Planning B, 16:417-449.

[10] Kochhar, S. and Friedell, M. 1990. User control in cooperative computer-aided design, SIGGRAPH Symposium on User Interface Software and Technology (UIST'90), October 3-5, Snowbird, Utah.

[11] Kochar, S. Marks, J., and Friedell, M., Interaction Paradigms for Human-Computer Cooperation in Graphical-Object Modeling, in Proceedings of Graphics Interface '91, Calgary, Canada, June 1991, pp180-191.

[12] Koning, H., and Eizenberg, J., 1981. The Language of the Prairie: Frank Lloyd Wright's Prairie Houses, Environment and Planning B, 8:295-323.

[13] Marks, J., 1990. A syntax and semantics for network diagrams, Proceedings of the IEEE Workshop on Visual Languages, pages 104-110, October, Skokie, Illinois.

[14] Mitchell, W. *The Logic of Architecture*, MIT Press, 1990.

[15] Myers, B., Buxton, W., Creating Highly-Interactive and Graphical User Interfaces by Demonstration, Siggraph Proceedings August 18-22,1986 20:4 pp249-258.

[16] Oxman, R. and Gero, J. 1987. Using an expert system for design diagnosis and design synthesis, Expert Systems, February, vol. 4, no. 1, pp4-15.

[17] Pavlidis, T. and Van Wyk, C. An automatic beautifier for drawings and illustrations, Computer Graphics (Proceedings of Siggraph '85), 19(3), July, 1985.

[18] Rosenberg, J., Kraut, R., Gomez, L., Buzzard, C., Multimedia Communications for Users, IEEE Communications Magazine, May 1992.

[19] Rosenberg, J. Geographical Data Structures, IEEE Transactions on CAD, Vol. CAD-4, No. 1, January 1988.

[20] Stiny, G., 1980. Introduction to Shape and Shape Grammars, Environment and Planning B, 7:343-351.

[21] Stiny, G., and Mitchell, W., 1978. The Palladian grammar, Environment and Planning B, 5:5-18.

[22] Stiny, G., and Mitchell, W., 1980. The grammar of paradise: on the generation of Mughul gardens, Environment and Planning B, 7:209-226.

[23] Wittenburg, K., Weitzman, L. and Talley, J. Unification-Based Grammars and Tabular Parsing for Graphical Languages, in Journal of Visual Languages and Computing (1991) 2, 347-370.

[24] Wittenburg, K. Earley-style Parsing for Relational Grammars, Proceedings of IEEE Workshop on Visual Languages, Sept. 15-18, 1992, Seattle, Wa.

[25] Wittenburg, K. 1992. The Relational Language System. Bellcore Technical Memorandum TM-ARH-022353.