Understanding Text With an Accompanying Diagram

William C. Bulko

Artificial Intelligence Laboratory*
University of Texas at Austin
Taylor Hall 2.124
Austin, Texas 78712

Abstract

Because English is typically not a good language for expressing spatial relationships, people frequently use diagrams to supplement textual descriptions in engineering and scientific applications. BEATRIX is a computer program which takes as input an English statement of a physics problem and a figure associated with it, understands the physical objects described and the relationships between them, and produces a data structure containing a model of the problem. Using a simple graphic interface provided by BEATRIX, the user sketches a picture using a mouse-based menu system similar to other familiar drawing programs. The user may also enter normal English sentences through the keyboard, using the mouse to help in editing. Meanwhile, BEATRIX transparently creates an internal representation of the picture similar to that which might be produced as the output of a vision system. It then translates the text and the picture representation into a semantic network model, resolves the references between objects common to the two sources of knowledge, and produces a unified model of the problem world. The correctness and completeness of this model can be validated when it is supplied as input to a physics problem-solving program which is currently under development.

A major problem in the field of knowledge representation is that of *coreference*: recognizing when two descriptions contain references to the same object or set of objects so that information is associated with the correct object in the internal model. Flexible, opportunistic control is necessary in order to recognize coreference and to act upon it; accordingly, the understanding module of BEATRIX uses a blackboard control structure. The knowledge sources for the blackboard contain information for identifying and matching common objects in physics problems. The blackboard also handles the semantics of parsing the English text and the identification and association of picture elements.

It is natural and often necessary for people to communicate using a combination of text and pictures. There are many applications in science and engineering where the ability to communicate in this way with a computer could be used to a great advantage. We believe that BEATRIX illustrates a control structure and collection of knowledge that can successfully implement text and picture understanding by computer. We also believe that this organization can be applied successfully to similar understanding tasks in domains other than physics problem-solving, where sources of knowledge such as the output from vision systems and speech understanders can be used in place of text and pictures.

*This research was supported by the U.S. Army Research Office under contract DAAG29-84-K-0060. The A.I. Lab has also benefitted from major equipment grants by Hewlett Packard and Xerox.

1 Introduction

This paper describes a program which understands elementary physics problems, parsing the text and picture components of the problem together, and produces an abstract model of the information contained within. The program, BEATRIX, is part of a project in automatic physics problem solving currently underway in the Artificial Intelligence Laboratory of the University of Texas at Austin. The development of BEATRIX was proposed for the investigation of two major problems:

- How can we understand the mechanisms by which multiple and symmetric knowledge bases containing references to common objects can be parsed together? How can we discover the kinds of knowledge necessary to implement these mechanisms?
- Can the parsing of pictures be approached in ways similar to those used in parsing natural language? Are there "picture grammars" where picture elements can be classified into "parts of speech"? A syntactic approach to picture parsing has previously been formalized [1]. How can this be exploited in a practical problem-solving system?

The first problem we term the *coreference problem.* Coreference between two knowledge bases refers to the property where each knowledge base contains information about objects known to the other. Within our domain, these knowledge bases are the text component and the picture component of the physics problem. We organize the information contained within the two coreferent knowledge bases into three levels of abstraction:

- a *concrete* level, which corresponds to the information presented as input by the user. The diagram is viewed at this level as merely a collection of lines, boxes, and circles, and the input text is simply a list of sentences.
- an *intermediate* level consisting of frames corresponding to objects represented by elements on the concrete level. Here, the picture elements of the diagram have been interpreted as masses, surfaces, angles, and other objects, and the text has been partially parsed.
- an *abstract* level which contains all objects described in the knowledge bases, and relationships between them. This can be viewed as the union of the intermediate levels with all coreferences resolved and the relationships between the objects specified. This level also forms the basis for the final model produced by the system.

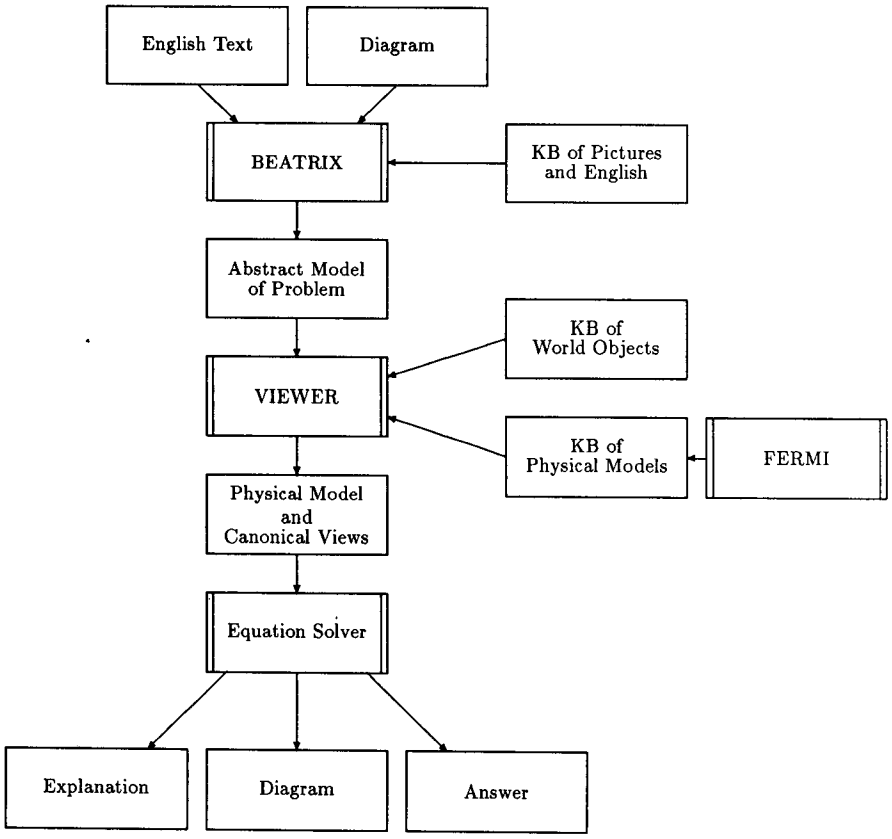


Figure 1: The Physics Problem-Solving Project

BEATRIX uses a blackboard data structure to implement this organization of its knowledge bases and their resulting levels of abstraction.

As a result of restricting our study to the domain of elementary physics problems, we provide ourselves with easy access to large collections of ready-made test cases (in the form of college-level textbooks) of which the vocabulary size and the number of basic picture elements are manageable for an experimental system. As test input to the system, we have selected a set of physics problems of varying types from several college-level physics textbooks. We are applying these problems to learn what new knowledge each problem introduces and how it can be represented within BEATRIX.

A run of BEATRIX begins with the input of the physics problem into the system. A graphic interface is built into BEATRIX to facilitate the entry of the text and the drawing of the picture. When the problem has been entered and is ready to be processed, BEATRIX analyzes the picture at the pixel level to determine which picture elements touch each other: this process provides a great deal of information which will later assist in the identification of picture elements by the ways they interrelate. Next, the text, picture elements, and touch relations are posted on the lower levels of a blackboard, and identification of the objects begins. The text is partially parsed, and hypotheses are made for identifying the picture elements as objects in a physical system. Hypotheses are propagated up the blackboard, with objects appearing on the top level as matches are made between objects from the picture and objects from the text. The result is a model of the problem listing the physical objects mentioned in the problem, their properties, the ways they touch and are attached to each other, and the forces they exert on each other.

2 The Physics Problem-Solving Project

BEATRIX is the first phase of a project investigating methods by which a computer program can understand an informal statement of a physics problem, model the physical situation described by the problem, and produce a numerical or algebraic solution to it. The origins of the project come from previous work on a program called ISAAC [2] which was able to understand rigid body statics problems stated in English, solve them, and output diagrams derived from the models it formed. The goals of the current project include solving a greater range of problems, handling more diverse forms of human input, and providing the user with more freedom in controlling the problem-solving process.

Figure 1 illustrates the current organization of the Physics Problem-Solving project. In the projected system, the user would enter a physics problem in the form of English text and a diagram. The first phase of the system, BEATRIX, will have a knowledge base containing information about picture elements and the real world items they could represent, an English vocabulary, and rules for parsing English. BEATRIX will produce an abstract model of the problem listing the objects described and the ways they interact with each other.

The second phase of the system, called VIEWER, is also currently under construction [3]. VIEWER is an expert inference system which uses knowledge about world objects and how to model them according to their roles in a physical situation, and translates the abstract problem model into a canonical physical model; for example, VIEWER would be able to determine that a box should be interpreted as a point-mass in a pendulum problem, but as a surface with a coefficient of friction in a normal-force inclined plane problem. Currently, the canonical

objects and physical models known to VIEWER are fixed and predefined, and the user interactively selects the appropriate model. An interface is being developed (FERMI) which would facilitate this selection process.

The physical model produced by VIEWER will be of a form suitable for use by an equation solver similar to that used by the ISAAC program to produce a solution to the physics problem. The output of a diagram based on this final model may be useful for comparison to the input in order to study the mechanisms of the problem-solving process. An explanation system using the histories of BEATRIX, VIEWER, and the equation solver may also be included.

3 BEATRIX

The Xerox 1108 workstation was selected for the BEATRIX project because of its graphics capabilities and its environment. BEATRIX is written in Interlisp, the native language of the 1108, and GLISP [4], an object-oriented extension of Interlisp.

3.1 The graphic interface

BEATRIX provides a graphic interface to assist the user in entering a physics problem. When the interface is initialized, five windows appear on the screen:

- The *chalkboard*, a large window where the picture is drawn by the user.
- The *text window*, a window where English text can be entered and edited.
- The *chalkboard operations window*, which contains a central menu from which all major BEATRIX commands may be selected.
- The *top level typescript window*, a Lisp listener or executive window.
- The *prompt window*, where prompts to the user and other help messages are printed.

One function of the BEATRIX graphic interface is to facilitate entry of the physics problem text. When the *edit* icon of the command menu is selected, a mouse-based text editor is invoked in the text window. The user can then type in a collection of English sentences in the form of a Lisp list-of-lists. Cursor movement using the mouse works as one would expect.

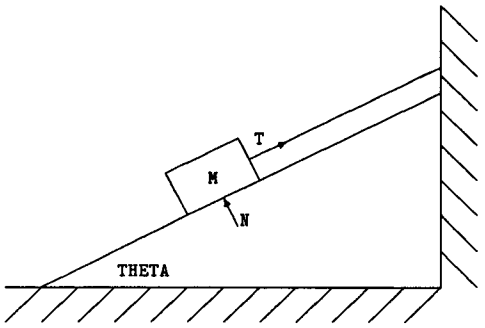
To draw a picture, the user selects one of several graphic icons from the menu in the operations window. These icons correspond to function calls which create new instances of lines, circles, spline curves, and other useful picture elements. When one of these is selected, a representative object of the appropriate type appears in the corner of the chalkboard window. Using the mouse, the user moves the object within the chalkboard to the desired location, and shapes and sizes the object as he wishes. By selecting other icons, the user may create additional new picture elements of various types, delete or modify existing ones, or erase everything. Also included is an *undo* command.

As each new picture object instance is added to the picture, BEATRIX creates a corresponding GLISP object which stores its characteristics, such as its endpoints or extent, and its class-dependent properties. These GLISP objects collectively define a model for the picture. A typical object `LINE8`, a dashed line with an arrowhead at its left endpoint, might appear as:

```
(class LINE
 endpt1 (129 .  142)
 endpt2 (354 .  173)
 dashed T
 ah1 T
 ah2 NIL)
```

A useful feature of the interface is a "correction" facility by which minor drawing errors are immediately and automatically corrected. For example, if the user wants to position a new line so that it shares an endpoint with an existing line, it isn't necessary to place the cursor precisely at the desired pixel of the chalkboard: if the mouse button is clicked anywhere within ten pixels of the existing endpoint, the cursor will immediately jump to the intended location. The correction facility uses a predefined hierarchy to determine what correction should be made. A *special point* is defined as either a point specified by the user when the object was drawn or a point on the object that is normally considered to be significant; for example, the endpoints or the midpoint of a line, the corners of a box, or the center of a circle *and* the other end of the radius supplied by the user. If the cursor is near a special point when the mouse button is clicked, the special point is interpreted as the intended destination. If the cursor is not near any special point, but is close to *any* other point on another object (such as the side of a box), the cursor will jump to the nearest point of that object, as measured along a perpendicular. Otherwise, the cursor is left alone. Of course, the correction facility may be turned off by resetting a flag, so that the user may remain the ultimate authority of his own intentions, a useful feature when specifying points in an area crowded with "special points". In addition, the user may reset the *threshold* value or sensitivity of the interface, the maximum number of pixels away an object may be to be considered sufficiently close for a correction to occur. Like many features of the BEATRIX interface, the correction facility is not a new invention [5]. It is, however, a useful feature worthy of notice, and should be considered in any standardization of modern graphics packages and drawing programs.

There are also menu commands enabling the user to save the current problem text and picture to a file, and to load physics problems saved in such a way from a file. Figure 2 illustrates a typical problem suitable for BEATRIX. When the user is satisfied with the picture and English text, the *solve* menu command can be selected to initiate processing. First, BEATRIX analyzes the picture to determine which of the picture objects touch, creates a *relation* object for each instance, and adds it to the picture model. It then initializes the core of BEATRIX, the *understanding module*, which contains the mechanisms for parsing the picture and sentences and for resolving the coreferences between them.




```
((A MASS IS HELD IN POSITION BY A CABLE ALONG A SMOOTH
  INCLINE)
 (IF THETA = 60 DEGREES AND M = 50 KG , FIND THE TENSION
  IN THE CABLE AND THE NORMAL FORCE EXERTED BY THE
  INCLINE))
```

Figure 2: A typical problem for BEATRIX

3.2 The blackboard structure

The understanding module is a blackboard system, and was constructed using BB1, a domain-independent tool for building systems with blackboard architectures [6]. The blackboard, the central data structure used by BEATRIX, is organized into five levels:

- the PICTURE level, which contains the picture objects created by the graphic interface and the relation objects constructed during the pixel analysis to recognize contact points.
- the TEXT level, consisting of objects containing copies of the input sentences.
- the PICTURE-MODEL level, where abstract objects constructed during parsing of the picture are placed.
- the TEXT-MODEL level, where structures created during the English language parsing are placed.
- the PROBLEM-MODEL level, where the final model is constructed. It contains objects based on information from one or both of the previous two levels, with coreferences resolved and incomplete properties filled in.

The PICTURE and TEXT levels correspond to the "concrete levels" of organizing coreferent knowledge bases mentioned earlier, and the PICTURE-MODEL and TEXT-MODEL levels to "intermediate levels." The PROBLEM-MODEL level is the most abstract.

Objects on the blackboard are represented as frames. In the notation of BB1, frame slots fall into two categories: *attributes*, which list an object's properties, and *links*, which are pointers to other blackboard objects. Attributes are used in BEATRIX to store specific information or descriptions of an object, while links are useful for showing its relationship to other objects in the knowledge base.

3.3 Knowledge sources

In a blackboard system, forward progress is controlled by *knowledge sources*, daemons that respond to changes on the blackboard, causing further changes to be made and possibly causing other knowledge sources to trigger. In BEATRIX, a special knowledge source called *Post-the-Problem* automatically triggers when the understanding module is initialized. Post-the-Problem creates blackboard objects from the English sentences and elements of the GLISP picture model, and places them on the TEXT and PICTURE levels of the blackboard, respectively. Other knowledge sources are programmed to trigger when these objects appear on the blackboard, and they cause new objects to be propagated up the higher blackboard levels.

BEATRIX knowledge sources (KSes) can be grouped according to the blackboard levels on which they operate. Members of the first class, the *identify* class, examine picture elements and try to identify which abstract objects they could represent. For example, a slanted line in the picture could be used to illustrate a taut rope, an inclined plane, or even part of an arrow representing a force. Identify-class KSes operate by triggering when a specific kind of object is added to the PICTURE level, making a hypothesis about the abstract object it could represent, and producing a new object on the PICTURE-MODEL level. The major fields of *Identify-Pulley*, a typical identify-class KS which recognizes pulleys in pictures, is illustrated in Figure 3. Note that an individual identify-class knowledge source does not attempt to make any definite identification of the trigger object, but serves only to post a suggestion on the blackboard for other KSes to notice. Each new object added to the PICTURE level may cause several knowledge sources to trigger if the picture element is used in such a way that it could represent more than one abstract object. In that case, additional knowledge in another KS is needed to resolve the ambiguity.

Movement from the TEXT level to the TEXT-MODEL level of the blackboard is achieved by the *parse-class* knowledge sources, which

```
trigger conditions:
       an object has been added to the PICTURE level
       that object is a CIRCLE object

local variables:
       THE-CIRCLE: set to the internal name of the trigger
                   object.
       THE-BRACKET: if there is a line touching the center
                   of the circle, set THE-BRACKET to the
                   internal name of that line; otherwise,
                   set it to NIL.

actions:
1.  Add a new PULLEY object to the PICTURE-MODEL level.
    Set its CLASS attribute = PULLEY
    Set its CENTER attribute = the center of THE-CIRCLE
    Make a PICTURE-ELEMENT link between it and THE-CIRCLE
2.  If there was a bracket saved on THE-BRACKET, mark
    the line representing it as matched and add a
    PICTURE-ELEMENT link to THE-BRACKET.
```

Figure 3: *Identify-Pulley*, an Identify-class knowledge source

handle the parsing of the English text. Most of the processing is done by a single knowledge source, *Parse-the-Sentences*, which when triggered runs an English language parsing module to handle all of the syntactic processing and some of the semantic. The parser is an augmented transition network (ATN) parser, and is written using the notation described by Charniak and McDermott in [7]. Many of the words contained in the input sentences cause a corresponding object to be placed on the TEXT-MODEL level, with modifiers such as adjectives and prepositional phrases forming structures stored within its attributes and links. Final semantic parsing is done by specialized parse-class KSes that perform tasks like "executing" verb actions or carrying out arithmetic operations, and which then modify the TEXT-MODEL frames accordingly. It would be an interesting project to try to replace the ATN parser with a set of knowledge sources that parse the English text directly.

The most interesting collection of knowledge sources comprise the *matching* group, those which produce objects actually belonging to the final model of the problem. Match-class KSes attempt to resolve coreferences between the PICTURE-MODEL and TEXT-MODEL levels, and coerce the two intermediate objects into one object on PROBLEM-MODEL. Because BEATRIX knowledge sources have been written using stepwise refinement, most of them start life as simple rules that state,"to find a match for an object on PICTURE-MODEL, find an object on TEXT-MODEL of the appropriate class and call them matched." Of course, this only works fine as long as there is exactly one of each object class in both the PICTURE-MODEL and TEXT-MODEL levels. As more complex problems are attempted that involve more than one instance of an object class, heuristic analysis of the objects must be done to find the best match. For each class of object, a function BEST-FIT-classtype must exist which operates on an object of that class on PICTURE-MODEL and a list of objects of corresponding class on TEXT-MODEL and returns the most likely candidate for matching based on the attributes of the objects.

Just as objects on PICTURE-MODEL and on TEXT-MODEL are considerably more complex than objects on PICTURE and TEXT, objects on PROBLEM-MODEL contain even more detail. PROBLEM-MODEL objects deal more closely with the roles that the physical objects play in the problem. A good example would be to compare relations on each level: on the PICTURE level, a TOUCH relation indicates that two graphic objects touch each other, and has as an attribute the coordinates of the contact point. On the PICTURE-MODEL level, a TOUCH relation indicates that two physical objects

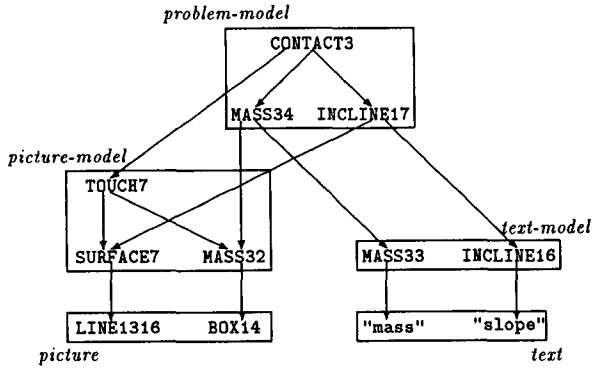


Figure 4: Part of a sample blackboard

touch each other, and has attributes describing what subpart of the objects (such as the "left end" of a rope) are in actual contact. Finally, on the PROBLEM-MODEL level, a CONTACT relation indicates that two objects in the problem come in contact, and has attributes describing the probable type of contact ("pinjoint" or "hinge", for example), the nature of the force transmitted through the contact (e.g. frictional or gravitational), and significant information about the contact forces (like the coefficient of friction).

Figure 4 represents a subset of a typical blackboard and illustrates how objects on each level are related to each other. `LINE1316` and `BOX14` are picture elements drawn by the user, and `"mass"` and `"slope"` are words contained in the English input text. (Individual words do not actually appear on the TEXT level of the blackboard as they do in the figure. The basic unit of the TEXT level is the sentence, but for simplicity, only the words relevant to the example are shown in the figure.)

When BEATRIX started up, *Post-the-Problem* placed (among other things) `LINE1316` and `BOX14` on the PICTURE level and the sentences on the TEXT level of the blackboard. These actions caused other KSes to trigger; in particular, identify-class KSes *Identify-Surface* and *Identify-Mass* have presented hypotheses about the line and the box. (Incorrect hypotheses are not shown in the figure.) Another identify-class KS, *Propagate-Touches*, used information from the picture analysis indicating that the line and the box come into contact in the picture, and concluded that the surface and mass must also touch.

At some point, the sentences are parsed, and a MASS object and an INCLINE object are placed on the TEXT-MODEL level by *Parse-the-Sentences*. Match-class KSes start to come into play, and matches are made between objects on the PICTURE-MODEL level and the TEXT-MODEL level. The TOUCH relation is carried upward to indicate contact between objects in the final model.

4 Conclusion

The use of a blackboard system for handling coreference has been quite successful. The data structure represents a natural way to organize the various levels of abstraction necessary for parsing language and pictures. In addition, the control structure is sufficiently flexible to allow for the handling of erroneous hypotheses and false interpretations of picture objects. The use of a blackboard system has provided a simple testbed for trying out new parsing rules and heuristics.

In BEATRIX's current state of development, the jury is still out on the question of the feasibility of picture grammars. For the set of test problems currently implemented, the number of picture objects which could be organized into a hierarchy is extremely small. Consequently, a parse tree of one of these problems would be extremely shallow, with high fan-out. It is hoped that as additional problems are analyzed, more suitable applications for picture grammars will emerge.

One problem that must be acknowledged in any final evaluation of BEATRIX is: how easy is it to modify BEATRIX so as to accommodate new problems with new features of physics, and what are the effects of modification on performance? Currently, it is not possible to project how well our stepwise-refinement approach to building on BEATRIX's knowledge sources will work as more test problems are accommodated. Although it is expected that modifications will be more difficult to handle as the number of knowledge sources increases, most of the "hard" modifications encountered up to now have involved either departures from our intended problem domain or convoluted special cases chosen specifically for the purpose of "stumping" the system. BEATRIX will hopefully be able to withstand a large range of typical problems upon completion.

Efficiency is another concern in evaluating BEATRIX. One way of "helping" a blackboard system arrive at a solution quickly is by *sequencing* the knowledge sources to influence their order of execution. (This may be done by having one KS produce a blackboard object or set a flag whose only purpose is to cause a second KS to trigger.) Because this is contrary to the opportunistic philosophy of BEATRIX and of blackboard systems in general, it is to be avoided. Instead, control heuristics will be used to a greater extent in helping the scheduler select the best KS. We plan to study the effects of these heuristics on BEATRIX's performance.

There are a great many directions for our research to continue in the future. Simple expansion within the domain of physics problem solving is one example: one could teach BEATRIX to handle other types of mechanics problems, or expand it to include the field of thermodynamics. Improving BEATRIX would also be an interesting direction to follow: an aesthetic weakness of its design is its asymmetry regarding treatment of text and pictures. Development of its two-phase system of parsing English would increase the symmetry and possibly provide insights for other types of information parsing. It would also be interesting to apply our approach to other types of knowledge beyond text and pictures, e.g. other forms of input, such as auditory data or output from a vision system. BEATRIX has been successful as a testing ground for picture parsing and analyzing the coreference problem. It is hoped that what can be learned from it will serve as a springboard for future research.

References

[1] Fu, King Sun. *"Syntactic Pattern Recognition and Applications."* Prentice-Hall: New Jersey, 1982.

[2] Novak, Gordon S. Jr. *"Computer Understanding of Physics Problems Stated in Natural Language."* Technical Report NL-30, Department of Computer Sciences, The University of Texas at Austin, 1976.

[3] Kook, Hyung Joon. *"Representation of Physical Models for Expert Physics Problem Solving."* Ph.D. Proposal, Department of Computer Sciences, The University of Texas at Austin, 1987.

[4] Novak, Gordon S. Jr. *"GLISP User's Manual."* Technical Report STAN-CS-82-895, Department of Computer Science, Stanford University, 1982.

[5] Sutherland, I. E. *"SKETCHPAD: A Man-Machine Graphical Communication System."* AFIPS: Proceedings of the Spring Joint Computer Conference, 1963.

[6] Hayes-Roth, Barbara. *"BB1: An Architecture for Blackboard Systems that Control, Explain, and Learn About Their Own Behavior."* Technical Report STAN-CS-84-1034, Department of Computer Science, Stanford University, 1984.

[7] Charniak, Eugene, and Drew McDermott. *"Introduction to Artificial Intelligence."* Addison-Wesley: Reading, Mass., 1985.