CIFE CENTER FOR INTEGRATED FACILITY ENGINEERING

Diagrammatic Reasoning for Qualitative Structural Analysis

By Yumi Iwasaki,
Shirley Tessler, and
Kincho H. Law

Technical Report
Number 94

May, 1994

STANFORD UNIVERSITY

If you would like to contact the authors please write to:

c/o CIFE, Civil Engineering,
Stanford University,
Terman Engineering Center
Mail Code: 4020
Stanford, CA 94305-4020

SUMMARY
CIFE TECHNICAL REPORT #94

Title: Diagrammatic reasoning for qualitative structural analysis
Authors: Yumi Iwasaki, Shirley Tessler, and Kincho H. Law
Publication Date: April 1994
Funding Sources:
- **Name of Agency:** Center for Integrated Facilities Engineering
- **Title of Research Project:** Pictorial Reasoning for Structural Analysis

1. Abstract:

This report describes a preliminary study and the development of prototype systems for diagrammatic reasoning in the domain of structural analysis. Diagrammatic reasoning is prevalent in human problem solving behavior, especially for problems involving spatial relationships among physical objects. Our research examines the relationship between diagrammatic reasoning and symbolic reasoning in analysis of simple frame structures. Preliminary evaluation shows that a diagrammatic reasoning capability provides an environment where inferences about the physical results of proposed structural configurations can take place in a more efficient as well as more intuitive manner than that possible through purely symbolic representations.

2. Subject:

For preliminary analysis of structures, human engineers often employ diagrams as a visual language to study and to gain intuitive understanding about the behavior of structures. This paper reports on a preliminary study and the development of prototype systems that reason with diagrams to solve structural analysis problems to better emulate the intuitive visual problem solving techniques of human engineers.

Diagrammatic reasoning is a type of reasoning in which the primary means of inference is the direct manipulation and inspection of a diagram. Humans often use diagrams not only to solve problems but also to decide how to do so most efficiently. Our research examines the relationship between diagrammatic reasoning and symbolic reasoning in a concrete problem solving context of determining the deflection shape of a frame structure under a load. We have built two systems called REDRAW-I and REDRAW-II, which draw diagrams and manipulate them in order to determine the deflection shape just as a human engineers do when they solve such problems qualitatively.

A program that is capable of reasoning qualitatively with diagrams are much more understandable to humans than comparable programs that reason only with purely symbolic representations when reasoning about spatial properties is essential part of the problem. Therefore, such programs would be much more suitable for helping students acquire intuitive understanding of the phenomena being analyzed, e.g. the behavior of a frame structure under a load in our case. Furthermore, a diagrammatic reasoning program may be much more efficient because the spatial information explicit in a diagram can be used to control the system's reasoning process effectively.

3. Objectives/Benefits:

Although diagrams play an important role in human reasoning about physical structures, little attempt has been made to capture the power of diagrammatic reasoning in a problem solving system. Existing CAD systems can be used to draw a diagram but they can neither relate the drawing to abstract concepts nor reason with the diagram as humans do. Conventional AI systems can reason with abstract concepts but cannot relate them to what an engineer actually sees. A technology to make a computer reason with diagrams just as human engineers do will make systems much more understandable to a human user, and in some cases, more computationally efficient than purely symbolic systems. A multi-purpose, diagrammatic reasoning tool can greatly facilitate construction of problem solving systems in domains that require reasoning about spatial information. Structural analysis is just one example of such problems. Studying the role of diagrams in a problem solving process will also help us understand how to train engineering students effectively in visualization skills.

4. Methodology:

The project focused on constructing programs that can solve problems through drawing and manipulating diagrams. We chose determination of deflection shape as the problem, because diagrams play an important role when humans solve this type of problems.

5. Results:

We constructed two programs, REDRAW-I and REDRAW-II, both of which solve the same type of deflection shape problems qualitatively. REDRAW-I takes a more informal approach than REDRAW-II, but they both draw diagrams and solves problems through inspection and manipulation of the diagrams.

REDRAW systems appear to be much more effective in helping the user understand the behavior of frame structures under a load and learn how to solve problems of this type compared to a comparable system that solves the same type of problems purely symbolically. In addition, they are more efficient, mainly due to the fact the spatial information explicit in the diagrams can be used to focus the problem solving process.

6. Research Status:

We have successfully obtained a research planning grant from NSF to continue work on this subject. We are using the grant to work on the following tasks as well as to prepare a proposal for a larger grant.

- We are in the process of designing a general-purpose diagrammatic representation and manipulation shell. We plan to use the shell to implement problem solving systems for a variety of different tasks requiring use of diagrams.

- We are also working on a formal characterization of a diagrammatic representation in terms of its information content and the types of inference sanctioned by the representation. This is essential in elucidating the role diagrams play in problem solving, given a problem and a particular type of diagrams used.

Diagrammatic Reasoning for Qualitative Structural Analysis

Yumi Iwasaki*, Shirley Tessler*, and Kincho H. Law**

***Knowledge Systems Laboratory, Stanford University, Stanford, California, U.S.A.**
****Civil Engineering Department, Stanford University, Stanford, California, U.S.A.**

Abstract

For preliminary analysis of structures, human engineers often employ diagrams as a visual language to study and to gain intuitive understanding about the behavior of structures. This paper reports a preliminary study and the development of a prototype system for diagrammatic reasoning to better emulate the intuitive visual problem solving techniques of human engineers.

Diagrammatic reasoning is a type of reasoning in which the primary means of inference is the direct manipulation and inspection of a diagram. Diagrammatic reasoning is prevalent in human problem solving behavior, especially for problems involving spatial relationships among physical objects. Our research examines the relationship between diagrammatic reasoning and symbolic reasoning in a computational framework. We have built a system called REDRAW, which emulates the human capability for reasoning with pictures for qualitative analysis of simple frame structures. Diagrammatic representations provide an environment where inferences about the physical results of proposed structural configurations can take place in a more intuitive manner than that possible through purely symbolic representations.

ACKNOWLEDGMENT

This work is sponsored by the Center for Integrated Facilities Engineering and is conducted at the Knowledge System Laboratory at Stanford University.

1. INTRODUCTION

The traditional approach to the study of structural analysis has been based, almost exclusively, on quantitative (i.e. numerical) methods. Numerical values to loads and dimensions are needed to determine the numerical value of reactions and bending moments as well as to proportion the size of the members. Numerical analysis of structures, particularly statically indeterminate structures, requires that the size of all the structural members be specified before an analysis can be carried out. There is, however, an important step before that numerical analysis can take place: the preliminary analysis and design phase where the schematic of the structure is being defined and rough behavior of the structure is being studied. The detailed, numerical analysis is intended to be a check on the preliminary analysis and design of the structure.

Preliminary analysis requires a quite different set of techniques to determine the relationship between the load and the resulting behavior of the structure based on "qualitative", i.e. non-numerical, information. Qualitative analysis plays a significant part in the understanding of structural behavior and the overall design checking procedures, which must be constructed to ensure the correct use of computer modeling and numerical analysis programs.

There have been many investigations on extending the methodologies developed in qualitative physics research [Iwasaki 1989; Weld and Kleer 1990] to the problem of qualitative structural analysis [Slater 1986; Fruchter, Law and Iwasaki 1991; Roddis and Martin 1991]. Though all these works focused on symbolic and/or mathematical modeling of structures, human engineers often rely on sets of coherent diagrams rather than mathematical models in preliminary analysis. In many engineering problems, drawing a diagram is a crucial step in the problem solving process. Drawings often reveal important information that may not be explicit in a written description, and can help one gain insights into the nature of the problem. In our work, we attempt to explore the use of diagrammatic reasoning for qualitative analysis of structures.

Our work is aimed towards understanding the role of diagrammatic reasoning in engineering problem solving. In this study, we explore the potential of diagrammatic reasoning in determining the deflection shape of a building frame structure under load. We have constructed a computer program called REDRAW (Reasoning with Drawings) that solves this problem qualitatively using a diagram in a way similar to human engineers. Our hypothesis is that since humans reason with so much apparent ease with diagrams, a program that could reason directly with a diagrammatic representation would be more understandable to the user than a program that reasons exclusively with a purely symbolic representation of the same information. Such a program may also be useful in imparting visualization skills to students of engineering disciplines.

An advantage of the deflection shape problem for studying the role of visual reasoning in problem solving is the fact that it is rich with domain-specific knowledge that has significant implications on how the diagram is manipulated and interpreted. Another possible domain in which to study diagrammatic reasoning is geometry, where pictures are abstract diagrams without being a representation of anything in the world. However, in geometry, the only property one reasons about is the geometric property. There are no other types of information, apart from that represented in the diagram, that one must take into account when manipulating and inspecting the diagram.

In contrast, pictures used for reasoning in engineering design are not simply abstract geometric shapes but actually represent things in the real world. Furthermore, how a picture is interpreted and manipulated depends significantly on what it represents. For example, a line in our domain represents a beam or a column. Changing the length of the line would change the information represented by the diagram. In other types of diagrams, such as a circuit diagram, one could change the length or curvature of the line representing an electrical connection without changing the informational content of the diagram. For the goal of better understanding the role of visual reasoning in problem solving and its relation to symbolic reasoning, it is important to work with a problem requiring a wealth of domain knowledge that has significant influence on the way diagrams are used and interpreted.

This paper is organized as follows: In the remainder of this section, we define diagrammatic reasoning and discuss its role in problem solving in general. We, then, discuss related work in qualitative reasoning about physical systems and reasoning with images. Before going on to describe REDRAW, we briefly review the problem of deflection of frame structures subject to load. In Section 2 and 3, we describe the architecture of the two implementations of REDRAW, REDRAW-I and -II, in detail. Section 4 concludes with a summary and a discussion of future work.

1.1 DIAGRAMMATIC REASONING AND THE ROLES OF DIAGRAMS IN PROBLEM SOLVING

The goal of symbolic or diagrammatic reasoning programs is to make inferences by manipulating and inspecting the internal representations of information of the domain. A symbolic reasoning program makes inferences through a purely descriptive representation of the knowledge of the domain and the problem itself. A diagrammatic reasoning program, on the other hand, represents at least some of the information, especially geometric information, in a more depictive form, i.e. in a form that reflects the geometric and topological structure of what is represented more directly than a purely descriptive form.

We define diagrammatic representation not only in terms of the distinction between the depictive and the descriptive but also in operational terms, i.e. what types of operations on the data structure are allowed and how they are used by the program. A diagrammatic reasoning program performs at least part of its inferences using data structures with the following characteristics:

1. The information represented explicitly in the data structure is the type of information that is explicit in diagrams (i.e. geometric or otherwise visual information) and that is detected easily by human visual inspection.

2. The operations that are permitted on the data structure are those that humans perform easily with diagrams. Such operations can include both visual inspection operations as well as manipulation -- through either mental imagery or through actual modification of the drawing.

In contrast with diagrammatic reasoning programs, current symbolic reasoning programs use only symbolic forms of representation such as logic, frames, semantic nets, etc. An important difference between a symbolic representation and a diagrammatic one is that the information represented explicitly in a symbolic program is not necessarily what is explicit in a picture. Furthermore, reasoning is performed through some inference rules, which do not necessarily reflect the types of inferences humans make with an image.

Computer graphics has played a significant role in the advancement of computer aided design. However, graphics programs are used primarily to produce pictures, not to perform reasoning tasks. Even when the picture produced represents an engineering entity instead of a purely abstract geometric shape, the program itself has no knowledge of nor does it reason about what is depicted. Graphics programs make no attempt to interpret or reason with the pictures produced. It is usually the user (the human viewer of the pictures) who interprets and sometimes reasons about the pictures.

A salient feature of diagrammatic reasoning in many situations is its qualitativeness. People reason with diagrams to get rough, qualitative answers. If a more precise, quantitative answer is needed, they must resort to more formal, mathematical techniques. However, qualitative techniques are extremely useful in gaining valuable insight into the range of possible solutions. An initial qualitative understanding thus obtained can guide the later analysis by constructing appropriate (numerical) models and employing appropriate analysis methods.

In order to develop an intuitive understanding of the response of the structure under a load, we find that diagrams fulfill many of the same roles as those articulated by researchers in other fields. First, diagrams are used as "a visual language of structural behavior that can be understood with the minimum of textual comments" [Brohn 1984]. The language allows the engineer to express explicitly the constraint or physical law that is relevant at each part of the proposed structure, in such a way that the constraints and some of the consequences are immediately apparent to the reader without further reasoning. Furthermore, the diagram serves as a place holder or short-term memory device by allowing the designer to sketch out the result of one deformation and then go back to see if there is a further effect or interaction that needs to be addressed. Finally, visual inspection of diagrams can serve as a means to guide the engineer to choose a more efficient problem solving method than she might otherwise.

1.2 RELATED WORK

There have been many investigations on extending the methodologies developed in qualitative physics research to the problem of qualitative structural analysis [Slater 1986; Fruchter, Law and Iwasaki 1991; Roddis and Martin 1991]. We have previously built a program called QStruc to solve the same deflected shape problem described in this paper, but using a traditional, symbolic AI approach[Fruchter, Law and Iwasaki 1991]. The program determines the qualitative values of forces, moments, and displacements in a frame structure under a load. The inputs to the system are a symbolic representation of the structure in terms of its members and connections, and a load on the structure. There is no explicit representation of the shape of a structure in the program. The shape is implicitly represented by the existence of such physical processes as bending, and the qualitative values (positive, negative, zero or unknown) of such parameters as displacements. QStruc has successfully analyzed several simple two-dimensional structures, thus demonstrating the feasibility of performing qualitative analysis of structures on a computer. However, our experience with QStruc shows us that it does not help an engineer to gain an intuitive understanding of the deflection process because its solution strategy of setting up all applicable equilibrium equations and solving all of them reflects neither the way humans usually solve the problem nor the causal process through which the load makes the structure deform.

Research has been reported in cognitive psychology as well as artificial intelligence on the roles of diagrammatic reasoning in human problem solving. Larkin and Simon discussed extensively the advantages of diagrams for facilitating inference about topological or geometric relationships [Larkin and Simon 1987]. One important advantage of diagrammatic representation is that it makes explicit the spatial relationships that might require extensive search and numerous inference steps to detect using a symbolic representation [Larkin and Simon 1987]. Chandrasekaran and Narayanan [Chandrasekaran and Narayanan 1990], Novak and Bulko [Novak and Bulko 1992], Borning [Borning 1979] and others have also pointed out the usefulness of diagrams to human problem solvers as a device to aid in visualization, "gedanken experiments" or prediction. Chandrasekaran and Narayanan proposed a visual modality-specific architecture, using a visual representation scheme, consisting of symbolic representations of the purely visual aspects (shape, color, size, spatial relations) of a given situation at multiple levels of resolution [Larkin and Simon 1987]. Their objective is "to propose a cognitive architecture underlying visual perception and mental imagery that explains analog mental imagery as well as symbolic visual representations" [Larkin and Simon 1987]. Lindsay uses constraint maintenance techniques to manipulate a diagrammatic representation to make inferences and test conjectures in qualitative geometric reasoning [Lindsay 1992]. His goal is to demonstrate that a combination of propositional and pictorial representations offers more psychologically plausible and computationally efficient ways of reasoning about mathematical problems. Novak and Bulko [Novak and Bulko 1992], Koedinger [Koedinger and Anderson 1990] and others have explored the idea that diagrams may sometimes be used not primarily for making base-level inference, but rather to help in the selection of an appropriate method to solve a problem; that is, as an "aid in the organization of cognitive activity" [Chandrasekaran, Narayanan and Iwasaki 1993].

In Lindsay's research in qualitative geometric reasoning [Lindsay 1992], he has developed a computational model of human visual reasoning in the domain of plane geometry. Lindsay uses constraint maintenance techniques to manipulate a diagrammatic representation to make inferences and test conjectures. His goal is to demonstrate that a combination of propositional and pictorial representations offers more psychologically plausible and computationally efficient ways of reasoning about mathematical problems.

The work by Forbus and his colleagues on FROB [Forbus 1980] and the CLOCK project [Forbus, Nielsen and Faltings 1991] is aimed at automating the qualitative spatial reasoning process. They also take a multi-faceted approach to representing information about shape. They represent the detailed metric information about the shape in a Metric Diagram. The Metric Diagram is used to compute a more qualitative, symbolic description called a Place Vocabulary. In their systems, a Metric Diagram, which represents precise geometric information explicitly, is used to compute a symbolic representation using the Place Vocabulary, by most of the reasoning itself takes place only using the Place Vocabulary.

In the modality-specific architecture proposed by Chandrasekaran and Narayanan [Chandrasekaran and Narayanan 1990], the visual representation is linked to an underlying analogical representation of a picture so that visual operations performed on the analogical representation are immediately reflected on the visual representation and vice versa. The architecture of REDRAW is greatly influenced by the ideas of Chandrasekaran & Narayanan [Chandrasekaran and Narayanan 1992] as well as those of Kosslyn [Kosslyn 1980], regarding human cognitive architecture, in which they argue that some types of reasoning are tightly coupled with perception. This idea of "perceptually grounded reasoning" is reflected in the architecture of REDRAW, which consists of symbolic and diagrammatic layers that are closely coupled. Furthermore, the problem solving approach of REDRAW is designed to mimic the qualitative structural analysis method of human engineers.

1.3 THE DEFLECTION PROBLEM OF FRAME STRUCTURES

In this section, we explain the deflection shape problem, including the diagrammatic and symbolic components of the solution process. Determining the qualitative deflected shape of a frame structure under a load is one of the fundamental steps in analyzing and understanding the behavior of a structure. Engineers first sketch a simple, 2-D drawing of the shape of the given frame structure. Given a load on the structure, they modify the shape of the structural member under the load. They inspect the modified shape to identify the places where constraints for equilibrium and geometric compatibility conditions of the structure are violated. Those constraint violations are corrected by modifying the shape of connected structural members, propagating deflection to other parts of the structure. This process is repeated until all the constraints are satisfied. The drawing thus produced shows the final deflected shape of the frame under the given load.

Figure 1 illustrates an example of this type of reasoning process. (a) shows the given frame structure. Under the load, *B1* deflects in the same direction as the load as shown in (b). Since *J1* and *J2* are rigid joints, they must maintain a 90-degree angle. Inspecting the shape (b) shows that they are not. To make them 90 degrees, the columns *C1* and *C2* are rotated around the joints as shown in (c). However, since the supports of *C1* and *C2* do not allow displacement, the columns must be bent to keep the ends fixed as shown in (d). Furthermore, since the supports of *C1* and *C2* are rigid supports, the lower portion of the columns must remain perpendicular to the ground as shown in (e). Inspection of the shape (e) shows that the moment equilibrium around *J1* is not satisfied because both members connected by *J1* are deflected clockwise indicating the sum of the moments around the joint to be non-zero, violating the moment equilibrium condition. The same can be said for *J2*, also. Finally, both of the end portions of *B1* are bent upwards slightly in order to achieve moment equilibrium around *J1* and *J2* as shown in (f).

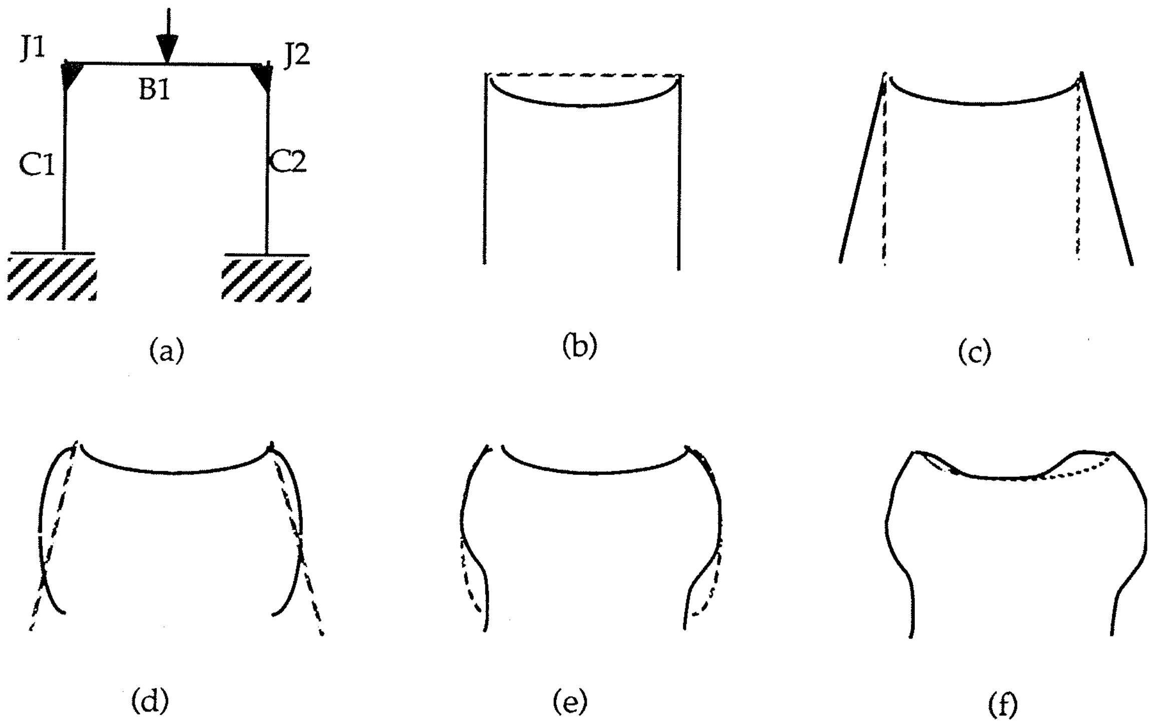


Figure 1: Steps in determining the deflected shape

The above example of problem solving process involves a representation, such as a diagram, that reflects spatial information much more directly than conventional sentential representations as well as manipulation of such a representation to make inferences. It also involves use of non-visual knowledge. Another important feature of the problem solving process is that the reasoning carried out is qualitative. The answer produced in this case is a picture of a deflected shape. Although the resulting picture is qualitatively consonant with the problem solution, it is not, nor does it need to be, mathematically precise or to scale.

This type of problem can also be solved by a more formal, mathematical technique involving setting up equations for forces, bending moments and deflection. However, even when the more formal method is used, visualization is an indispensable first step that provides an engineer with an intuitive understanding of the behavior of the structure and enables her to recognize a good strategy for further analysis.

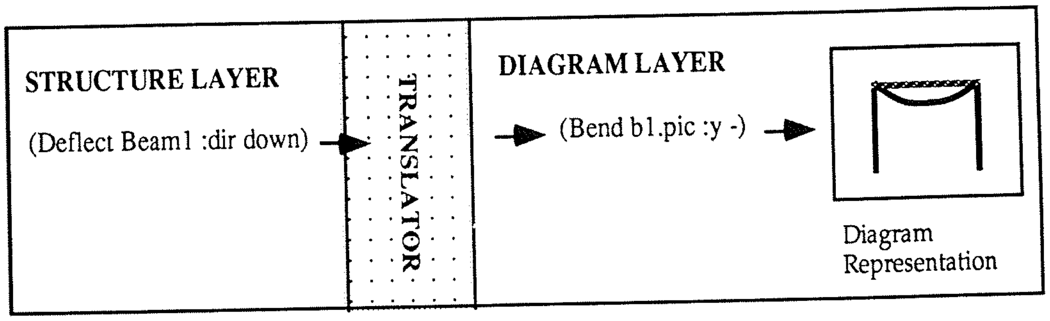
We set out to build a computer program that can reason about the deflection shape problem using a diagram just as a human engineer would. Our first implementation of such a program, REDRAW-I solves this type of deflected shape problems by directly manipulating a representation of the shape in the manner shown in Figure 1. Our second implementation, REDRAW-II, solves the same problems by a more systematic formal approach, computing the forces, bending moments, and the deflection shape, while also using diagrams. We will describe the two implementations in detail in the following sections.

2. REDRAW-I

REDRAW-I solves the deflected shape problem, following the same solution process outlines in Figure 1. Given a diagram of a frame structure and a load, REDRAW-I produces the underlying symbolic model in order to facilitate reasoning about non-diagrammatic concepts. Then, the program uses its structural engineering knowledge to propagate constraints on the diagram of the structure, inspecting and modifying this picture until a final shape is produced that represents a stable deflected structure under the given load as illustrated in Figure 1. REDRAW-I directly manipulates a representation of the shape in the same manner as depicted in the steps shown in the figure. As with the qualitative nature of human visual reasoning, the reasoning carried out by REDRAW-I is also qualitative. The answer it produces is a picture of a deflected shape.

2.1 SYSTEM ARCHITECTURE OF REDRAW-I

From examining the way deflection shape problems are solved by humans, it is apparent that solving this type of problem requires not only an ability to manipulate and inspect diagrams but also substantial structural engineering knowledge. Structural engineering knowledge about the properties of various types of joints and supports is necessary to identify the constraints applicable to the state (shape) of the structure. Such knowledge is best represented and manipulated symbolically. On the other hand, information about the shapes is best represented as a picture. Many types of modification and inspection of the shape are also more easily carried out with a picture.



Structure Layer
- Objects: beams, columns, connections, supports, load, etc.
- Operators: generate-force-equilibrium-conditions, generate-moment-equilibrium-conditions, etc.

Diagram Layer
- Objects: lines, splines, circles
- Operators:
  - Manipulation: rotate, bend, translate, smooth, etc.
  - Inspection: get-angular-displacement, get-displacement, symmetrical-p, etc.

Figure 2: Two-layered architecture of REDRAW

The requirement for both pictorial and non-pictorial representation and reasoning suggests a layered architecture. Figure 2 shows the architecture of REDRAW-I schematically. Thus, REDRAW-I includes both symbolic reasoning and diagrammatic reasoning components. The former contains the symbolic representation of the structural components and the structural engineering knowledge about various types of structural members, joints, supports, and the constraints they impose on the shape. It also includes a rule-based inference mechanism to make use of the knowledge. The latter, diagrammatic reasoning component includes an internal representation of the two-dimensional shape of the frame structure as well as a set of operators to manipulate and inspect the shape. These operators, some of which are shown in

Figure 2, correspond to the manipulation and inspection operations people perform frequently and easily with diagrams while solving deflected-shape problems.

The REDRAW-I system was developed using KEE[1], a Lisp-based object-oriented knowledge engineering development environment. KEE is a flexible environment that supports graphics for simple two-dimensional drawings as well as multiple inheritance, objects, methods and active values for changing slot values programmatically.

2.1.1 The Structure Layer

The Structure Layer contains a symbolic representation of the domain objects as well as the domain-specific knowledge. It stores non-visual information (such as that a hinged joint can rotate while a rigid joint cannot), various types of structural members, equilibrium conditions, as well as heuristic knowledge for controlling the structural analysis process. The classes of engineering objects of the domain (namely, objects representing beams, columns, supports, loads and structures) are arranged in a class-subclass hierarchy. The instances of such classes themselves contain information about its connected neighbors as well as about sub- and super-components, forming a partonomic hierarchy. All of the symbolic objects also contain a pointer to their pictorial object counterparts.

2.1.2 The Diagram Layer

The Diagram Layer represents the two-dimensional shape of a structure. There are several operators that directly act on this representation to allow inspection as well as transformation of the shape. These operators correspond to the operations people perform easily with diagrams. The internal representation of a shape is a combination of a bitmap whose elements correspond to each "point" in a picture, and a more symbolic representation where each line is represented by a set of x-y coordinates.

The Diagram Layer is independent of the structural engineering domain in the sense that it does not contain any structural engineering concepts. The basic objects are graphical primitives such as lines, splines and circles. However, the types of both manipulation and inspection operators provided for the layer do reflect the requirements of the domain. For example, the assumption that the frames consist of incompressible members made a particular set of operators necessary (e.g. the program requires a bend operator but not a stretch or compress

[1] KEE is a registered trademark of IntelliCorp Inc.

operator). The effects of the reasoning mechanisms also depend on the specific functioning of those required operators (for example, the bend operator creates a moderate curve rather than a complete bend that would cause the line endpoints to touch or cross; or, the inspect operator may look at components connected to the component in question, but will not compare that component to any other, as it might in some other domain.) The objects contain information only about their current shape and position with respect to other pictorial elements in the graphics window.

Beam and column objects are implemented as splines at the picture level; that is, the set of coordinate points in the object's point slot are connected by a curve. Besides such splines that depict the shape and, therefore, are actually used for reasoning, there are other iconic objects at the diagram layer depicting such things as load and different types of joints and supports. For example, fixed joints and supports and pinned supports are depicted with polylines, hinged joints are shown with circles, and roller supports are a combination of polylines and circles. The load objects are simple lines with an arrow at one end to show the direction of the load on a component. These iconic entities are only used to indicate specific component types to the user when the structure is actually drawn on the screen.

The relationships among the pictorial objects are also quite straightforward. The objects relate to each other in qualitative spatial terms such as connected-to, near, left, right, above and below. Moreover, only those primitive geometric properties that are easily identified by visual inspection rather than by reasoning involving multiple steps are used in the process of determining the deformation shape of a structural component. Such properties include whether two lines are approximately parallel and whether the angle between them is acute, obtuse or right angle. The pictures are not drawn in precise proportion. Only such information as approximate relative size, shape and proximity are used to draw them.

The Diagram Layer operators affect the position of a graphical object as well as its shape by making changes to the coordinates in the object's position and the list of points depicting the objects. Making changes to these coordinate values cause the picture object to be redrawn immediately in the KEE graphics window. For example, a spline with y-coordinate points of all zeros describes a straight beam. By replacing these zero y-values with appropriate negative numbers, the spline will be redrawn as a downward curve to represent a beam under a direct point load. Picture object points can be replaced individually or by equations, and both are done in this implementation. An important point to recall is that the drawing of the curves do not need to be precise; they need only be approximately correct and "look right" to a user. The system doesn't need any more precision than the user would require as if the same picture is drawn on the back of an envelope. The system needs only to know the basic direction and shape of the curves in order to make inferences about the shape and stability of a particular frame structure under load.

2.1.3 Linking the Structure and Diagram Layers

There is a close link between the information in the two layers. Communication between the two layers takes place by sending commands and posting constraints by the Structure Layer, which is carried out or checked by the Diagram Layer. There is a translator between the two layers to mediate the communication between the two layers as shown in Figure 2. The system relates the representation of a particular beam in the Structure Layer to a spline in the Diagram Layer, and the concept of deflection of a beam to an operation on a spline to transform its shape. Likewise, the system is able to identify features of a shape (e.g. direction of bending, existence of an inflection point) and to communicate them to the Structure Layer. When the Structure Layer posts a constraint or a command, the Translator translates it into a call to a Diagram Layer operator that can directly act on the representation of the shape to manipulate or inspect it. The result is again translated back to concepts that the Structure Layer understands.

2.2 EXAMPLES

In this section, we illustrate the problem solving process by REDRAW-I with two simple frame analysis examples.

2.2.1 Example 1

In the first example, we illustrate the type of communication that takes place between the Structure Layer, the Diagram Layer, and the Translator, which we will denote as *S*, *D*, and *T*, respectively. Given the frame structure of Figure 3(a), with a load, *LOAD3*, placed on the middle of the beam, *S* sends a command, "Deflect *BEAM3* in the same direction as the load," which *T*, translates into an operation "Bend *BEAM3.pic* in the negative direction of the y-coordinate", where *BEAM3.pic* is the label on the spline showing the shape of *BEAM3*. Carrying out this operation will result in the shape shown in Figure 3(b).

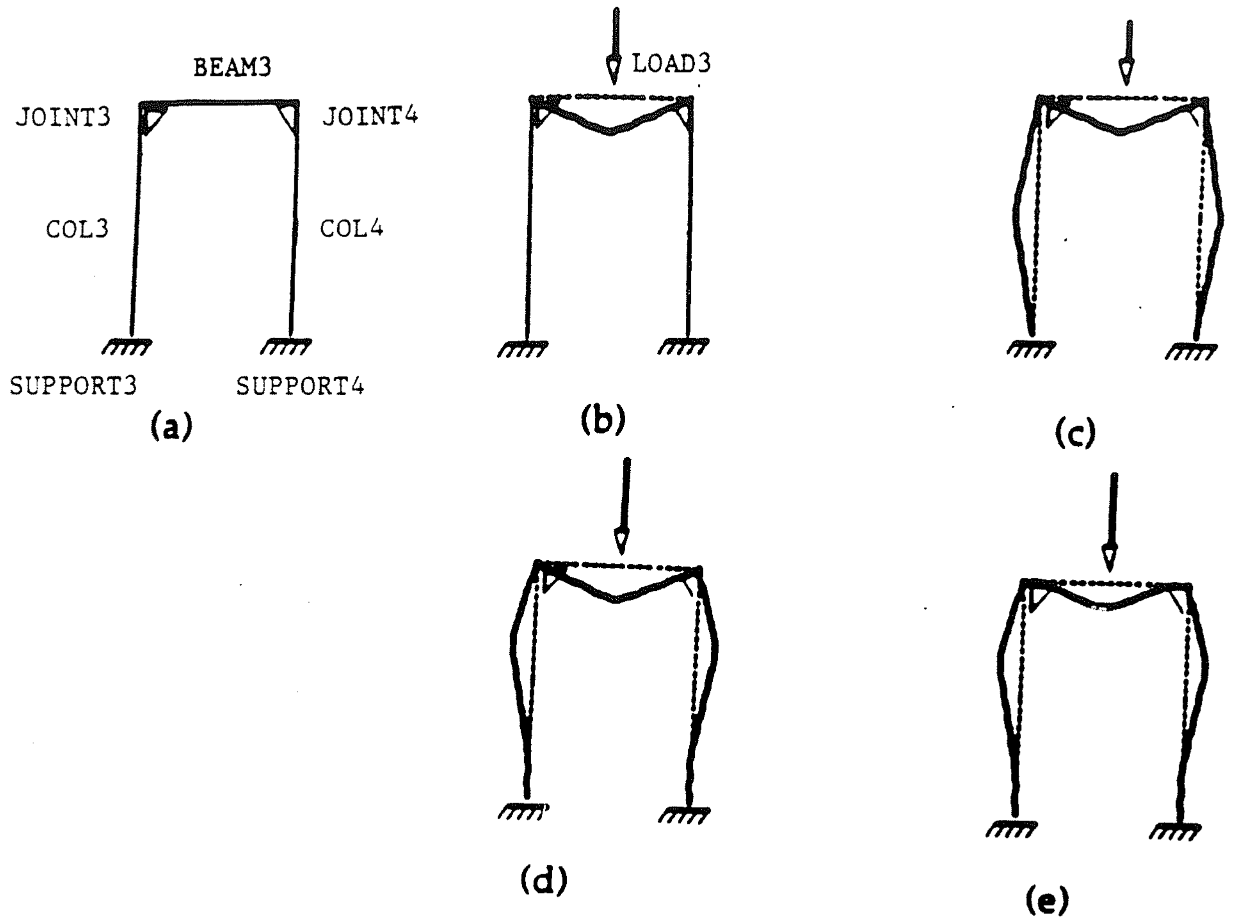


Fig. 3: Solution Sequence by REDRAW-I - Example 1

Continuing the interpretation process, *S* infers that since *JOINT3* is a rigid joint, *BEAM3* and *COL3* must maintain the same angle, i.e. perpendicular to each other at *JOINT3*, before and after application of the load. *S* issues a query to test this constraint. The query is translated into "get the angle between *BEAM3.pic* and *COL3.pic* at the ends connected by *JOINT3.pic*" for *D*. The answer, the actual angle between the two lines, is communicated to *S* as the answer that the constraint is not satisfied. *S* now issues a command to satisfy this constraint while keeping *BEAM3* fixed, which is translated into "make the angle between *BEAM3.pic* and *COL3.pic* at *JOINT3.pic* be 90 degrees without modifying *BEAM3.pic*" for *D*. REDRAW-I follows the same line of reasoning for *COL4*, also. Carrying out these operations will result in the shape shown in Figure 3(c).

Communication between the Structure and the Diagram Layers continues until all the constraints are satisfied. The results are depicted as shown in Figures 3(d) and 3(e). Figure 4 summarizes the symbolic reasoning activities carried out by REDRAW-I for this example.

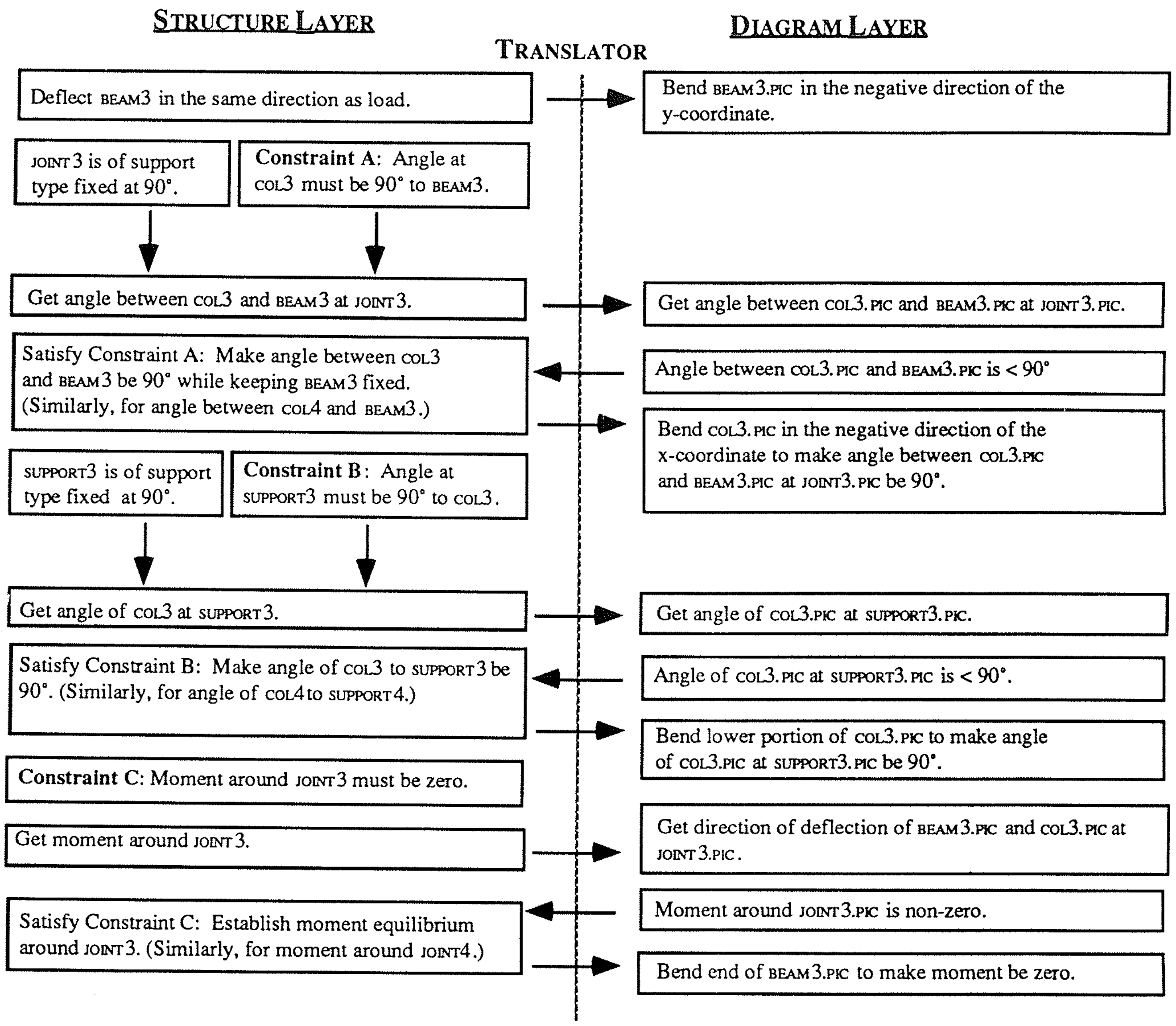

STRUCTURE LAYER
TRANSLATOR
DIAGRAM LAYER
Deflect BEAM3 in the same direction as load.
Bend BEAM3.PIC in the negative direction of the y-coordinate.
JOINT3 is of support type fixed at 90°.
Constraint A: Angle at COL3 must be 90° to BEAM3.
Get angle between COL3 and BEAM3 at JOINT3.
Get angle between COL3.PIC and BEAM3.PIC at JOINT3.PIC.
Satisfy Constraint A: Make angle between COL3 and BEAM3 be 90° while keeping BEAM3 fixed. (Similarly, for angle between COL4 and BEAM3.)
Angle between COL3.PIC and BEAM3.PIC is < 90°
Bend COL3.PIC in the negative direction of the x-coordinate to make angle between COL3.PIC and BEAM3.PIC at JOINT3.PIC be 90°.
SUPPORT3 is of support type fixed at 90°.
Constraint B: Angle at SUPPORT3 must be 90° to COL3.
Get angle of COL3 at SUPPORT3.
Get angle of COL3.PIC at SUPPORT3.PIC.
Satisfy Constraint B: Make angle of COL3 to SUPPORT3 be 90°. (Similarly, for angle of COL4 to SUPPORT4.)
Angle of COL3.PIC at SUPPORT3.PIC is < 90°.
Bend lower portion of COL3.PIC to make angle of COL3.PIC at SUPPORT3.PIC be 90°.
Constraint C: Moment around JOINT3 must be zero.
Get moment around JOINT3.
Get direction of deflection of BEAM3.PIC and COL3.PIC at JOINT3.PIC.
Satisfy Constraint C: Establish moment equilibrium around JOINT3. (Similarly, for moment around JOINT4.)
Moment around JOINT3.PIC is non-zero.
Bend end of BEAM3.PIC to make moment be zero.

Figure 4: Illustration of the inter-layer communication of REDRAW

2.2.2 Example 2

The second example is a simple frame structure with a hinge as shown in Figure 5. A point load is placed on the middle of the beam. In this example, we provide the details for the object representation and the execution of REDRAW-I. As noted earlier, once the frame structure is

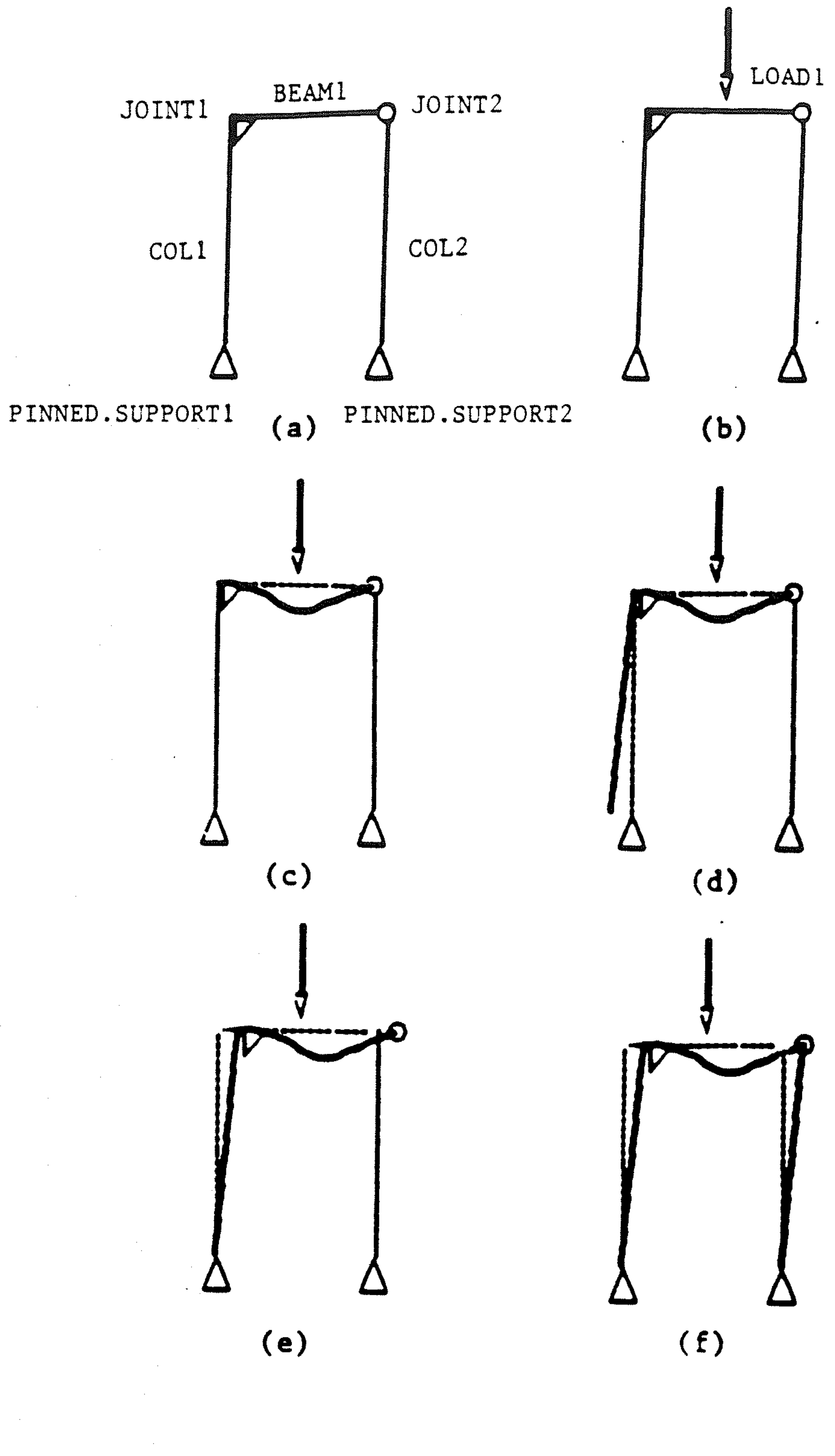


Figure 5: Solution Sequence by REDRAW-I - Example 2

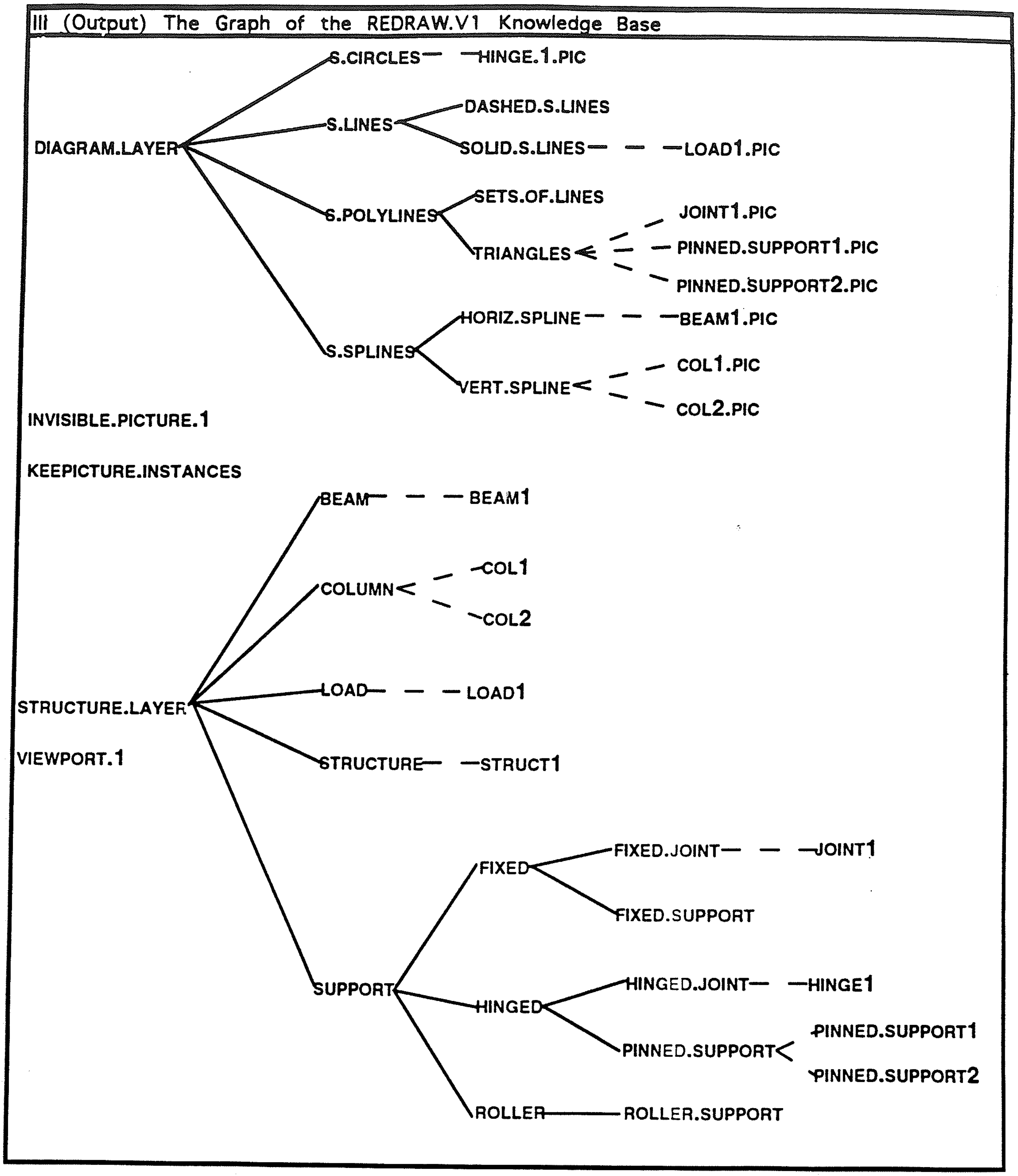


Figure 6: Object hierarchy of REDRAW-I

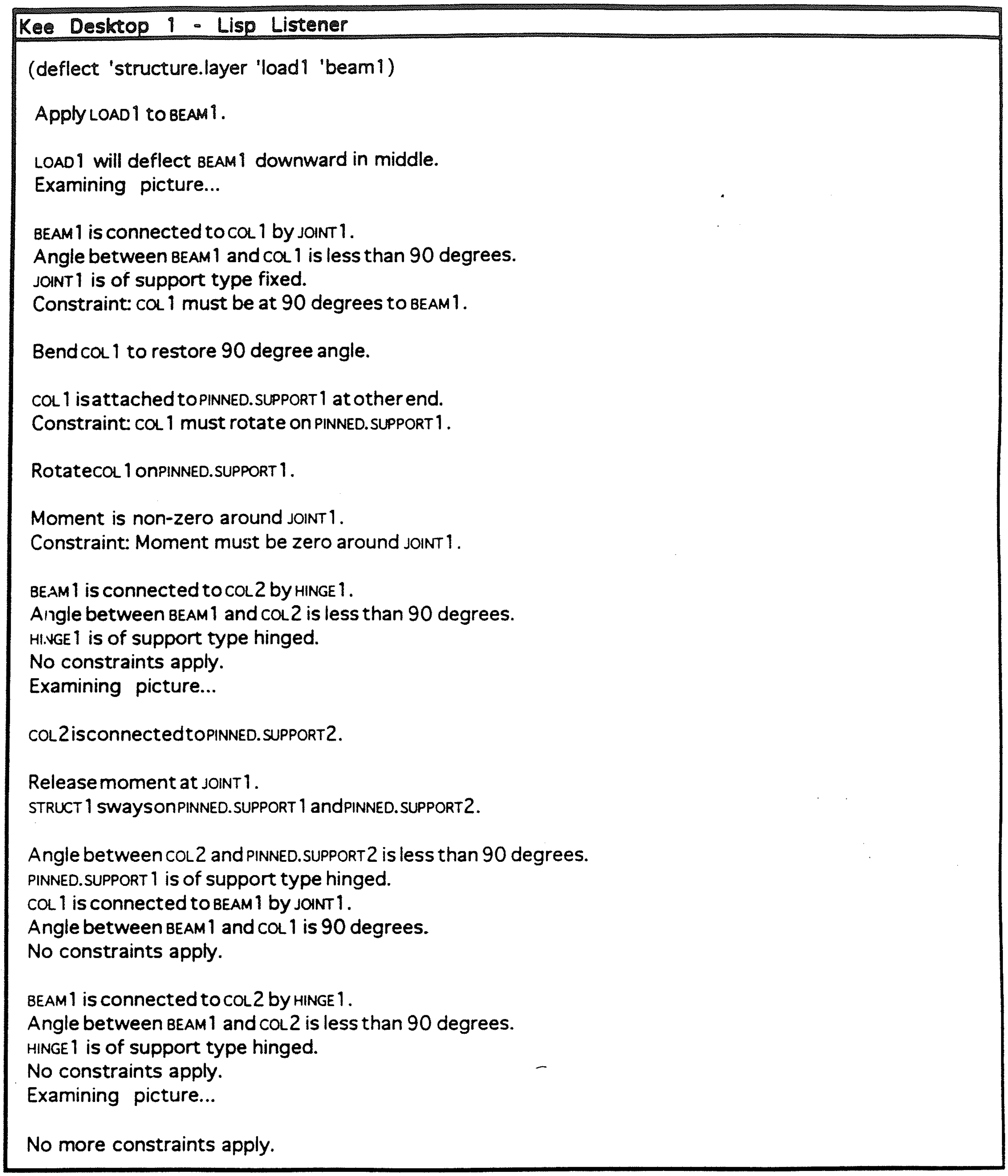
In this example, the Diagram Layer first examines the load arrow for the direction of the load and it checks the coordinates of the arrow relative to the beam to determine that the load is placed approximately in the center of the beam. REDRAW-I will immediately call the bend operator in the Diagram Layer to redraw the picture of the beam so it bends under load in the direction of the load. The bending of the initial component on a structure compels the program to examine the picture and try to determine the effects of that bend on the rest of the structure.

After the structural component, *BEAM1,* upon which the load was placed has been examined, the system proceeds to investigate all of the components connected to the loaded component, and so on. The examination will be in terms of the structural constraints that are associated with the different components. Most of the constraints consist of knowledge concerning the various support conditions and their resulting reactions. For example, the load on the beam is constrained by the fact that the angle at the fixed joint must remain at its original 90 degrees. The Structure Layer sends a message to the Diagram Layer to determine the angle at the joint after bending. In the Diagram Layer, the *check.angle* operator is defined to measure the angle between the column and the tangent of the deformed beam. The operator then replies that the angle at the fixed joint is currently less than 90 degrees. The Structure Layer then issues a command to deflect *COL1* to restore the angle at the rigid joint. However, in order for the bend operator in the Diagram Layer to perform properly, the system must determine the type of connection at the other end of *COL1*. The system searches the object list in the Structure Layer for the structural component whose associated picture object is positioned at the other end of the column, which is determined to be *PINNED.SUPPORT1*. Finally, an order for the bend operation can be issued to the Diagram Layer. In this case, the column will not actually bend at all but will rotate on its pinned support to restore the angle at the fixed joint.

In pictorial terms, the spline representing the shape of the column receives the new coordinates from the rotate operator. As in a real paper and pencil drawing, the column (pictorially the spline) is now shown in the correct rotation but is no longer attached to the support at the bottom end. An implicit piece of knowledge says that all structural components connected by a joint remain connected throughout the deformation process. Therefore, the beam (pictorially the spline) is moved in the direction of the rotation of the column so that the column is again attached to the support.

REDRAW-I now proceeds to examine other parts of the picture. The system notes the hinged joint at the other end of the beam but finds no constraints that apply. That is, the original bending of the beam caused a rotation in the hinge which was allowable. REDRAW-I notes at this point that *COL2* is attached at the other end to another pinned support. The system then checks the moment around the fixed joint, *JOINT1* and determines that the entire structure must sway towards the hinged joint in order to release the moment around *JOINT1*. Again, the system need not determine the numerical value of these forces, only whether the moments were zero, negative or positive. The Diagram Layer replaces the coordinates of the spline for *COL2* just as it did with *COL1* and then changes the column's position so that it still attached to the pinned support at the other end.

The system goes through the drawing again to look for additional constraint violations in the current deflected shape of the frame structure. The sequence of the execution steps is displayed as shown in Figure 7. If all applicable constraints are built into the system, then the drawing should depict a stable deformed structure under load. (In the current implementation, however, REDRAW-I does not have the capability to recognize unstable structures.)


Kee Desktop 1 - Lisp Listener

```
(deflect 'structure.layer 'load1 'beam1)

Apply LOAD1 to BEAM1.

LOAD1 will deflect BEAM1 downward in middle.
Examining picture...

BEAM1 is connected to COL1 by JOINT1.
Angle between BEAM1 and COL1 is less than 90 degrees.
JOINT1 is of support type fixed.
Constraint: COL1 must be at 90 degrees to BEAM1.

Bend COL1 to restore 90 degree angle.

COL1 is attached to PINNED.SUPPORT1 at other end.
Constraint: COL1 must rotate on PINNED.SUPPORT1.

Rotate COL1 on PINNED.SUPPORT1.

Moment is non-zero around JOINT1.
Constraint: Moment must be zero around JOINT1.

BEAM1 is connected to COL2 by HINGE1.
Angle between BEAM1 and COL2 is less than 90 degrees.
HINGE1 is of support type hinged.
No constraints apply.
Examining picture...

COL2 is connected to PINNED.SUPPORT2.

Release moment at JOINT1.
STRUCT1 sways on PINNED.SUPPORT1 and PINNED.SUPPORT2.

Angle between COL2 and PINNED.SUPPORT2 is less than 90 degrees.
PINNED.SUPPORT1 is of support type hinged.
COL1 is connected to BEAM1 by JOINT1.
Angle between BEAM1 and COL1 is 90 degrees.
No constraints apply.

BEAM1 is connected to COL2 by HINGE1.
Angle between BEAM1 and COL2 is less than 90 degrees.
HINGE1 is of support type hinged.
No constraints apply.
Examining picture...

No more constraints apply.
```

Figure 7: Summary of problem solving process of REDRAW-I in Example 2

2.3 DISCUSSION

REDRAW-I successfully analyzed simple deflection shape problems. In REDRAW-I, the domain knowledge in the Structure Layer determines how the structural members are deformed one by one and, thus, how the reasoning proceeds. Since the constraints in the symbolic layer contain implicitly the knowledge that deformations propagate from one component to those connected to it, examination of the diagram also proceeds from the component sustaining the original load to the components connected to it, and so forth. In addition, an issue arises concerning the necessity of a "local versus extended" examination of a component in the propagation of the deformation. A hinge joint, for example, allows rotation of the components connected to it. The effect of the hinge on two connected components is localized at the connection point. A fixed joint, on the other hand, requires an examination of the type of attachment at the other end of the component so that an appropriate constraint can be applied and the correct deformation shape be imposed. Thus a more complex or extended examination of a component must take place to correctly implement the fixed joint constraint.

REDRAW-I allows the user to concentrate on the qualitative features of the structure, without requiring the specification of details. The diagrammatic components of the system facilitate the visualization of the particular deformation problem and its likely range of solutions. To aid in this visualization, we purposely include a "write-over" ability such that at each step after a shape transformation, the previous configuration is shown in dotted lines, just as a person draws a deformation right over the original line rather than create a separate new drawing. Displaying the "before" and "after" shapes allows the user to visually inspect and verify the inference process that was used in the shape transformation. The explanation facility of REDRAW-I, which explains every step of the reasoning process, provides the user with further insight into the constraints imposed and the inferences made to arrive at the final stable deflected shape.

We learned from the first implementation reported herein much about how the system should function at both the Structure Layer and at the Diagram Layer. However, since REDRAW-I followed a very informal analysis method, its heuristics did not work well when the structure became more complicated. The most critical shortcomings of its informal method is that it did not reason explicitly about forces and bending moments underlying the heuristics for determining how the deformation should propagate. As a result, when there were some ambiguities due to multiple rules being applicable to propagate deformation, it could not resolve the ambiguity or even reason about the real cause of such ambiguity, which would normally be due to effects of competing forces or bending moments. Furthermore, because there were no explicit reasoning about forces and moments, one could not proceed directly from the informal analysis to a more formal analysis, which would require reasoning about those concepts. REDRAW-II, described next, was implemented to address these problems, by employing a more formal analysis technique that would allow us to reason explicitly about forces, moments and the shape, while taking full advantage of diagrammatic reasoning capabilities.

3. REDRAW-II

REDRAW-II solves the deflection shape problem by explicitly reasoning about the forces, moments, and shape. Its analysis method better reflects the formal analysis technique employed by structural engineers, and, in principle, can be used to analyze much more complex structures than REDRAW-I could.

As in REDRAW-I, REDRAW-II has two layers: the Structure Layer which contains the symbolic representations of the domain knowledge, and the Diagram Layer which provides an environment for reasoning with a depictive representation. REDRAW-II differs from REDRAW-I in two main respects: First, the system's knowledge base and the reasoning mechanism makes explicit the three subproblems -- the force equilibrium, moment equilibrium and deflection -- which the structural engineer must consider simultaneously in order to determine the final deflected shape. REDRAW-II draws and manipulates three different diagrams, a force diagram, a bending moment diagram, and a shape diagram, to reason about the subproblems and display the results. In general, the three subproblems can be solved in any order, but the most efficient solution strategy depends on the problem and what information is available at any given point. A partial solution to one subproblem can be used to obtain a partial solution to another subproblem, and the solutions can be checked for mutual consistency. Secondly, the subproblems are not solved only through manipulating diagrams, but also through setting up equations and solving them. Diagrams are used not only to solve the subproblems but also to make control decisions such as what force to solve for next and what equation is likely to produce an answer for the force.

3.1 SYSTEM ARCHITECTURE OF REDRAW-II

As in REDRAW-I, REDRAW-II consists of the Diagram Layer and the Structure Layer. In REDRAW-II, the knowledge base of structural engineering knowledge in the Structure Layer is divided into two parts. One part is the set of three rule bases for rules relating to the analysis of force, moment and deflection. The relevant rule base is examined in full each time a different stage of the analysis is chosen for action. The rules express straightforward pieces of engineering knowledge which can be applied immediately to the problem at hand. Examples of such rules are "the horizontal force at a vertical roller support is zero," and "the vertical and horizontal forces at the unsupported end of cantilevered segment is zero." The other part of the Structure Layer knowledge base consists of methods for selecting an equation to set up that is most likely to lead to a solution and methods for solving equations.

3.1.1 The Task Hierarchy

In REDRAW-II, each problem solving step represented as a well-defined task, and the system maintains queues of tasks for the three separate subproblems. This design allows the system to move back and forth among the three subproblems as relevant knowledge becomes known.

The initial task is always a task to analyze a structure that has been specified both symbolically and diagrammatically. It, in turn, creates three high-level tasks to determine the bending moments, reaction forces and deflected shape of the proposed structure. Each task is placed in the separate task queue. The analysis tasks in any particular queue may be in a form that the system can act upon immediately, or it may spawn sub-tasks of its own to break down the problem even further. For example, *Find-all-forces* is the initial task that is put on the force queue. This task cannot be immediately performed. Instead, it causes creation of a sub-task to try to apply all of the force-related rules in the knowledge base to the problem of analyzing the force equilibrium. If the force rules are not successful, then new sub-tasks will be created to formulate and solve a force or moment equilibrium equation around a particular point in the structure, and then solve that equation qualitatively.

When a task is completed successfully, the new piece of information about the structure is placed in slots of the frames associated with the relevant structural members. For example, if a particular vertical or horizontal force is calculated at a point, then the qualitative value (zero, positive or negative) is placed to a slot in the force object associated with the structural member (i.e. the force object *Force.P* associated with the beam *PS*). When a task fails, it is marked "failed" and put back on the queue.

The current control strategy for deciding what queue to turn to look for the next task is as follows: The queues are prioritized in the order of force, moments, and shape, with the force queue having the highest priority[2]. Initially, the queue with the highest priority becomes the current queue. The system keeps dequeuing tasks from the current queue until the queue is empty or until none of the tasks on the queue succeeds. Then, the queue with the next highest priority becomes current. If there is no lower-priority queue than the current one, the highest-priority queue becomes the current one. This continues until all the queues are empty.

3.2 EXAMPLE

Figure 8 illustrates the three sub-problems of a problem solved by REDRAW-II. Figure 8a. shows graphically the information that is specified about a structure to be analyzed. This structure is specified both symbolically and diagrammatically. Figures 8b. - 8d. show the finished results of the force, moment and deflected shape analyses. To give an idea of the reasoning that REDRAW-II uses, we will describe part of each of the three stages of the analysis.

Focusing on the force diagram in Figure 8b. we are given only the vertical downward force imposed by the point load at *E*. In order to correctly determine the deflected shape, the system must identify all force reactions to the load and ensure that all forces are in equilibrium. The system will first examine the force rule-base for any rules that might be applicable. Finding no useful force rules, the system must set up and solve several qualitative equations to determine all of the force reactions. For example, in order to find the vertical force $V_{F\text{-}A}$ at *A*, the system will set up a moment equilibrium equation around point *C*. This equation will be of the form:

$$V_{F\text{-}E} * 1/2\ |BC| + BC * V_{F\text{-}A} = 0$$

Solving for $V_{F\text{-}A}$ qualitatively, the system determines that the vertical force at *A* is positive (upward). A similar equation set up around *B* will show that the vertical force at *D* is also positive, allowing all forces to be in equilibrium.

For the moment diagram in Figure 8c. the moment rule-base was utilized extensively. For example, the values of the bending moment at points *A* and *D* are both zero, since the specified hinged supports allow rotation of the attached column when a load is applied. This result is confirmed diagrammatically in Figure 8d. in which the deflected shape also shows the rotation of the hinged support.

[2] The order is actually arbitrary.

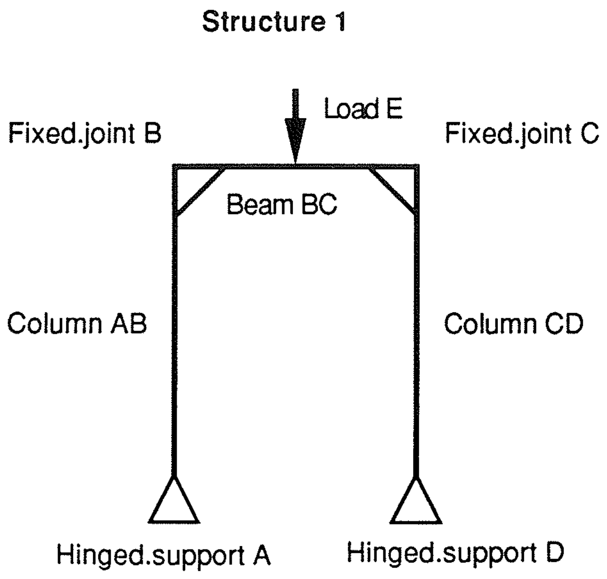


Figure 8a. Example Structure

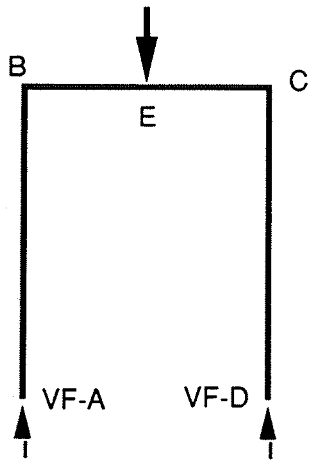


Figure 8b. Force Diagram

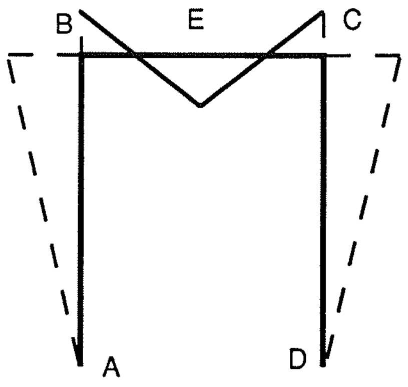


Figure 8c. Moment Diagram

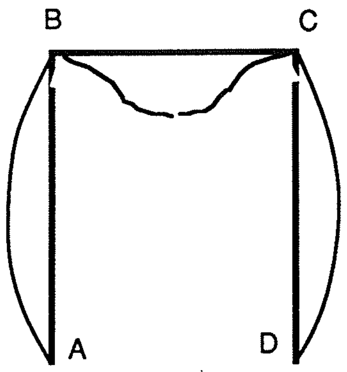


Figure 8d. Shape Diagram

Figure 8: An example of a problem solved by REDRAW-II

3.3 DISCUSSION

REDRAW-II is able to solve the same type of deformation shape problems as REDRAW-I could, while computing the forces and bending moments in addition to the shape. The analysis method of REDRAW-II is more formal than that of REDRAW-I. REDRAW-II explicitly reasons about theoretical concepts of interest such as forces and moments and inflection point, which underlie determination of deformation shape but which themselves are not visual. In REDRAW-I in contrast, such knowledge was not explicit in the rules for propagating deformation. Because REDRAW-II reasons explicitly about theoretical concepts and used equations to reason about them, it is possible to enumerate systematically all the possible qualitative solutions when there is ambiguity and identify the causes of such ambiguity. Also, this would make it more straightforward to proceed to a more precise quantitative analysis from the results of REDRAW-II. Furthermore, the fact REDRAW-II reasons about and generates diagrams of the forces, moments and shape makes it possible to check the consistency of the solutions.

Because of the more formal analysis method REDRAW-II employs, we believe that its problem solving capability is more general and also more easily extensible in principle than REDRAW-I. With respect to diagrammatic reasoning aspect, however, REDRAW-II relies more on symbolic reasoning to solve problems and relies less on diagrams than REDRAW-I. This is not surprising, since this type of problem is solvable in principle without any diagrammatic reasoning capability at all as demonstrated by QStruc[Fruchter, Law and Iwasaki 1991]. This is not evidence that diagrams are useless when a more mathematical analysis method is employed, but, as Larkin and Simon have shown in their work, diagrams are useful for controlling the reasoning process even when a formal (mathematical) technique is used[Larkin and Simon 1987].

In REDRAW-II, the diagrammatic information is used more for control purposes than for actually solving a problem, especially for computing the forces and the bending moments. For example, forces are computed by rules in a few simplest cases, and by setting up and solving equations for the rest. REDRAW-II's heuristic methods to determine what equation is most likely to produce an answer for a particular force given the available information relies on diagrammatic information. REDRAW-II's ability to make such control decisions is much weaker at this point than a human engineer's partly because we simply have not articulated many such heuristics in a general enough form to put them into the system. Another important reason for the weakness is that the set of diagrammatic inspection operators currently implemented in REDRAW is incomplete and does not allow us to implement some heuristics, especially those requiring detection of global features of diagrams (e.g. detecting symmetry).

In implementing REDRAW-I and -II, we initially intended all the diagrammatic operators, such as bend, rotate and smooth, to be domain- and task-independent. However, it has become clear that while some operators are domain-independent, others are quite domain- and task-specific. For example, our "bend" operator bends a straight line into a simple curve that resembles the curve even a novice would draw to indicate the shape of a stick under a load. However, the implementation of this "bend" operator reflects the assumptions implicit in the domain and the task -- for instance, the curvature of the bent line is large enough so that it can be clearly seen, but not so large that the structural member would appear to be broken. Also, the particular choice of inspection operators we have implemented reflect the nature of the problem we chose to work on. A more general-purpose diagrammatic reasoning layer will require a larger set of operators. The operators also need to be parameterized to work in a larger variety of situations. They must include operators for inspecting and manipulating both local and global features of a diagram. They must cover the types of operations humans can do fairly easily with diagrams.

4. CONCLUSION

This paper described our work on exploring the potential of diagrammatic reasoning in a concrete problem-solving context. We have built prototype programs REDRAW-I and -II, which reason qualitatively about deflection shape problems using diagrams. They solve the problem in a more computationally efficient manner than a similar system, QStruc [Fruchter, Law and Iwasaki 1991], in which a purely symbolic approach was taken to the same frame structure problem. The efficiency advantage over QStruc is due to the fact that use of the diagram allows the system to focus the solution process much better than QStruc, which literally blindly sets up all equilibrium equations that apply and tries to solve them. Our informal evaluation of the systems shows that the solution process of REDRAW programs are much more instructive in helping the user to gain intuitive understanding of how frame structures behavior under a load.

REDRAW-I's informal analysis technique involving propagating deformations to other parts of the structure uses a shape diagram extensively. REDRAW-II's analysis method involves more formal, symbolic reasoning, including formulating force and moment equilibrium equations and solving them. Unlike REDRAW-I, REDRAW-II explicitly solves the three different subproblems, namely forces, bending moments and the shape. When a more formal method of solving the problem is employed, the diagram is useful for controlling the inference.

We believe that diagrammatic reasoning has many advantages over purely symbolic reasoning in problems dealing with spatial information. The explicit representation of the geometric information greatly facilitates certain types of inferences about spatial configuration, that might require many inference steps using purely symbolic representation. Furthermore, since people do use diagrams extensively in many types of problems involving spatial information, programs that use diagrammatic representation in the similar manner will be much easier for people to understand. For this reason, programs that are based on diagrammatic representation will also be much more useful for teaching purposes.

We emphasize that REDRAW-I and-II are prototype systems that were developed primarily to explore the role of diagrammatic reasoning in qualitative structural analysis. The primary objective is to provide a good environment for studying diagrammatic reasoning, and how that type of reasoning is integrated with symbolic reasoning for engineering problem solving. The approach that we have undertaken allows us to examine and model more readily the flow of pictorial and symbolic reasoning as well as to better identify the visual operators which are important in the process of reasoning with diagrams. As a result of this study, we are in a better position to identify interesting problems concerning organization of information consisting of both symbolic and pictorial components and the complexity of problem solving process that uses such information. By developing a strong understanding of the role visual reasoning plays in the problem-solving process, we hope to be able to construct a general tool that can be used to build diagrammatic reasoning systems for other engineering problems.

4. 1 FUTURE WORK

The diagrammatic reasoning component of REDRAW that has been implemented thus far meets only the minimum required to complete the target task. We are in the process of designing a more general purpose diagrammatic representation and manipulation module. The module will contain a direct representation of a diagram using an array of cells as well as a symbolic representation of the diagram consisting of hierarchically organized elements of the diagram. It will also provide a whole range of parameterized operators for generating, manipulating and inspecting the diagram. We plan to make the module the kernel of a diagrammatic reasoning shell that can be used to implement problem solving systems for a variety of different tasks requiring use of line diagrams.

We are also working on a formal characterization of a diagrammatic representation. Diagrams are very useful in representing certain types of relations and making inferences about them, but if one is not careful about what types of inferences are warranted with a particular diagrammatic representation, one could make incorrect inferences or fail to make an inference. Therefore, it is important to be able to formally characterize a diagrammatic representation in terms of its information content and the types of inference sanctioned by the representation.

This is also essential in elucidating the role diagrams play in problem solving, given a problem and a particular type of diagrams used.

In addition to examining the role of diagrammatic reasoning in problem solving, we are considering the generality of our work and its extendibility to other areas of technical design such as in architecture and mechanical engineering. Larkin and Simon [Larkin and Simon 1987]show that even with a symbolic representation, problem solving efficiency in some cases can be greatly improved by organizing the information in a way that reflects the physical structure of the object represented. With a mixed symbolic and diagrammatic approach, interesting problems concerning the organization of the information and the computational complexity of the problem solving algorithm may arise that could later effect both scalability and generality. By developing a strong understanding of the role that visual reasoning plays in the overall problem-solving process, we hope to be able to construct a general tool that can be used to build diagrammatic reasoning systems in other problem domains.

REFERENCES:

Borning, A. H. (1979). Thinglab -- a Constraint-Oriented Simulation Laboratory. Ph. D. thesis. Department of Computer Science, Stanford University.

Brohn, D. (1984). Understanding Structural Analysis. Oxford, BSP Professional Books.

Chandrasekaran, B., Narayanan, N. H., and Iwasaki, Y. (1993). Reasoning with Diagrammatic Representations: A Report on the AAAI Spring Symposium. AI Magazine. **14:** 49-56.

Chandrasekaran, B. and Narayanan, N. H. (1990). Towards a Theory of Commonsense Visual Reasoning. Foundations of Software Technology and Theoretical Computer Science. Springer-Verlag.

Chandrasekaran, B. and Narayanan, N. H. (1992). Perceptual Representation and Reasoning. AAAI Symposium on Reasoning with Diagrammatic Representations, Stanford, CA,

Forbus, K. (1980). Spatial and Qualitative Aspects of Reasoning about Motion. the First National Conference on Artificial Intelligence, Stanford, CA,

Forbus, K., Nielsen, P., and Faltings, B. (1991). "Qualitative Spatial Reasoning: The CLOCK project." Artificial Intelligence **51**(1-3): 417-471.

Fruchter, R., Law, K. H., and Iwasaki, Y. (1991). QSTRUC: an Approach for Qualitative Structural Analysis. the Second International Conference on the Application of Artificial Intelligence to Civil and Structural Engineering, Civil-Comp Press.

Iwasaki, Y. (1989). Qualitative Physics. The Handbook of Artificial Intelligence. Addison-Wesley.

Koedinger, K. and Anderson, J. R. (1990). "Abstract Planning and Perceptual Chunks: Elements of Expertise in Geometry." Cognitive Science **14**: 511-550.

Kosslyn, S. M. (1980). Image and Mind. Cambridge, MA, Harvard University Press.

Larkin, J. H. and H. A. Simon (1987). "Why a Diagram is (sometimes) Worth Ten Thousand Words." Cognitive Science **11**:

Lindsay, R. K. (1992). Diagrammatic Reasoning by Simulation. AAAI Symposium on Reasoning with Diagrammatic Representations, Stanford, CA,

Novak, G. and Bulko, W. (1992). Uses of Diagrams in Solving Physics Problems. AAAI Symposium on Reasoning with Diagrammatic Representations, Stanford, CA,

Roddis, W. M. K. and Martin, J. L. (1991). Qualitative Reasoning about Steel Bridge Fatigue and Fracture. the Fifth International Workshop on Qualitative Reasoning about Physical Systems,

Slater, J. H. (1986). Qualitative Physics and the Prediction of Structural Behavior. ASCE Symposium on Expert Systems in Civil Engineering,

Weld, D. S. and de Kleer, J. Ed. (1990). Readings in Qualitative Reasoning about Physical Systems. The Morgan Kaufmann Series in Representation and Reasoning. San Mateo, CA, Morgan Kaufmann.