VISUAL GRAMMARS FOR VISUAL LANGUAGES

Fred Lakin

Center for the Study of Language and Information, Stanford University
Center for Design Research, Stanford University
Rehabilitation R&D Center, Palo Alto Veterans Hospital
3801 Miranda Ave, Palo Alto, California 94304
ARPAnet: lakin@csli.stanford.edu

ABSTRACT In modern user interfaces, graphics play an important role in the communication between human and computer. When a person employs text and graphic objects in communication, those objects have meaning under a system of interpretation, or "visual language." *Formal visual languages* are ones which have been explicitly designed to be syntactically and semantically unambiguous. The research described in this paper aims at spatially parsing expressions in formal visual languages to recover their underlying syntactic structure. Such "spatial parsing" allows a general purpose graphics editor to be used as a visual language interface, giving the user the freedom to *first* simply create some text and graphics, and *later* have the system process those objects under a particular system of interpretation. The task of spatial parsing can be simplified for the interface designer/implementer through the use of *visual grammars*. For each of the four formal visual languages described in this paper, there is a specifiable set of spatial arrangements of elements for well-formed visual expressions in that language. Visual Grammar Notation is a way to describe those sets of spatial arrangements; the context-free grammars expressed in this notation are not only visual, but also machine-readable, and are used directly to guide the parsing.

I. VISUAL LANGUAGES IN HUMAN/COMPUTER INTERACTION

When a person employs a text and graphic object in communication, that object has meaning under a system of interpretation, or "visual language." Visual languages can be used to communicate with computers, and are becoming an important kind of human/computer interaction. Phrases in a formal visual language can be used to direct searches in a data base [Odesta85]; construct simulations [Budge82]; provide communication for aphasics [Steele85]; or serve as expressions in a general purpose programming language [Sutherland65, Christianson69, Futrelle78, Lakin80c, Robinett81, Tanimoto82, Lanier84, Kim84, Glinert84].

Using a general purpose graphics editor as a visual language interface offers flexibility, but necessitates computer processing of visual languages. This paper describes the use of visual grammars in parsing phrases from visual languages. Both the visual grammars and the phrases were constructed in the **vmacs**^{TM} graphics editor for the **PAM** graphics system. The grammars are machine-readable and are employed directly by the parser; examples of grammars and parsing for four different visual languages will be given.

II. DRAWBACKS OF SPECIAL PURPOSE VISUAL LANGUAGE INTERFACES

A visual language interface should provide the user with two capabilities: the agility to create and modify phrases in the visual language, and the processing power to interpret the phrase and take appropriate action. All of the interfaces currently available (to the author's knowledge) which allow creation and processing of visual objects employ some kind of special purpose editor which is syntax-driven. Such editors achieve graphical agility and interpretative power, but at the expense of generality. In lieu of understanding, these editors substitute restriction. From a practical point of view, they limit the user's freedom: he can't spontaneously arrange text and graphics in new ways, or add a piece of text to an object already defined as graphical, or edit the text in a pull-down menu, or create a new kind of diagram. From a theoretical point of view, such editors never deal with the general issues of understanding diagrams: the meaning has been built into the structures and procedures of the predefined object categories[1].

III. ADVANTAGES OF PARSING VISUAL LANGUAGES IN A GENERAL PURPOSE GRAPHICS EDITOR

A general purpose editor could be used to construct visual language phrases, giving the user more graphic freedom. But of course the deficiency of general purpose graphics editors is that although we can draw anything we want, there is no specialized help for drawing special purpose things (by definition). Added to this is the fact that when we're finished we can't *do* anything with the drawing.

Spatial parsing offers a way to cure these deficiencies and obtain special purpose utility from a general purpose graphics editor. Spatial parsing recovers underlying syntactic structure so that a spatial arrangement of visual objects can be interpreted *as* a phrase in a particular visual language. Interpretation consists of parsing and then semantic processing so that appropriate action can be taken in response to the visual phrase. Appropriate action may include: assistance for agile manual manipulation of objects, compilation into an internal form representing the semantics, translation into another text-graphic language, or simply execution as an instruction to the computer.

Previous work [Lakin86a] has shown that recovering the underlying structure of the elements in the phrase is the more difficult part of the problem. Once a parse tree has been constructed, then semantic processing — at least for the *formal* visual languages considered in this paper — is relatively straightforward. Through spatial parsing the system can do semantic processing of visual phrases, and thus the user can have the advantages of employing a general purpose graphics editor as a visual language interface. The user simply creates some text and graphics, and

[1] A parallel can be drawn between special purpose, syntax-driven graphics editors and menu-driven so-called 'natural language' interfaces to data bases. The latter interfaces allow the user to construct a natural language query through choosing from a series of menus containing predefined natural language fragments. As each succeeding fragment is selected, it can be put immediately into its proper place in the final query because the offering of subsequent menus is guided by the logical form of a query. Parsing (and understanding) has been finessed. Compare this approach to a general purpose natural language front end such as LUNAR [Woods74] or TEAM [Martin83]. These "English understanding" systems are much more complex, but allow the user to type in query sentences that he or she makes up. The menu-driven approach has short-term practical advantages: it will run faster on cheaper computers; malformed sentences are not permitted so they don't have to be handled. On the other hand, the comprehensive approach used by the LUNAR and TEAM projects has long-term advantages: it gives the user freedom and tries to handle 'the sentence

later has the system process those objects as a visual expression under a particular system of interpretation. This differs from syntax-driven editors which force the user to select the semantic type of objects before they are created. By choosing the system of interpretation and when it is to be applied, the user gets flexibility. This is like the freedom in the separate but tightly coupled interaction between text editing and interpretation/compilation in **emacs**-based LISP programming environments [Stallman81]. Text is typed into the editor at time1, and then at time2 it is selected for interpretation/compilation "as" a LISP expression. Such freedom in a graphics editor allows different visual languages to be used in the same image, blackboard-style.

IV. SPATIAL PARSING FOR VISUAL LANGUAGES

The previous section introduced the notion of spatial parsing for visual languages and explained some advantages. This section will discuss such parsing in more detail as a user interface technique. The following section then presents visual grammars as a way to accomplish spatial parsing.

A. Definitions The purpose of spatial parsing is to aid in the processing of visual languages. As an operational definition of visual language, we say: *A visual language is a set of spatial arrangements of text-graphic symbols with a semantic interpretation that is used in carrying out communicative actions in the world*[2]. Spatial parsing deals with the spatial arrangement of the text-graphic symbols in a visual phrase from a visual language: *Spatial parsing is the process of recovering the underlying syntactic structure of a visual communication object from its spatial arrangement*[3].

B. Examples of Visual Languages Examples of communication objects (or visual phrases) from five different visual languages are shown in Figure 1 (images were constructed in the **vmacs** graphics editor, Section VIII). The first four communication objects are from formal visual languages; the fifth object is a piece of informal conversational graphics to remind us of the theoretical context of this work. An expression in a simple **Bar chart** language is in the upper left corner of Figure 1. **Feature Structures** are a notation employing brackets of differing sizes

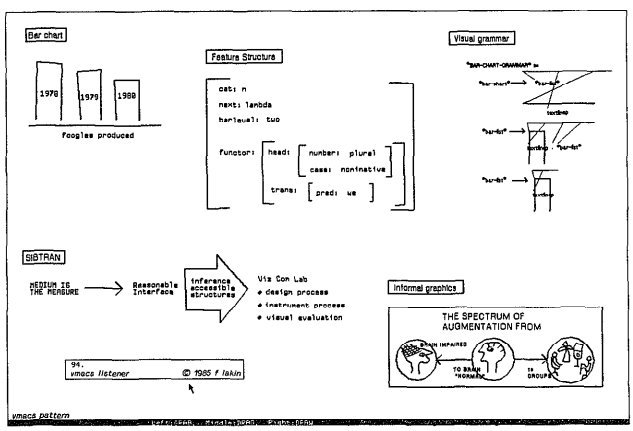

Figure 1. Visual communication objects from 5 different systems.

he was thinking of' as opposed to forcing construction from predefined pieces; it can handle arbitrary embedding of phrases; and insofar as the projects are successful, general principles about computer understanding of natural language will be discovered.

[2] Note that if we substitute "strings of textual symbols" for "spatial arrangements of text-graphic symbols" we have something strikingly similar to a characterization of written natural language. Interestingly, the text-graphic definition includes the textual one. A paragraph of text is one kind of arrangement of text-graphic symbols.

[3] Again the definition is parallel to one for textual parsing: textual parsing is the (rule-governed) process of recovering the underlying syntactic structure from the linear form of a sentence.

to encode information for a natural (textual) language expression. The **Visual Grammar** Notation uses text and graphics to represent a context-free grammar for a visual language. **SIBTRAN** is a fixed set of graphic devices which organize textual sentence fragments and provide additional semantic information. And the title block in the lower right corner is from an **Informal graphic** conversation discussed in Section VII.

C. Parsing and Interpretation of Visual Phrases in the User Interface The text-graphic objects representing parses for four of the visual communication objects are shown in Figure 2 (the 'spiderwebs' are a concise way of diagramming tree structure recovered by parsing; the more traditional notation will be shown later). The four parses were all performed in the same image space by the same function. `parse-appropriately` tries `spatial-parse` with each grammar from a list until the parse succeeds (if no parse succeeds, then "unrecognized visual expression" is signaled). The claim is that `parse-appropriately` represents progress is building user interfaces for formal visual languages: we now have one general purpose function with a list of 'known' formal visual languages for which it can handle visual expressions. The elements for the expressions were created in a general purpose graphics editor, and then the spatial syntactic context of the elements was utilized in parsing them. After parsing a visual expression, the structure thus recovered is then used directly by a specialized semantic processor in taking appropriate communicative actions. Interpretations based on the parses in Figure 2 are shown in Figure 3. Again, the four interpretations were all performed in the same image space by a single interpretation function which calls `parse-appropriately` and selects the proper semantic processor based on the result of the parse.

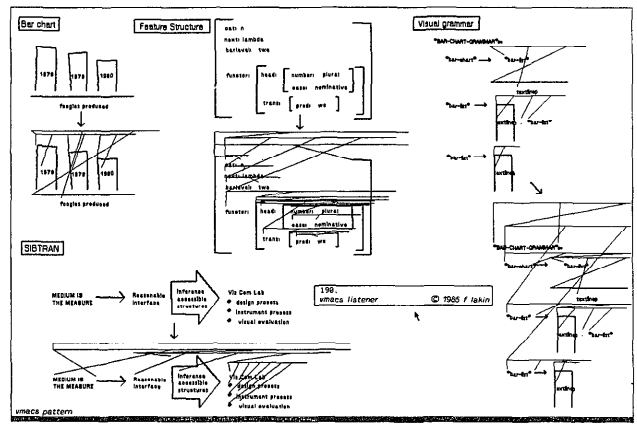

Figure 2. Appropriate parses for different visual communication objects residing in the same image.

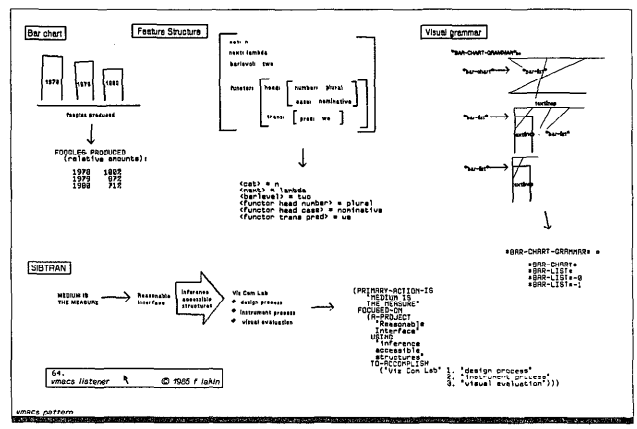

Figure 3. Appropriate interpretations for different visual communication objects residing in the same image.

D. Spatial Parsing Versus Image Processing Note that the visual communication objects under discussion are fundamentally *symbolic* objects, although the symbols are text-graphic. In spite of the fact that they are referred to as 'images,' they are very different from the raster images produced by a TV camera scanning a diagram on a blackboard[4]. The entire lower level of interpretation called recognition has been finessed when a human constructs a visual phrase within the **vmacs** graphics editor. Figure 4 presents the taxonomy of visual objects available to the **vmacs** user. Text-graphic objects are either lines or patterns: lines are visual atoms (single drawlines or textlines); and patterns are groups of text-graphic objects. Since the user employs **vmacs** to construct phrases, they are input as drawn lines and pieces of text (i.e. *atomic* text-graphic symbols)[5] in spatial juxaposition. Thus the focus of the research is on the recognition/parsing of the *spatial arrangment* of atomic visual elements serving as terminals in visual phrases. This is roughly analogous to language understanding work which begins with a typed-in sentence as input rather than starting with an aural image.

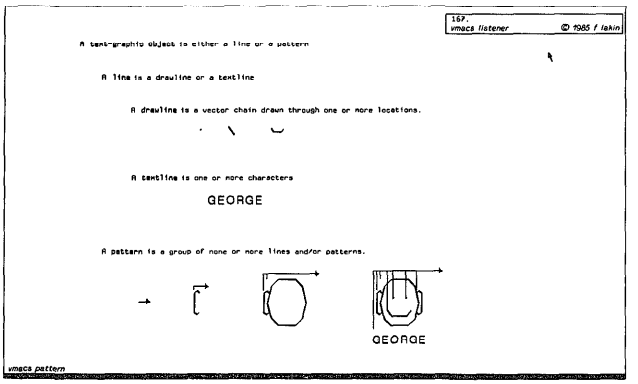

Figure 4. Taxonomy for graphic objects available in the **vmacs** graphics editor.

V. VISUAL GRAMMAR-DIRECTED PARSING OF VISUAL LANGUAGES

The problem with procedurally directed parsing is that knowledge about the syntax of each language is embedded implicitly in procedures, making it hard to understand and modify. In the current work, a spatial parser has been written that utilizes context-free grammars which are both visual and machine-readable. The parser takes two inputs: a region of image space and a visual grammar. The parser employs the grammar in recovering the structure for the elements of the graphic communication object lying within the region. One advantage of a visual grammar is that it makes the syntactic features of the visual language it portrays explicit and obvious. Grammars also increase modularity — by parameterizing one parser with different grammars, it is easy to change the behavior of the parser to handle new visual languages.

To illustrate spatial parsing using visual grammars, consider the visually notated context-free grammar for the very simple family of bar charts shown in Figure 5. The input region to be parsed is marked by a dashed box. Two different views of the successful parse are shown: the concise spiderweb diagram, and the traditional parse tree with labeled nodes (which unfortunately violates the spatial integrity of the arrangement of input elements; the more concise spiderwebs will be used for the remaining examples). Once a visual expression has been parsed as a bar chart, then semantic processing is straightforward. The piece of text at the lower left of Figure 5 represents the interpretation of the bar chart as a table. Parsing also facilitates other kinds of processing; next to the table is a copy of the original bar chart which has been automatically 'prettified'.

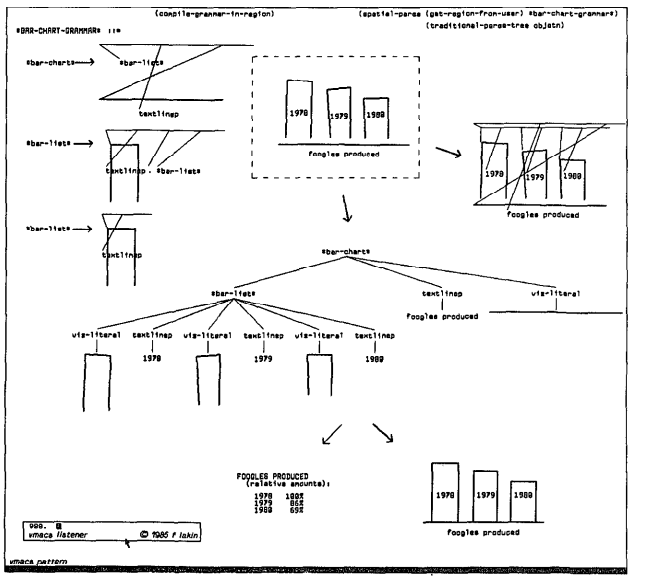

Figure 5. A visual grammar for a very simple family of bar charts, the input region, and two views of the resulting parse tree; and then, based on the parse, textual interpretation and automatic 'prettifying'.

A. Visually Notated Context-Free Grammars A context-free grammar is a 4-tuple consisting of a *terminal vocabulary*, a *non-termininal vocabulary*, a *start symbol*, and a *set of productions* (or rules). The terminal vocabulary is the set of all simple (atomic) visual objects that might be found in an expression to be parsed, and the non-terminal vocabulary is the set of symbols used to represent combinations of terminals and non-terminals. The start symbol is the non-terminal which is the name of the topmost entity being parsed (for instance, "S" for sentence). A production has a non-terminal on the left hand side, and a combination of terminals and non-terminals on the right hand side.

The first three parts of a context-free grammar have been implicitly expressed in the visual representation of the productions, which we call the Visual Grammar Notation. Thus in the grammar for bar charts (Figure 5), the start-symbol is the symbol on the left hand side of the topmost rule, i.e., `*bar-chart*`. Terminals are visual literals appearing only on the right hand side, such as the bar and the horizontal line. Non-terminals are symbols that appear on the left hand side of rules, such as `*bar-chart*` and `*bar-list*`.

B. A Spatial Parser Which Uses Visual Grammars The function `spatial-parse` takes two inputs: a region of space and a visual grammar. It then tries to parse whatever objects are in the region using the grammar. As used by `spatial-parse`, the rules or productions in a visual grammar are visual in two ways: first they are 'about' visual matters; and second they are themselves spatial objects whose spatiality is integral to their functioning as rules.

Each rule is a spatial template, showing the parser where to search spatially for the members of the expression (i.e., where members should be according to that production). In addition, the rule shows what kind of object is acceptable at a location. This can be by example, as with the bar chart bar literal in Figure 5. Testing for acceptable literals can also be by pattern matcher-like predication (e.g. `textlinep` and `?` used in some of the grammars). And finally, each rule expresses the target tree structure to be returned in the parse for its kind of expression.

When a spatial rule is applied to a region of space, either a visual object is returned or the parse fails. In the case of success, the visual object is a copy of the objects in the region put

[4] Robert Futrelle has described an expert system under development which will parse X-Y plots in the form of digital binary images [Futrelle85].

[5] The point of spatial parsing is exactly that the user does *not* have to manually create higher-level pattern structures (even though **vmacs** offers that capability).

into a pattern with its members in the proper tree structure (as described by the grammar). In the case of failure, the parser gives up unless there are any other rules of the same name as yet untried in the grammar, in which case it tries the next one in top-down order [6]. Having two rules with the same name permits termination of recursion, an important feature of context-free grammars. For example, because the second rule in the bar chart grammar defines the non-terminal `*bar-list*` recursively, the grammar can handle bar charts with an arbitrary number of bars. The recursive rule keeps succeeding until there is just one bar left, in which case it fails and the simple rule is tried, which succeeds and terminates the parsing of the `*bar-list*`.

VI. EXAMPLES OF SPATIAL PARSING USING VISUAL GRAMMARS

A. Feature Structures (Directed Acyclic Graph Notation Used by Linguists)

Visual language description: Feature Structures are a notation for directed acyclic graphs of a certain type used by linguists. The Feature Structure (FS) in Figure 6 encodes various kinds of syntactic information *(barlevel, category, next)* and semantic information *(functor)* for a natural language expression (further details on the meaning and function of FS's may be found in [Shieber85]). Looking at the grammar for the FS notation, we see that compared to the grammar for bar charts it uses enclosures and is deeply recursive. The basic FS (`*f-s*`) is a pair of brackets surrounding an atribute-value pair list (`*a-v-p-list*`). An attribute-value pair list is either an attribute-value pair (`*a-v-pair*`) on top of an attribute-value pair list, or simply one attribute-value pair.

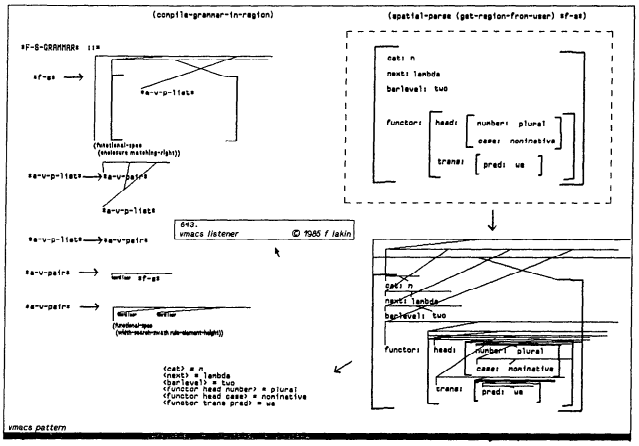

Figure 6. Grammar, parse and interpretation for an expression in Feature Structure notation.

Action taken based on the parse: The parse tree from the FS notation is easily processed to produce an isomorphic LISP s-expression which is then used as input to a function that creates an internal structure of the kind employed by the PATR natural language understanding system at SRI [Shieber85].

B. SIBTRAN (Graphic Devices for Organizing Textual Sentence Fragments)

Visual language description: David Sibbet is a San Francisco based graphic designer who makes his living by writing and drawing on walls to help groups think. He is shown at work in Figure 7. As a first step in dealing with the richness of the informal conversational graphics in Figure 7, a formal visual language was devised. This language, called SIBTRAN, formalizes a limited subset of the visual imagery system developed by Sibbet. SIBTRAN is a fixed set of graphic devices to organize textual sentence fragments under a system which provides a layer of additional semantic information (beyond the meaning of the words in the fragments). A SIBTRAN expression consists of one or more of the graphic elements (hollow arrows, bullets or straight arrows) placed in specified spatial relationships with either pieces of text (sentence fragments of typically six words or less) or other SIBTRAN expressions. The visual grammar for SIBTRAN expressions is shown in Figure 8. The parse and interpretation (text translation derived from meanings used by Sibbet in his work) for a standard SIBTRAN expression were presented back in Figures 2 and 3. This leaves us free to show an extension to SIBTRAN in Figure 8: expressions from two other formal visual languages, Bar charts and Feature Structures, are defined in the grammar as proper SIBTRAN expressions. Thus Visual Grammar Notation allows us to express heterogeneous embedding, using SIBTRAN as a meta-level schema. Figure 8 shows the parse and interpretation for a mixed language SIBTRAN expression with a Feature Structure phrase to the right of the hollow arrow.

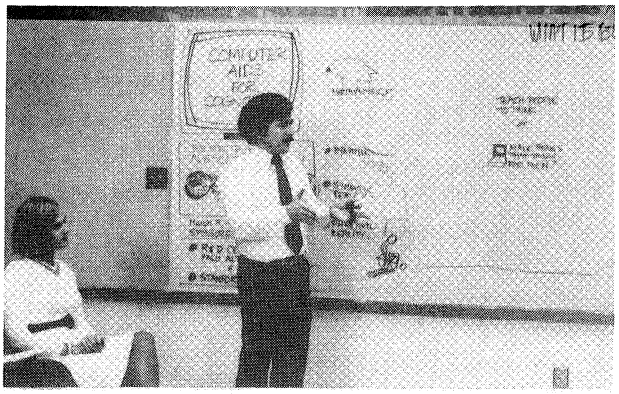

Figure 7. An example of informal conversational graphic communication activity.

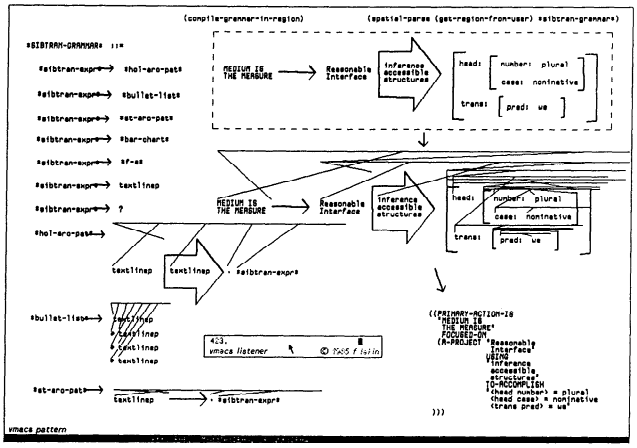

Figure 8. Grammar, parse and interpretation for a mixed language SIBTRAN expression.

Action taken based on the parse: The SIBTRAN-assistant is a helpful interactive software module designed to facilitate graphic communication. The assistant uses the grammar in recognition and parsing. Its job is first to recognize when an arrangement of graphic objects is a SIBTRAN expression, and then to issue special prompts depending on the identity of the expression. The functioning of the SIBTRAN-assistant and its bearing on the conversational graphics problem (Section VII) is discussed in [Lakin86a,86b].

[6] Because the parser calls itself recursively, even if the invocation of a rule finally fails many levels down, control will return back to the originating level where any other rules of the same name will be tried.

C. VISUAL GRAMMAR NOTATION

Visual language description: The Visual Grammar Notation is also a formal visual language. All of the visual grammars used in this paper are described by the grammar presented in Figure 9. Reading informally, an expression in Visual Grammar Notation is a piece of text with the visual literal ": :=" on its right, and a list of rules below it. A list of rules is either a rule on top of a list of rules, or just a rule by itself. And a rule is a piece of text with a straight arrow to the right, and any visual object (drawn line, piece of text or group of visual objects)[7] to the right of that. On the left of Figure 9 is a region containing notation for a visual grammar (using the Visual Grammar Notation to describe itself) and on the right is the proper parse tree for the expression in that region.

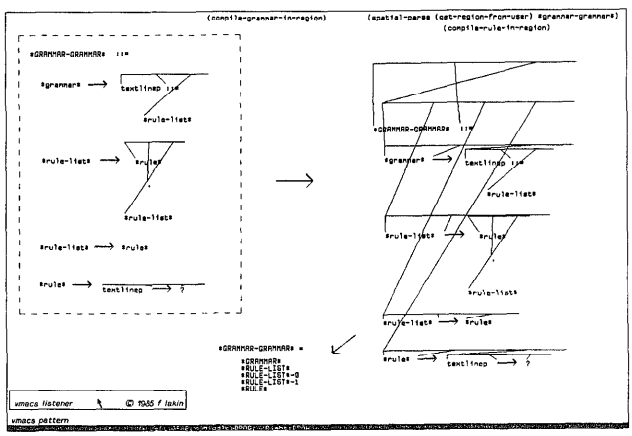

Figure 9. Grammar, parse and interpretation for a visually notated grammar.

Action taken based on the parse: Once a parse tree has been produced for an expression in Visual Grammar Notation, the grammar compiler can then take the tree and convert the grammar to the internal form which is used by the parser. The compiler returns the piece of text at the lower right of Figure 9, showing the name of the grammar and the rules it in. All of the parsing presented in this paper was done using grammars compiled in the above fashion.

VII. THEORETICAL CONTEXT: UNDERSTANDING CONVERSATIONAL GRAPHICS

The overall goal of this research is effective computer participation in human graphic communication activity like that which takes place on blackboards. Blackboard activity is a kind of graphic conversation, involving the *spontaneous generation and manipulation of text and graphics for the purpose of communication.* Figure 7 shows a group participating in conversational graphics. The image in Figure 10-a is the final frame in the text-graphic performance of Figure 7. For purposes of study, that image was transcribed into text-graphic symbols using the **vmacs** graphics editor, Figure 10-b, becoming the corpus for further analysis [Lakin80a,86a]. As a general purpose editor, **vmacs** is a tool for exploring the rules used by humans to collect elementary visual objects into conceptual groups. One possible underlying grouping structure for the image from Figure 10-b is shown in Figure 10-c, and future work will attempt to recover these structures. In the meantime, since phrases from special purpose, *formal* visual languages are often embedded in the imagery of conversational graphics, parsing such languages in order to assist in their use is the immediate goal of the research. We expect that strategies and tools developed for processing visual communication objects in these languages can then be taken 'back' and applied to the more unrestricted, informal domain. Visual Grammar Notation is one such tool, useful for perspicuously describing the patterns of spatial arrangement in specialized subsets of visual communication activity.

[7] Any non-atomic visual objects which serve as right hand sides for rules must be grouped manually by the linguist user when inputing a visual grammar (thus in Figure 9 the right hand sides in the input region to the parser already have spiderwebs). Machine grouping of these objects is very difficult because the tree structure of the right hand side is how the linguist specifies both the spatial search sequence for recognition of that kind of expression, and the tree structure to be returned in case of a successful parse.

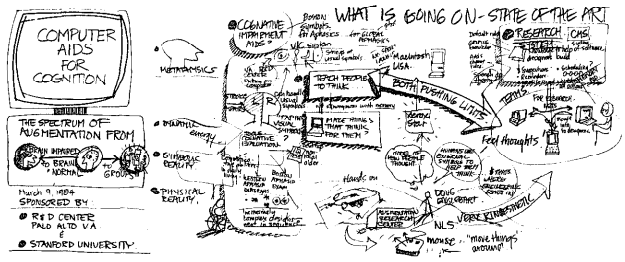

Figure 10-a. Final frame in the conversational graphics performance depicted in figure 7.

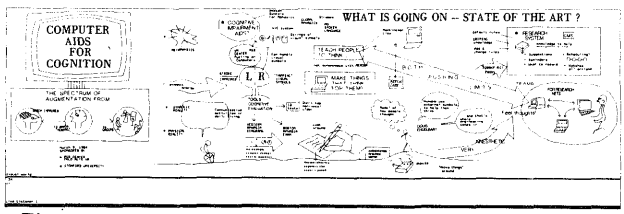

Figure 10-b. **vmacs** transcription of image from figure 10-a.

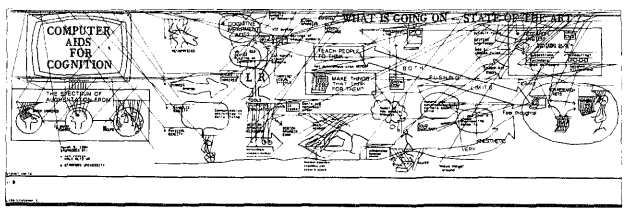

Figure 10-c. Underlying grouping structures for visual objects in figure 10-b.

VIII. SOFTWARE FRAMEWORK The basic software for the research is **PAM**, a LISP-based interactive graphics environment. **PAM** stands for **PA**ttern **M**anipulation, and is an extension of LISP from *computing with symbolic expressions* to *computing with text-graphic forms*, [Lakin80a,80c,83a]. The **PAM** graphics language/system provides tree structured graphic objects together with functions for manipulating them. **vmacs**, the graphics editor in the **PAM** environment [Lakin84a,86a,86b], is the means for arranging **PAM**'s objects into visual language phrases. **PAM** functions can compute with visual objects created in **vmacs** — including both the visual objects representing the grammars and the elements in the visual communication objects to be parsed. The **vmacs/PAM** graphics system is implemented in ZetaLISP on a Symbolics 36xx.

IX. RELATED WORK Use of grammars to analyze formal visual languages was investigated some time ago. Shi-Kuo Chang parsed 2-D mathematical expressions using a "picture-processing grammar" [Chang71]; however the grammar itself was in a non-visual, tabular form and the rules were encoded manually. King Sun Fu extended 1-D string grammar notation with 2-D concatenation operators and graphic objects as the terminal symbols [Fu71]; a graphical grammar to analyze stylized sketches of houses was described, but apparently never implemented. Alan Mackworth has done interpretation of maps sketched freehand on a graphical data tablet; primary local cues are interpreted with help of a grammar-like cue catalog [Mackworth83].

vmacs and **PAM** improve on these earlier researches because they support grammar notations which are both visual *and*

machine-readable. That is, the linguist can directly input a perspicuous notation using **vmacs**. In addition, the visual language users employ **vmacs** to generate the images to be parsed.

X. CONCLUSION *Visual grammars can be useful in the spatial parsing of formal visual languages.* Spatial parsing allows a general purpose graphics editor to be used as a visual language interface. This provides the user with the freedom to use different visual languages in the same image, blackboard-style. He or she can *first* simply create some text and graphics, and *later* have the system process those objects under a particular system of interpretation. The task of spatial parsing can be simplified for the interface designer/programmer through the use of visual grammars. For each of the formal visual languages described in this paper, there is a specifiable set of spatial arrangements of elements for well-formed visual expressions in that language. Visual Grammar Notation is a way to describe the spatial criteria (or rules) which distinguish those sets of spatial arrangements *and* the associated underlying structures; the context-free grammars expressed in this notation are not only visual, but also machine-readable, and are used directly to guide the parsing. Once a visual grammar has been written for a formal visual language, parsing can be accomplished. And once parsed, expressions can then be processed semantically and appropriate action taken. Visual grammars and semantic processing for four formal visual languages have been presented.

Understanding informal conversational graphics taking place in a general purpose graphics editor is the broader theoretical context for this work. Enroute to the overall goal of computer participation in conversational graphics (such as blackboard activity), we began with the parsing of special purpose visual languages. Not only are they simpler, but since they are often embedded in (and may have grown out of) general purpose graphics activity, lessons learned there will likely be applicable to the more difficult problem.

ACKNOWLEDGMENTS The development of spatial parsing within **vmacs** has profited from contributions by Harlyn Baker, John Bear, Pascal Fua, Scott Kim, Larry Leifer, Mike Lowry, Paul Martin, Rob Myers, Alex Pentland, Lynn Quam, Fernando Pereira, Warren Robinett, Ted Selker, Stuart Shieber, Josh Singer, Richard Steele, Hans Uszkoreit, Mabry Tyson, and Machiel Van der Loos.

REFERENCES

[Budge82] Budge, William, "Pinball Construction Kit" software, BudgeCo, Berkeley, CA, 1982.

[Chang71] Chang, S.K., "Picture Processing Grammar and its Applications," INFORMATION SCIENCES, Vol. 3, 1971, pg 121-148.

[Christianson69] Christianson, Carlos, and Henderson, Austin, "AMBIT-G," Lincoln Labs, 1969.

[Fu71] Fu, K.S., and Swain, P.H., "On Syntactic Pattern Recognition," SOFTWARE ENGINEERING, Vol. 2, ed. by J.T. Tou, Academic Press, 1971.

[Futrelle78] Futrelle, R.P. and Barta, G., "Towards the Design of an Intrinsically Graphical Language," SIGGRAPH '78 Proceedings, pages 28-32, August 1978.

[Futrelle85] Futrelle, R.P., "Towards Understanding Technical Documents and Graphics," IEEE/MITRE Conference on Expert Systems in Government, November 1985.

[Glinert84] Glinert, Ephraim P. and Tanimoto, Steven L., "PICT: An Interactive, Graphical Programming Environment," COMPUTER, 17 (11):7-25, November 1984.

[Kim84] Kim, Scott, "VIEWPOINT: A dissertation proposal towards an interdisciplinary PhD in Computers and Graphic Design," Stanford University, August 28, 1984.

[Lakin80a] Lakin, Fred, "A Structure from Manipulation for Text-Graphic Objects," published in the proceedings of SIGGRAPH '80, Seattle, Washington, July, 1980.

[Lakin80b] Lakin, Fred, "Diagramming a Project on the Electric Blackboard," video tape for SIGGRAPH '80, July 1980.

[Lakin80c] Lakin, Fred, "Computing with Text-Graphic Forms," published in the proceedings of the LISP Conference at Stanford University, August 1980.

[Lakin83a] Lakin, Fred, "A Graphic Communication Environment for the Cognitively Disabled," published in the proceedings of IEEE Spring Compcon '83, San Francisco, March 1983.

[Lakin83c] Lakin, Fred, "Measuring Text-Graphic Activity," published in the proceedings of GRAPHICS INTERFACE '83, Edmonton, Alberta, May 1983.

[Lakin84a] Lakin, Fred, "Visual Communication Expert," Public Broadcast television segment on COMPUTER CHRONICLES KCSM, as part of show on Artificial Intelligence, March 22, 1984.

[Lakin84b] Lakin, Fred, "A VISUAL COMMUNICATION LABORATORY for the Study of Graphic Aids," RRandD Merit Review Proposal, VACO, Rehab R&D Center, Palo Alto VA, February 1984.

[Lakin86a] Lakin, Fred, "Spatial Parsing for Visual Languages," chapter in VISUAL LANGUAGES, edited by Shi-Kuo Chang, Tadao Ichikawa, and Panos. A. Ligomenides, Plenum Press, 233 Spring Street, NY NY, 1986.

[Lakin86b] Lakin, Fred, "A Performing Medium for Working Group Graphics," published in the proceedings of the CONFERENCE ON COMPUTER-SUPPORTED COOPERATIVE WORK, Austin, Texas, December 3-5, 1986.

[Lanier84] Lanier, Jaron, `Mandalla` visual programming language as illustrated on the cover of SCIENTIFIC AMERICAN, special issue on computer languages, August 1984.

[Mackworth83] Mackworth, Alan, "On Reading Sketch Maps," Proceedings of IJCAI-77, Cambridge, Mass, August 1977.

[Martin83] Martin, Paul, Appelt, Douglas, and Pereira, Fernando, "Transportability and Generality in a Natural Language Interface System," PROC. 8TH INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE, p 573, IJCAI, Karlsruhe, West-Germany, August 1983.

[Odesta85] Odesta Company, "Helix" software, Northbrook, Illinois, 1985.

[Robinett81] Robinett, Warren, ROCKY'S BOOTS visual circuit programming video game, The Learning Company, 1981.

[Shieber85] Shieber, S.M., "The Design of a Computer Language for Linguistic Information," in PROCEEDINGS OF THE 22ND ANNUAL MEETING OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS, University of Chicago, Chicago, Illinois, July, 1985.

[Stallman81] Stallman, Richard, "EMACS, the Extensible, Customizable Self-Documenting Display Editor," MIT AI Memo 519a, March 1981.

[Sutherland65] Sutherland, Ivan E., "Computer Graphics: Ten Unsolved Problems," Datamation, pages 22-27, May 1966.

[Steele85] Steele, Richard, Illes, Judy, Weinrich, Michael, Lakin, Fred, "Towards Computer-Aided Visual Communication for Aphasics: Report of Studies," Submitted to the Rehabilitation Engineering Society of North America 8th Annual Conference in Memphis, Tennessee, June 1985.

[Tanimoto82] Tanimoto, Steven L. and Glinert, Ephraim P. "Programs Made of Pictures: Interactive Graphics Makes Programming Easy," Technical Report 82-03-03, Dept of Computer Science FR-35, University of Washington, Seattle, Washington 98195, March 1982.

[Woods74] Woods, William, Kaplan, R. M. and Nash-Webber, B, "The Lunar Sciences Natural Language Information System: Final Report," Report 3438, Bolt Beranek and Newman Inc., June 1972.