AN INFORMATION PRESENTATION SYSTEM

Frank Zdybel, Norton R. Greenfeld, Martin D. Yonke, Jeff Gibbons

Bolt Beranek and Newman Inc.
10 Moulton Street
Cambridge, Massachusetts 02238

ABSTRACT

AIPS is a system for graphically presenting information. It promotes a high degree of interactivity between a user and a knowledge base or knowlege-based system, and is designed to be utmost domain independent and extensible. This paper describes the concept of an Information Presentation System (IPS), the intimate relationship between IPS goals and knowledge representation issues, and some of the architecture of AIPS.

I Information Presentation

Interactive graphics is an indispensible technique for putting people in touch with a large knowledge base or knowledge-based system. Graphic output is the best way to communicate a substantial amount of information to a human user because it exploits the high-bandwidth human visual channel.

Graphic input (i.e. user input which points at or otherwise indicates components of a graphic display) is an extremely economical way to describe something; it is much easier to designate an existing depiction than to generate some other descriptor. The descriptional economy of graphic input promotes a feeling of immediacy; the user has the sense of interacting directly with information rather than dealinq with an intermediary.

For these reasons, interactive graphics can play an important role in larqe knowledge-based systems, whether at the interface to the end user or at the interface to the implementor or maintainer. That role is not diminished by progress in natural language or even speech understanding. Graphics and natural language do not compete in terms of functionality; rather, they complement each other.

* This research has been supported by the Defense Advanced Research Projects Aqency under Contract No. N00039-79-C-0316, ARPA Order No. 3740, and monitored by the Naval Electronics Systems Command (Program Code 9D30).

A problem which limits the use of interactive graphics is the expense of implementing such interfaces. The composition of a graphic display involves a vast number of decisions ranging over issues of format, content, layout and style. Anyone who has prepared diagrams for a publication or a talk (even with the help of a skilled draftsman) understands something of the effort that goes into good graphic presentation.

The tools ("graphics languages") commonly used for building interactive graphics address this problem only at its lowest level: displays must be described in terms of primitive elements such as points, lines and regions. In effect, interactive graphics is costly because the implementor is forced to assume the draftsman's drudgery. Additionally, the resulting interface is usually heavily involved with representational details of the system for which it is intended. Chanqes made to the system or its underlying representations tend to propagate directly into the graphic interface.

An Information Presentation System (IPS) is a more powerful tool for interactive graphics. By an IPS, we mean a system that:

1. Automatically qenerates displays according to (primarily) content-oriented specifications
2. Provides a systematic basis for interpretation of user qraphic input
3. Functions reasonably well without demanding custom-tooling to a particular application
4. Is easily extensible to satisfy domain and user-specific display requirements.

This abstraction of the display generation function into a broadly applicable tool can confer many side-benefits because the IPS can be enhanced and elaborated in ways that would be impractical or uneconomical in a single use interface. The interface by which the end-user controls the display function can be built up, even to the point of allowing natural language specification of displays [1]. A large repertoire of display formats can be accumulated. A high degree of sensitivity to the human end-user can be built in. Not least important, an IPS is a place to embody a consistent set of decisions about the human factors of graphic display.

Perhaps the best early, yet embryonic, exemplars of information presentation occurred in the SMALLTALK programming environment [2] and some of the work based on it (such as THINGLAB [3] and the SMALLTALK Browser [4]). With the exception of some network-map displays generated by THINGLAB, these displays were limited to tabular formats. Nevertheless, they represented a definite step toward information presentation as we have defined it here. SMALLTALK'S class hierarchy and method inheritance mechanism made it possible not only to define and apply very general presentation methods, but also to modify or extend these methods to deal with class definitions further down in the hierarchy.

Other current work on the information presentation paradigm includes the VIEW system [5] under development at OCA. This effort is an attempt to produce an IPS which responds to queries by automatically generating SDMS [6] displays. Tt has also been reported in the literature [7] that the DBMS component of the Cedar programming environment for MESA [8] will include an IPS as the basis for an application-independent user interface.

Our work differs from these efforts in two respects. First, we are committed to generating (and interpreting user input over) a wide variety of graphic display formats, including not only tabular displays but also maps, graphs, and diagrams. Second, we view information presentation as fundamentally implying an extremely rich characterization of the structure of displays and their semantic content.

In the remainder of this paper we will first discuss the premises of our approach to information presentation, and then give a more detailed description of our current prototype system.

A. Foundations for Information Presentation

The premise of information presentation as a domain-independent activity is that it is possible to make reasonable decisions about display structure based on limited knowledge about display content. For example, it is possible to draw a map of a college campus without knowing very much about what buildings are or what they are used for. One can get along with only the limited knowledge that buildings are discrete physical objects, occupy fixed regions in a two-dimensional space, have names, etc. Moreover, the methods used to produce such a map would presumably suffice for any other set of discrete physical objects having those sorts of attributes. If this were not the case, a domain-independent display generation capability would be impossible.

It is also possible to make discriminating choices among alternative display formats on just such limited knowledge [9]. A map is a good choice for showing the locations of physical objects; pie charts are good for depicting exhaustive partitions on enumerable sets, and so forth.

However, to apply the limited "common sense" knowledge relied on by the IPS to some particular knowledge base, there must be a mechanism for supporting the necessary generalizations. To continue the above example, there must be some way to bridge the gap between "building" and "physical object". Our approach assumes that the external knowledge base will provide this in the form of a generalization hierarchy which, at its least specific levels, matches or relates to the distinctions utilized by the IPS. In other words, the subject knowledge base must include a covering lattice of "IS-A" links, some portion of which is also recognized by the IPS.

Fortunately, most current knowledge representation languages rely on this sort of hierarchy for organizing knowledge. Those languages which allow multiple super-categorization further simplify the problem of connecting the IPS and the knowledge-base. They make possible an approach of generating simplified descriptions of the subject domain for the specific purpose of driving the IPS, eliminating the requirement that the knowledge base incorporate exactly the distinctions utilized for information presentation.

The Advanced Information Presentation System (AIPS) [10, 11] which this paper describes assumes a knowledge base expressed in KL-ONE [12, 13]. Among current knowledge representation languages, KL-ONE affords a particularly good basis for information presentation because it provides a richer-than-usual generalization structure; one which extends to the parts or attributes of descriptive entities as well as to the entities themselves. This explicit attribute inheritance mechanism avoids the problem of "slot" naming confusions common to most network formalisms. From the standpoint of the IPS, the extra structure enables more informed decisions about how to depict parts or attributes of a description. Without it, the IPS would be either dependent on attribute names for making these decisions, or would have to deal strictly in terms of whole entities, without regard to the functional roles they play as constituents of other entities.

The foregoing discussion has focused on what a domain-independent IPS requires in terms of the general structure of the subject knowledge base. If these requirements are satisfied, there remains the problem of how to make the IPS easily extensible to handle domain-specific display requirements. For example, the client who supports development of AIPS desires certain map and table formats that have an established currency in the domain of Naval Command and Control. Similarly, a knowledge-based system for assisting the design of LSI circuits may require special display formats which are roughly "maps", but which conform to the particular conventions and requirements of that domain.

Also, the end user of a domain-tailored IPS may have his or her own unique requirements for the format of a display. Often, these can be expressed as slight variations on some display format already supported by the IPS. For example, a naval tactician might want an otherwise standard format situation map in which ships with a certain capability are given a distinct depiction. In other cases, individuals may need to create their own formats starting more nearly "from scratch". For example, the implementor of a knowledge-based system may want a special display format that helps track down some specific class of bug.

In sum, an IPS must comprise an open-ended and extensible model of the display generation process. Moreover, the structure of this model must be such that additions to it can make the maximum possible use of behavior that has already been captured. Accordingly, AIPS' most prominent architectural feature is a taxonomic hierarchy of display format descriptions. Inheritance of attributes and attached procedures down the structure of this hierarchy allows new display formats to be described to the greatest possible extent in terms of those already represented.

It is possible to pursue this kind of approach using a LISP enhanced with an object class hierarchy package, such as FLAVORS [14], or an object-oriented programming language such as .SMALLTALK. However, that would not address a remaining important issue affecting the extensibility and flexibility of an IPS: the extent to which its behavior is expressed as an interpretive process written in some programming language. Particularly for non-programming users, control and modification of the IPS ultimately depend on the degree to which display generation behavior can be declarative!y described. Unless the interpretive process of the IPS model is very general, changes or additions to the model will often require changes or additions to the interpreter.

Because we are committed to developing an IPS which offers maximum flexibility and control to the non-programming end user, our view of an IPS is that it is itself a knowledge-based system. Accordingly, we have iinplemented AIPS as a KL-ONE taxonomic hierarchy of display structure descriptions. The interpreter for this knowledge base is written in LISP, and the bulk of it is distributed over the hierarchy in the form of inheritable attached procedures. Our current research efforts are aimed primarily at the problems of reducing the LISP component of the system. Our ultimate goal is an IPS that can respond to the range of initiatives which might be expressed in a dialog with a human draftsman.

B. Presentation System Architecture

The following provides some details about the internal architecture of AIPS in order to lend concreteness to the above discussion. The system's description structure of displays and their formats, its characterization of the information content of a display, and its use of procedural knowledge are described. The discussion makes heavy use of KL-ONE terminology, and readers unfamiliar with KL-ONE will find [13] useful in learning more.

For the sake of clarity, we will adhere to the following typographical conventions. The names of KL-ONE Generic Concepts are printed entirely in bold capitals. The names of Individual Concepts are simply capitalized (where the name is a compound word, sub-words will also be capitalized). The names of Roles (properties of concepts) are capitalized and underscored. Thus, DISPLAY refers to the description of a category, Display refers to an individual of that category, and display refers to something seen on the screen of a graphics terminal.

1. Display Description Structure

As shown in Figure 1, various display formats are represented as sub-categories of DISPLAY, which is the principal top-level Concept of the information presentation model. There are two important themes in each such description: a characterization of the information involved in the display, and a characterization of its visible components. These are represented respectively by the Application and Realization Roles of DISPLAY.

The Realization Role is differentiated (split into multiple sub-Roles) and modified at the various descendants of DISPLAY. For example, MAP's visible components: Border, Label, Legend, Item, etc. are all represented as differentiatinq sub-Roles of DISPLAY'S Realization Role (See Figure 2).

Notice that the ValueRestriction of Realization is the concept DISPLAYITEM, which is a super-Concept of DISPIAY. DISPIAYITEM provides a characterization of a piece of a display in purely graphic terms. Attribute Roles of DISPLAYITEM relate such thinqs as the location, orientation, scale, width and height of a display element. Because DISPIAYTEM is a super-Concept of DISPLAY, the eventual fillers of (sub-Roles of) the Realization role in any Display may be either simply treated syntactically (as Displaylterns) or may in fact be Displays in their own right, carrying an explicit treatment of the information they depict. Also, any Display inherits all of the syntactic attributes of DISPLAYITEM, and thus can be described in terms of attributes such as location, width, heiqht, etc. The advantage of treating the structural components of a display as Displays (and thus making an explicit treatment of their semantic content) is that this can provide a basis for the interpretation of graphic input.

The Application Role of DISPLAY indicates the information being expressed in a given Display, thouqh not necessarily all of the information involved. Rather, this Role is a kind of binding mechanism that characterizes a Display's application to or depiction of specific information, as opposed to the inherent use of information that might be made in the Display's generic definition (e.g. a sub-category of MAP which always labels items with their names, regardless of whether or not that attribute is mentioned at the Application.)

Of course, a sub-category of DISPLAY may add additional Roles that are sub-Roles of neither

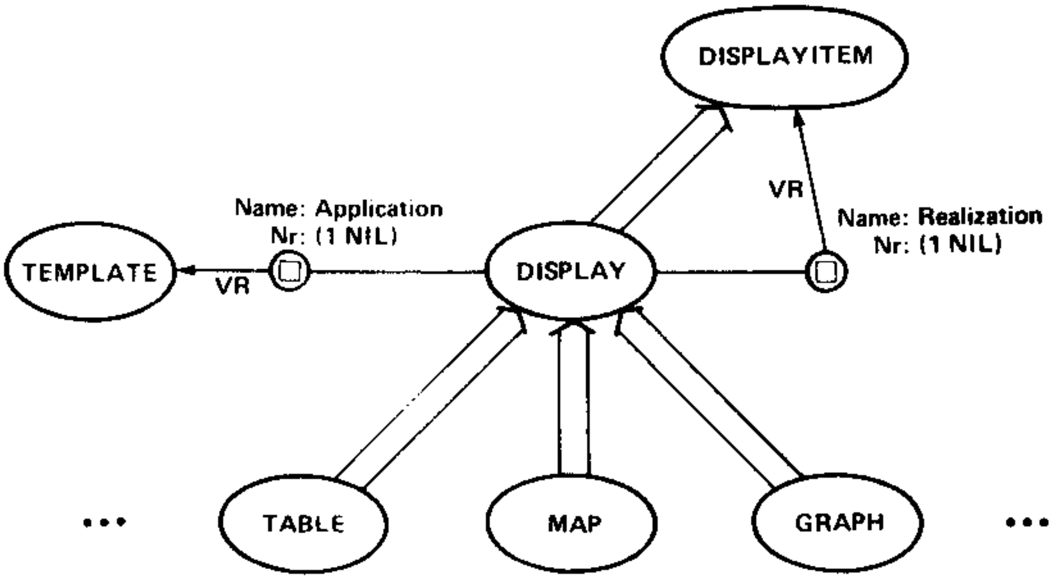

Figure 1: DISPLAY and its Sub-Categories

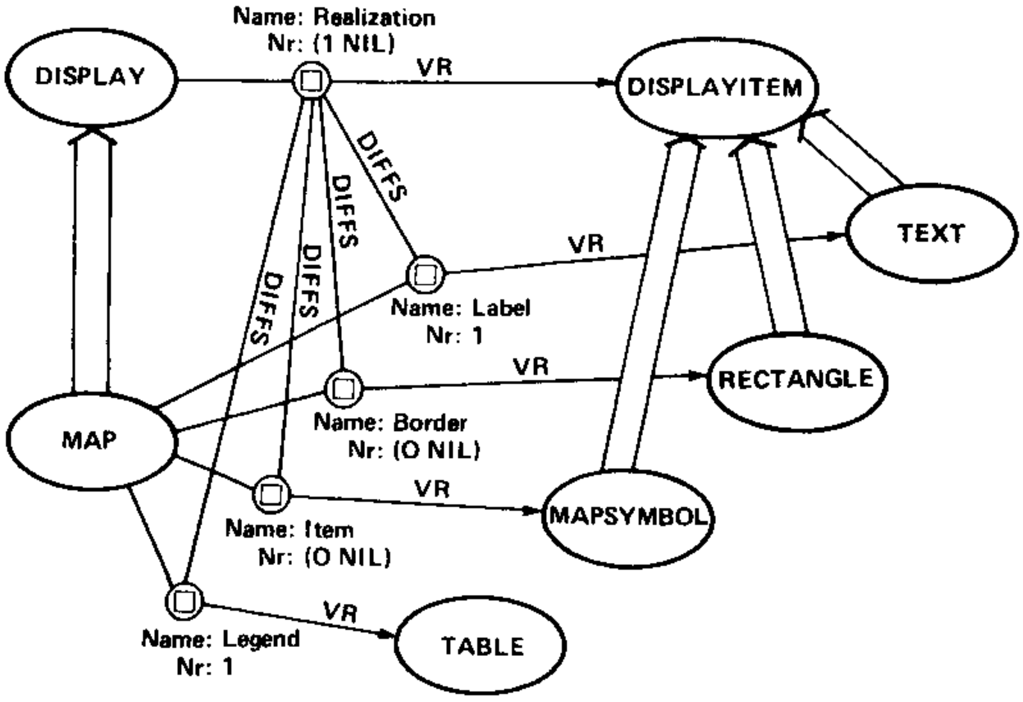

Figure 2: The Internal Structure of a Format's Realization

Realization nor Application. For example, MAP may have a Role Injection which characterizes the scaling transformation for a map.

2. The Characterization of Information

At DISPLAY, the Application Role is value restricted to one or more individuals of category TEMPLATE. A Template is simply a means of indicating the cross product of some set of descriptions in the subject knowledge base with some subset of their Roles. Thus, TEMPLATE has two roles: GonceptGroup and RoleGroup. The fillers of ConceptGroup indicate the domain model objects of concern In a Display; the fillers of RoleGroup indicate which attributes of those objects are involved. If the semantic content of a display does not factor into a single such cross product, several Templates may be used to capture the disjuncts.

A Template gets its necessary descriptive "grip" on Concepts and Roles in the domain knowledge base through use of KL-ONE's meta-description feature. The Roles to be included in a Template are themselves treated as objects to be described. Individuals of the meta-Concept ROLE are used to describe them. *These* individuals become the fillers of the RoleGroup of the Template. Meta-description is used in a similar way to indicate the Concepts involved in a Template via the meta-Concept CONCEPT. See Figure 3 for an example of a Template used to meta-indicate the names and locations of two ships.

Templates are also useful as meta-descriptions of display formats. For example, a Template can be used to indicate the kind of information that is well depicted in a particular format.

3. Procedural Knowledge in the Presentation System

The process of creating a display begins by creating a "blank" individual (i.e. none of its Roles are yet filled) of some sub-category of DISPLAY and filling its Application with the Template or Templates specified by the user. If the user does not specify the desired format, AIPS compares the Template with meta-descriptions on the formats it knows about and selects a suitable category. From this point, display construction proceeds in three phases: derivation, location, and drawing. Each phase is supported by its own set of procedural attachments to the knowledge base.

During the derivation phase, the description of the Display is expanded at least to the extent that all sub-Roles of Realization are filled with some DisplayItem; each of these descriptions is then

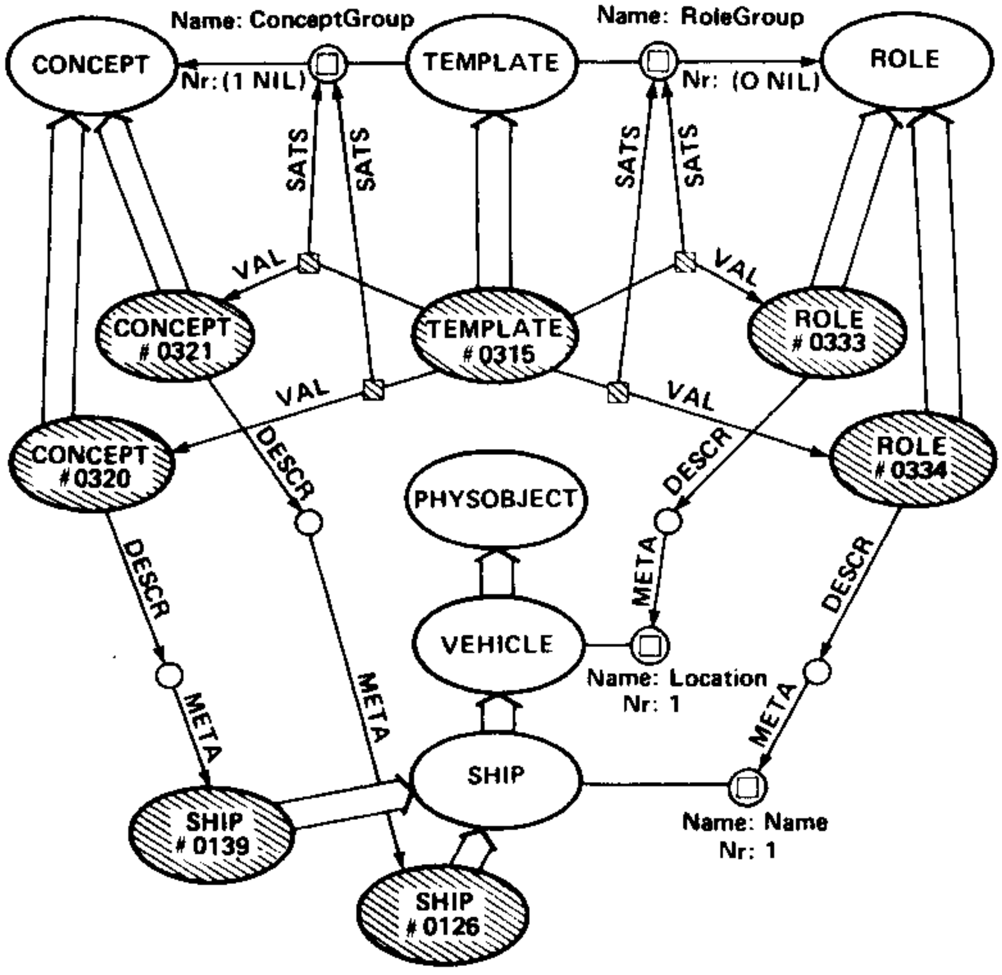

Figure 3: How a Template Meta-Indicates Information

recursively expanded in the same manner. For example, for the case shown in Figure 4, the Legend Role of the Map being constructed is filled with an individual of TABLE whose Application Role filler indicates the set of symbols used in the map. The derivation process then recurses on this Table to fill out its internal structure.

The derivation phase procedures are attached as tags to the Roles for which they produce fillers. These tags specify what information must already be known in order to run a procedure, so that a demand to fill a Role may first result in the filling of other Roles on which its derivation depends. Also, the body of an attached derivation procedure can dynamically call for the derivation of some other Role and suspend processing until the information is provided. Derivation procedures are inherited and more than one derivation procedure may be attached. At Role filler derivation time, all of the available procedures are tried in order from the more specific to the more general until one succeeds.

The second or location phase proceeds by means of messages passed among the constituent objects of the Display, which were identified and constructed in the preceding derivation phase. Displaylterns receive ToLocate messages which tell them where they are located relative to the coordinate system of the Display's viewing surface (which entities by now have also both been completely described) . If the recipient Displayl tern contains separately described components, the attached ToLocate procedure computes the locations of these constituents and recurses the location process by dispatching further ToLocate messages.

The final or drawing phase is handled in a similar manner. Displaylterns receiving ToDraw messages execute ToDraw procedures which ultimately call the drawing routines of a graphics package. Displaylterns with separately described components send further ToDraw messages.

C. Conclusions

The version of AIPS described here runs on a Decsystem-20 under the Interlisp-10 interpreter, using a bitmap graphics terminal of BBN's design. The BMG graphics language [15] used in this work was developed and implemented as part of the AIPS effort, which also made substantial contributions to the current implementation of KL-CNE.

AIPS was conceived as a display management tool suitable for work environments supported by fast personal machines with large virtual memories, such as the MIT CADR [16], Xerox PARC's Dorado [17], or the Jericho symbolic processor developed here at BBBN [18, 19]. The current AIPS is a carefully delimited prototype which barely fits into Interlisp-10's available storage. We are presently moving AIPS (and the Interlisp environment which

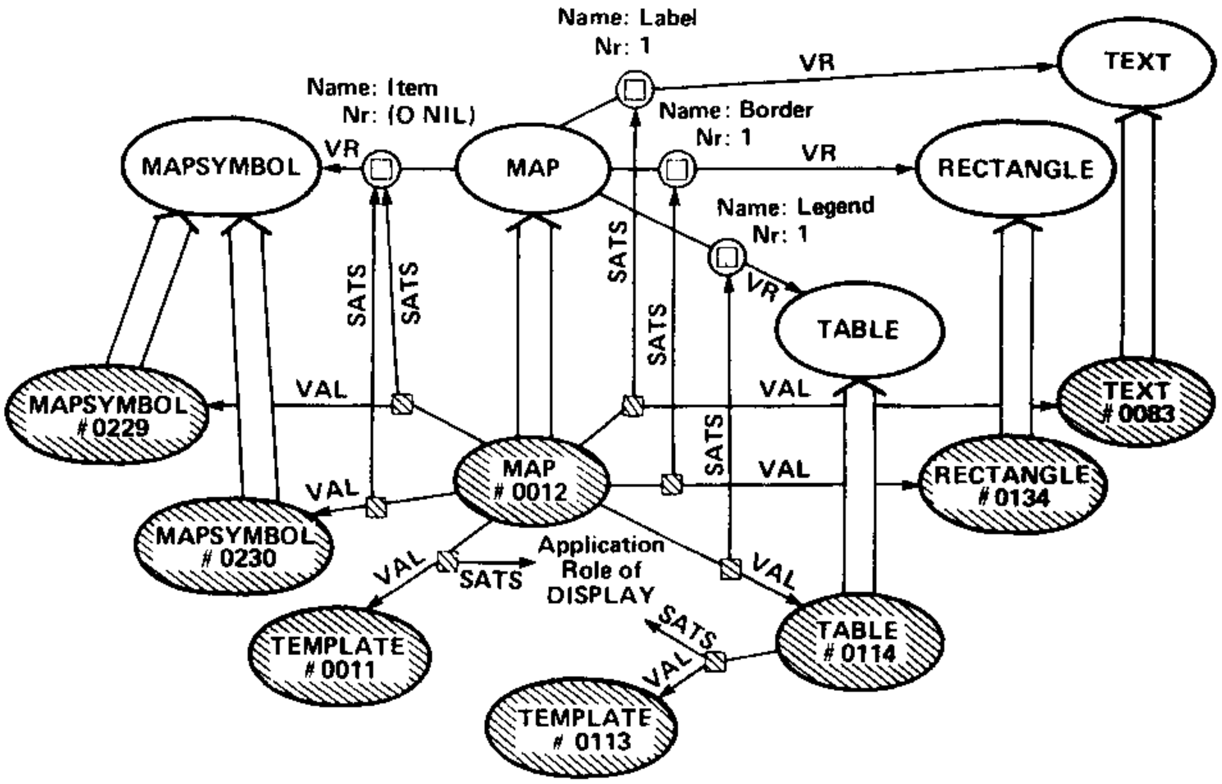

Figure 4: The Description of an Individual Display

supports it) onto Jericho. When that has been accomplished, we will be in a position to further elaborate the structure of AIPS' display descriptions.

Many of the benefits of information presentation are available through less powerful and more widely available tools than KL-ONE and LISP, albeit at the cost of some generality. The broadest importance of our work is that it demonstrates one method of raising the level of interaction between a knowledge base and its graphic display function. We are not alone in this pursuit. What distinguishes AIPS is its direct assault on the inherent knowledge representation issues.

REFERENCES

(I) Woods, W.A., "A Scenario for Natural Language Control of Visual Displays," BBN Report No. 4693, Bolt Beranek and Newman Inc., June 1981.

[2] Kay, A. and Goldberg, A., "Personal Dynamic Media," IEEE Computer, March 1977.

[3] Borning, A., THINGLAB - A Constraint-Oriented Simulation Laboratory, PhD dissertation, Stanford University, March 1979.

[4] Robson, D., personal communication, 1980.

[5] Herot, C.F., Wilson, G.A, "Semantics Versus Graphics - To Show or Not to Show," Technical Report CCA-80-09, Computer Corporation of America, March 1980.

[6] Donelson, W., "Spatial Management of Data," Proceedings of SIGGRAPH 78' Association for Computing Machinery, Atlanta, GA.

[71 Cattell, R.G.G., "Integrating a Database System and Programming/Information Environment," Proceedings of the Workshop on Data Abstraction, Databases, and Conceptual Modelling, National Bureau of Standards and the Association for Computing Machinery, Pingree Park, CO, June 1980.

(8) Mitchell, J., Maybury, W. and Sweet, R.Xerox Palo Alto Research Center, Mesa Language Manual, Version 5.0 ed., Palo Alto, CA, 1979.

(9] Gnanamgari, S., "Providinq Automatic Graphic Displays through Defaults," Proceedings of the Third National Conference of Canadian Society for Computational Studies of Intelligence, Canadian Society for Computational Studies of Intelligence, May 1980.

[10] Yonke, M.D., and Greenfeld, N.R., "AIPS: An Information Presentation System for Decision Makers," Proceedings of the Thirteenth Hawaii International Conference on System Sciences, the University of Hawaii and the Association for Computing Machinery, January 1980. A revised version of this paper is also available as BBN Report No. 4228, Bolt Beranek and Newman Inc., December 1979.

[11] Zdybel, F., Greenfeld, N.R., and Yonke, M.D., "Application of Symbolic Processing to Command and Control: An Advanced Information Presentation System, Annual Technical Report," BBN Report No. 4371, Bolt Beranek and Newman Inc., April 1980.

[12] Brachman, R.J., Bobrow, R.J., Cohen, P.R., Klovstad, J.W., Webber, B.L., Woods, W.A., "Research in Natural Language Understanding - Annual Report: 1 Sept 78 - 31 Aug 79," BBN Report No. 4274, Bolt Beranek and Newman Inc., August 1979.

[13] Brachman, R.J., "On the epistemological status of semantic networks," in Associative Networks —the Representation and Use of Knowledge in Computers, N.V.Findler, ed. ,, Academic Press, New York, 1979, pp. 3-50.

[14] Weinreb, D. and Moon, D., Lisp Machine Manual, Massachusetts Institute of Technology Artificial Intelligence Laboratory, Cambridge, MA, 1981, ch. 20.

[15] Greenfeld, N.R., Zdybel, F., Ciccarelli, E., and Yonke, M.D., "BMG Reference Manual," BBN Report No. 4368, Bolt Beranek and Newman Inc., April 1980.

[16] Greenblatt, R., Knight, T., Holloway, J., Moon, D., The LISP Machine, Massachusetts Institute of Technology Artificial Intelligence Laboratory, 1978.

[17] Lampson, B,W.,and Pier, K.A., "A Processor for a High-Performance Personal Computer," 7th Int. Symp. on Computer Architecture, 7th Int. Symp. on Computer Architecture, Le Baule, France, May 1980.

[18] Greenfeld, N.R., "Jericho: A Professional's Personal Computer System," 8th Int. Symp. on Computer Architecture, 8th Int. Symp. on Computer Architecture, Rochester, Mn., May 1981.

[19] Yonke, M.D., "A Lisp Machine should only be thought of as an expensive terminal," Proceedings of the 11th Annual Microprogramming Workshop, Proceedings of the 11th Annual Microprogramming Workshop, November 1978.