Problem-Solving with Diagrammatic Representations*

Brian V. Funt

*Computer Science Department, State University of New York at Buffalo, Buffalo, NY 14226, U.S.A.***

Recommended by Daniel G. Bobrow and Aaron Sloman

ABSTRACT*

Diagrams are of substantial benefit to WHISPER, a computer problem-solving system, in testing the stability of a "blocks world" structure and predicting the event sequences which occur as that structure collapses. WHISPER's components include a high level reasoner which knows some qualitative aspects of Physics, a simulated parallel processing "retina" to "look at" its diagrams, and a set of re-drawing procedures for modifying these diagrams. Roughly modelled after the human eye, WHISPER's retina can fixate at any diagram location, and its resolution decreases away from its center. Diagrams enable WHISPER to work with objects of arbitrary shape, detect collisions and other motion discontinuities, discover coincidental alignments, and easily update its world model after a state change. A theoretical analysis is made of the role of diagrams interacting with a general deductive mechanism such as WHISPER's high level reasoner.

1. Introduction

Diagrams are very important tools which we use daily in communication, information storage, planning and problem-solving. Their utility is, however, dependent upon the existence of the human eye and its perceptual abilities. Since human perception involves a very sophisticated information processing system, it can be argued that a diagram's usefulness results from its suitability as an input to this powerful visual system. Alternatively, diagrams can be viewed as containing information similar to that contained in the real visual world, the canonical entity the human visual system was presumably designed through evolution to interpret. From this latter perspective, diagrams are a natural representation of certain types of primarily visual information, and the perceptual system simply provides an appropriate set of database accessing functions. Both these viewpoints underly the work described in this paper.

The role of diagrams is explored in a computer problem-solving program, named

* This paper is a substantially lengthened version of a similar paper appearing in IJCAI-5, [6].

** Now with: Dept. of Computing Science, Simon Fraser University, Vancouver, B.C., Canada V5A IS6.

WHISPER, which refers to diagrams during its processing. WHISPER's high-level reasoning component (HLR), built along the lines of traditional procedural AI problem-solving programs, has the additional option of requesting observations in a diagram. It does this by asking its "perceptual system" to "look at" the diagram with its parallel processing "retina". The questions that the perceptual system can answer are called *perceptual primitives*. If necessary, the HLR can also make changes to the current diagram. Fig. 1 shows WHISPER's overall structure.

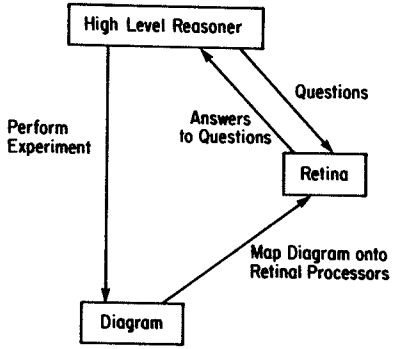

FIG. 1. The WHISPER proposal.

Upon receiving a diagram of a blocks world structure, WHISPER outputs a set of diagrams representing the sequence of events which occur as the structure collapses. The HLR contains knowledge about stability and the motion of falling objects. Using the retina to locate objects and their supports, it checks the stability of each object shown in the diagram. Unstable objects may either rotate or slide. In cases where one is rotationally unstable, the HLR asks the retina to "visualize" it rotating and thereby determine at what point it will hit some other object. Using this information WHISPER outputs an updated diagram showing the object rotated into its new position. Then with this new diagram, it restarts the problem-solving process from the beginning—rechecking the stability of each object, moving one of them, outputting another diagram, and restarting again. The process terminates when either all the objects are stable or the problem becomes too complex for the stability tester. A detailed discussion of the HLR will be postponed until Section 3.

1.1. Motivation

A strong case for computer use of diagrams as models for Geometry has been made by Gelernter (1963), and as general analogical representations by Sloman (1971). Networks with nodes representing "ideal integers" and arcs representing relationships between them were used as models for statements in arithmetic by Bundy (1973). Hayes (1974) and Bobrow (1975) comment on the theoretical nature of analogical representations; Hesse (1969) and Nagel (1961) discuss analogical reasoning.

There is a variety of reasons for using diagrams in computer problem-solving. Diagrams such as maps, architectural plans, and circuit diagrams routinely facilitate human problem-solving. Perhaps diagrams function not merely to extend memory capacity, but rather present the important information in a particularly useable form. If they do, then the human visual system provides a paradigmatic example of a system for accessing these representations. Since it exploits a high degree of parallelism, it leads us into the realm of a different type of hardware. This is an exciting step, however, because we can see how much hardware characteristics influence our thinking about the difficulty of various problems and the feasibility of their solution. For example, we know we could compute with Turing Machines—but would we? Because WHISPER's retina harnesses parallelism, it in effect extends the available machine instruction set with special ones for diagram feature recognition. WHISPER is primarily an exploration of the question: to what extent can problem-solving be simplified through experiment and observation with diagrams? This is in contrast (but not in opposition) to the usual method of deduction within a formal theory as explained in the next section.

1.2. Theoretical framework

Any problem-solving system needs a representation of the problem situation. The standard approach in AI is to formalize the domain. We choose a language and write down a set of statements (axioms, productions, assertions, or a semantic network) describing the world. So that the problem-solver can generate new statements from this initial set, we provide a general deductive mechanism (theorem prover, programming language control structure, network algorithm). In terms of the predicate calculus the axioms define a theory, T, and so long as it is not self-contradictory there will be at least one model M (an assignment of predicates to the predicate symbols, functions to the function symbols, and individuals to the constant symbols) which satisfies it. Since our intention in axiomatizing the world was to accurately describe it, we expect it to be one of the models satisfying T.

We may find a second model M' satisfying T (in general there will be many such models). Now—and this is the main thrust of WHISPER—in some cases we can use M' to provide information about M without deriving it from T. What is required is that some of the predicates, functions and individuals of M' correspond to some of the predicates, functions and individuals of M in such a way that it is possible to translate the results obtained when these predicates and functions are applied to individuals in M' into the results that would be obtained if the corresponding predicates and functions were to be applied to the corresponding individuals in M. The similarity between M and M' means that experiments and observations made in M' yield results similar to those that would be obtained in M. As shown in Fig. 2, for WHISPER M' is the combination of its diagram and diagram re-drawing procedures. WHISPER obtains information about the blocks world by using its retina to observe the results of experimental changes made to its diagrams by the re-drawing procedures.

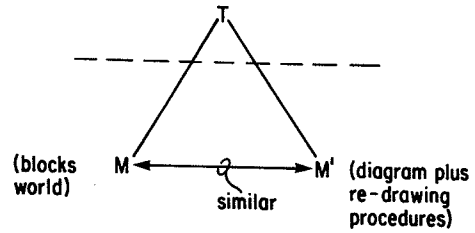

FIG. 2.

WHISPER is a prototype system designed to explore the extent to which problem-solving can be carried out below the dashed line of Fig. 2; however, it does do some reasoning above the line. WHISPER's success argues for working below the line, but not against working above the line. WHISPER's HLR is an above-the-line component.

A natural question is why use M' instead of M? If M is readily accessible then there is no reason not to use it; but, frequently it will not be. For example if we want to determine the stability of a pile of blocks on the surface of the moon, then we could construct a similar pile of blocks on earth and determine the result by experiment. In this case M, the pile of blocks on the moon is inaccessible. We can see that a lot can be learned from the blocks on earth, but some above-the-line inference must be done to handle the discrepancies arising as a result of the difference in gravity.[1]

2. Mechanisms for Diagram Interaction

The retina and perceptual primitives are designed to provide WHISPER with a new set of operations whose execution times are of the same order of magnitude as conventional machine instructions. To achieve this a high degree of parallelism has been incorporated into the system. The retina is a parallel processor, and the perceptual primitives are the algorithms it executes. (Do not be misled by the term "retina"; it refers to a general system of receptors and processors for the early stages of perceptual processing, rather than implying any close resemblance to the human retina.) Each perceptual primitive, when executed by the retina, determines whether some particular feature is present in the diagram. WHISPER's retina mixes parallel and sequential computation, so the features it can recognize are not subject to the same theoretical limitations as perceptrons (Minsky and Papert (1969)).

2.1. The retina

WHISPER's retina is a software simulation of hardware which, given the rapidly advancing state of LSI technology, should soon be possible to build. It consists of a collection of processors, each processor having its own input device called a *receptor*. There is a fixed number of processors, and they are all identical. As with the human eye, WHISPER's retina can be shifted to fixate at a new diagram location (also a feature

[1] I am grateful to Raymond Reiter for many of the ideas in Section 1.2.

of a program by Dunlavey (1975)), so that each processor's receptor receives a different input from the diagram. This fixation facility is important because the resolution of the retina decreases from its center to its periphery. Without being able to fixate, it would be impossible for WHISPER to examine the whole diagram in detail. Economy of receptors and processors dictates the use of decreasing resolution. (A declining resolution is also a characteristic of the human eye.) Each receptor covers a separate segment of the diagram and transmits a single value denoting the color of that region. The geometrical arrangement of the receptors and the area each covers is shown in Fig. 3.[2] The "circles" in the figure are called *bubbles*, and they are

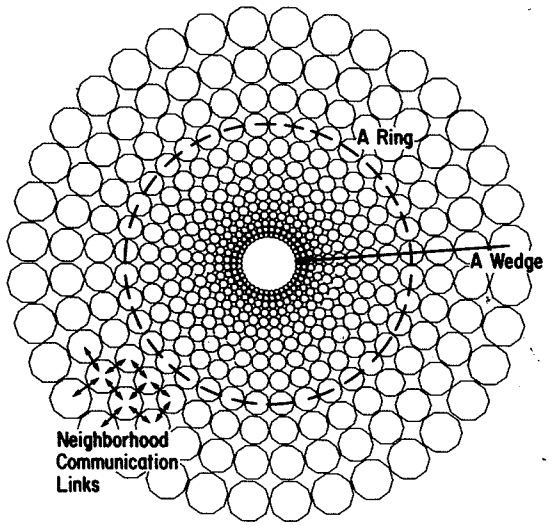

FIG. 3. WHISPER's retina.

arranged in *wedges* (rays emanating from the center) and *rings* (concentric circles of bubbles). The resolution varies across the retina because a larger portion of the underlying diagram is mapped onto a bubble depicted by a larger circle. Since the complete group of receptors is assumed to sense and transmit all signals in parallel, fixations are fast.

Each retinal processor has direct communication links to its nearest neighbors plus one additional link via a common databus connecting all the processors to a supervisory processor called the *retinal supervisor*. The communication topology has been restricted in this simple way to ensure a feasible future hardware implementation.

[2] There are more receptors filling the central blank area of Fig. 3; however, there is still one special case receptor in the very center which must be handled separately. In order to speed up the retinal simulation the bubbles lying in the blank central area can be fixated separately so they are mapped onto only when they are needed.

The bubble processors are each small computers with independent memory. They all simultaneously execute the same procedure; however, each bubble does not necessarily execute the same instruction at the same time. In the current implementation, a call to the LISP evaluator simulates a processor; and LISP MAPping functions simulate the parallel control structure.

Although the bulk of the processing of the perceptual primitives is done in parallel, there is also a small amount of sequential processing which is performed by the retinal supervisor. The retinal supervisor also directs the parallel processing by choosing which procedure the bubbles should execute next and broadcasting this common procedure to them.

2.2. The perceptual primitives

Each perceptual primitive detects a *problem domain independent* diagram feature. The HLR assigns these features interpretations pertinent to the problem it is solving. The current set of implemented perceptual primitives include ones to: find the center of area of a shape; find the points of contact between a shape of one color and a shape of another; examine curves for abrupt slope changes; test a shape for symmetry; test the similarity of shapes; and visualize the rotation of a shape while watching for a collision with another shape.

The CENTER-OF-AREA perceptual primitive is an illustrative example of the general operation of the perceptual primitives. It computes the center of area of a shape relative to the origin defined by the center of the retina. For each piece, ΔA, of the total area we need to compute the x and y components of its contribution to the total area. Dividing the vector sum of these contributions by the total area yields the coordinates of the center of area. Since each retinal bubble receives its input from a fixed sized area of the diagram and is at a fixed location relative to the retina's center, each bubble can independently compute the components of its contribution to the total area. The bubbles whose receptors do not lie over any part of the shape simply do not contribute. The retinal supervisor performs the summation and the division by the total area. A separate primitive computes the total area. It simply totals the area of all the contributing bubbles. If the computed center of area is far from the retina's center its accuracy can be improved by fixating the retina on the estimated center of area and then recomputing. The decision to iterate is made by the retinal supervisor. The accuracy improves because more of the central, high-resolution portion of the retina is used.

It is possible that systematic errors might lead to a discrepancy between the center of area as seen by the retina and the actual center of area of the object in the diagram. This is the case because the diagram-to-retina mapping does not take into account what fraction of a bubble's picture region is covered by an object. The bubble is simply marked whenever any portion of its region is covered. In practice, the accuracy of the center of area test was more than adequate for WHISPER; if necessary the accuracy could always be improved by adding more bubbles to the retina, increasing its resolution.

The center of area is used for more than simply providing the center of gravity of the objects in WHISPER's problem domain. Other primitives (symmetry, similarity, and contact finding) fixate on a shape's center of area before beginning their calculations. For example, if a shape is symmetrical its center of area will be on its axis of symmetry.

Another important primitive is RETINAL-VISUALIZATION. What is "visualized" is the rigid rotation of a shape about the retinal center. While the shape is rotating the collision detection primitive can be called as a demon to watch whether the rotation causes the shape to overlap with another stationary shape. This is useful both in "blocks world" environments involving moving objects and in testing whether two shapes are equivalent under rotation. The process is termed *visualization* because it does not involve modifying the diagram, but instead is totally internal to the retina itself. It simply entails an organized and uniform exchange of information amongst neighboring bubbles.

The geometrical arrangement of the bubble receptors facilitates the visualization of rotations. From Fig. 3 it can be seen that aligning the bubble centers along wedges results in a constant angular separation between bubbles of the same ring when they are from neighboring wedges, and that this constant is independent of the ring chosen. Thus, to rotate a shape clockwise each bubble marked by the shape simply sends a message to its clockwise ring neighbor asking it to mark itself. The sender then erases its own mark. A collision is detected if a bubble receives a message to mark when it is already marked by a shape other than the rotating one. Although the shape is rotated in sequential steps, the time required is still short because

(i) there are, as a maximum, only as many steps to be made as there are wedges on the retina (currently 36); and

(ii) all the message passing and collision checking occurs in parallel during each step.

The coarse retinal resolution means that the visualization process is much faster than the alternative of rotating the object by small increments directly in the diagram. However, the coarse resolution also means that the collision test may falsely predict a collision. Although the collision test may occasionally generate such "false alarms", it will never fail to correctly predict a true collision. The reason for this is that during the diagram-to-retina mapping a point in the diagram is blurred to fill a whole bubble on the retina with the result that the objects in the diagram appear slightly enlarged on the retina. To check out a possible false alarm the HLR

(i) calls the re-drawing procedures to rotate the object in the diagram to the point where the collision is expected,

(ii) fixates the retina at the predicted collision point,

(iii) asks the retina (now with its high resolution center) to see if the colliding objects are touching.

The CONTACT-FINDER primitive establishes the points at which an object touches other objects. The retina is first fixated on the center of area of the object

and then the retinal supervisor directs each retinal bubble to execute the following steps:

Step 1. If the bubble value is not the color of the object then stop.

Step 2. For each of its neighboring bubbles do Step (3).

Step 3. If neighbor's value is the color of a different object send a "contact-found" message to the retinal supervisor.

Step 4. Stop.

The retinal supervisor may receive quite a number of messages from bubbles in the contact regions. It must sort these into groups—one for each distinct area of contact. To do this the retinal supervisor sequentially follows the chain of neighborhood links from one contact bubble to another. Each bubble in the chain is put in the same contact group. If no neighboring bubble is a contact bubble, then the chain is broken. Long chains indicate that the objects touch along a surface while short ones indicate that they touch only at a point. The bubble coordinates of the endpoints of the chain represent the extremities of a contact surface, and the average of the coordinates of all the bubbles in the group is a good place at which to fixate the retina for a more detailed analysis of the contact.

When two objects touch there is a good chance that one supports the other unless they are just sitting side by side. To determine which object is the supporter and which the supportee, the coordinates of the touching bubbles are compared to find which is "above" the other in the diagram. The assignment of "up" is problem domain dependent and so is made by the HLR.

Another perceptual primitive, FIND-NEAREST, finds the bubble closest to the retinal center satisfying a given condition. For example, to find the object nearest to point P in the diagram the retina is fixated at P and then asked for the nearest marked bubble. The organization of the retina into rings, each an increasing distance from the center, facilitates the search for the required nearest bubble. To find the nearest bubble to the center of the retina satisfying condition C, the retinal supervisor executes the following algorithm:

Step 1. Direct each bubble to test C and save the result (either 'true' or 'false').

Step 2. For $n = 1$ to the number of rings on the retina do Steps 3 and 4.

Step 3. Direct each bubble to report its wedge and ring coordinates as a message to the retinal supervisor if the following hold: (a) it belongs to ring n, (b) its saved value is 'true'.

Step 4. If there is a message pending for the retinal supervisor from step (3), return the coordinates specified in that message (if there is more than one message pick any one of them—all bubbles in a ring are equidistant from the retinal center) to the calling procedure.

This algorithm is a good example of the difference between efficiency in sequential and parallel computation. Since testing C could be a lengthy computation, it is more efficient in terms of elapsed time to simultaneously test C on all bubbles as in Step 1, than to test it for only those bubbles in the scanned rings of Step 3. On a sequential processor it would be best to test C as few times as possible; whereas, on a parallel

processor the total number of times C is tested is irrelevant (assuming the time to compute $C(x)$ is independent of x). It is the number of times C is tested sequentially which is important.

The SYMMETRY primitive tests for symmetry about a designated vertical axis by comparing the values of symmetrically positioned bubbles. An object is symmetrical (WHISPER tests for vertical and horizontal reflective symmetry), if each bubble having its "color" value has a symmetrically located bubble with the same value. If when testing the vertical reflective symmetry of a blue object, say, the bubble in the third wedge clockwise from the vertical axis and in the fourth ring from the center has the value 'blue', then the value of the bubble in the third wedge counterclockwise from the vertical axis and in the fourth ring must be checked to see if it is also 'blue'. If it is not, then possibly the discrepancy can be ruled out as insignificant; otherwise, the object is asymmetrical. Neighborhood message passing is used to bring together the values from bubbles on opposite sides of the proposed axis. The technique is to cause whole wedges to shift in a manner perhaps best described as analogous to the closing of an Oriental hand fan. All the bubbles to the left of the axis send their values clockwise, while all those to the right send theirs counterclockwise. Messages which meet at the axis are compared and will be equal if the object is symmetrical.

The symmetry test must be supplied a proposed axis of symmetry. The center of area offers partial information on determining this axis since it must lie on it if the object is symmetrical. This does not, however, provide the orientation of the axis. Although the simplest solution may be to test the object in all of the wedge orientations by using the rotational visualization, if one more point on the axis of symmetry could be found the axis would be uniquely determined. Such a point is the center of the circumscribing circle of the object. The only problem is that thus far I have not managed to devise a quick parallel algorithm for finding this center. Although in some cases they may coincide, in general I expect the center of area and the center of the circumscribing circle to be distinct for objects with only a single axis of symmetry.

An unexpected and interesting property of WHISPER's retinal geometry leads to a simple method, employing neighborhood communication, for scaling the retinal 'image' of an object. The primitive is RETINAL-SCALING. An object is scaled correctly (i.e. without distorting its shape) if each bubble having its value, sends this value to a bubble in the same wedge, but a fixed number of rings away. As long as each value is moved the same number of rings either inwards or outwards from the bubble which originally holds it, the size of the 'image' of the object is changed but its shape is preserved (Fig. 4). This is the case because the constraint of aligning the bubbles into wedges such that each bubble touches all of its immediate neighbors is satisfied by increasing the bubble diameters by a constant factor from ring to ring. For a proof of this see Funt (1976). Scaling an object by neighborhood communication is implemented by having each bubble simultaneously send its value as a message to its neighbor in the same wedge in either the appropriate inwards or

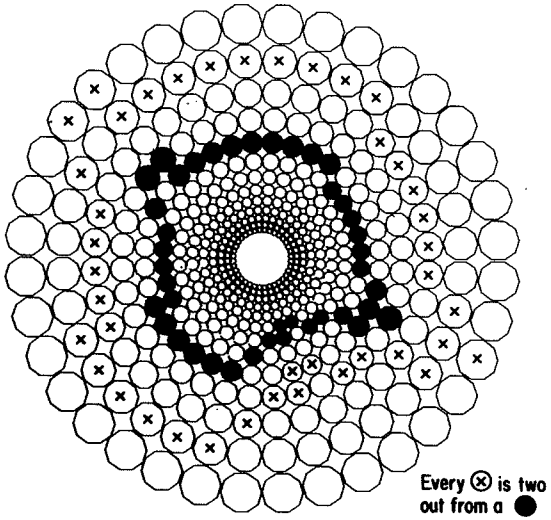

FIG. 4.

outwards direction, and repeating this message passing process sequentially as many times as necessary to bring about the required scaling.

The SIMILARITY PRIMITIVE determines whether two objects, A and B, are similar under some combination of translation, rotation and scaling, and if so returns the angle of rotation, direction and distance of translation, and the scale factor. It works by taking one object, say A, and translating, scaling and rotating it so it can be matched with the other. Since the center of area of an object is unique the centers of area of A and B must be aligned if they are to match. Thus the first step is to find the centers of area, and then to translate A. Rather than call the re-drawing transformations to move A in the diagram, its translation can be accomplished entirely on the retina by:

(i) fixating on the center of area of A,
(ii) asking all bubbles not containing A to mark themselves as empty space,
(iii) fixating on the center of area of B while superimposing this new image on the old one.

After translation A must be scaled. If A and B are to match, then their areas will need to be the same; therefore, we must scale A by a factor equal to the square root of the ratio of the areas of the two objects (i.e. scalefactor = squareroot(area(B)/area(A)). The areas of A and B are available as a by-product of the center of area calculation. Now that the objects are aligned and the same size, A is rotated about its center of area using retinal visualization to see if there is any orientation at which it matches B.

CURVE-FEATURES analyses curves. In order to begin, it must first find the

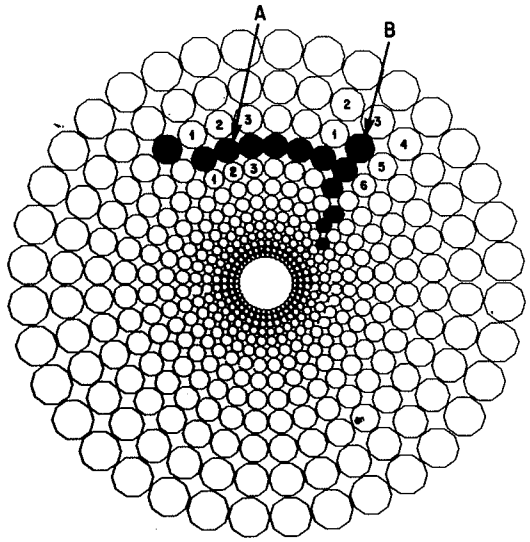

FIG. 5.

retinal bubbles on the curve. Given one bubble on the curve, the others can be found by following the chain of bubbles each having the same value. In WHISPER's diagrams the contours of objects are "colored" a different shade from their interiors, and this helps prevent the curve following process from getting lost tracing chains of bubbles which are part of an object's interior. It is not strictly necessary to color code the object contours, since a contour bubble can be determined by the type of neighbors surrounding it, but coding is cheaper and easier.

Once the set of bubbles on the curve is found, each bubble in the set can individually test for the occurrence of a particular feature; therefore, the whole curve is tested in parallel. A bubble detects a sharp bend in the curve if there is an imbalance in the number of its neighbors on opposite sides of the curve which are themselves not members of the curve. This is illustrated by Fig. 5 in which bubble A has three neighbors on each side of the curve, whereas bubble B has six neighbors on one side and none on the other. Thus, a bubble tests for bends by:

(i) asking its neighbors whether or not they are on the curve, and
(ii) comparing the number of responses originating from opposite sides of the curve.

For a simple closed curve, if the bubble knows which responding neighbors are interior and which are exterior, then it can additionally classify the bend as convex or concave.

The slope of a curve at any curve bubble is determined as the perpendicular to the bisector of the angle between the centers of its neighboring bubbles on the curve. This yields a rough approximation to the actual slope, but it is sufficient for quickly

testing whether any drastic slope change occurs over the length of the curve. To more accurately determine the slope at a particular point, the retina is fixated on it for higher resolution. The curve tangent is then the perpendicular to the bisector of the angle between wedges with the most bubbles on the curve. The angle between wedges can be used because they emanate directly from the center of the retina, just as the curve must when the retina is centered on it. This method is more accurate than measuring the angle between neighboring bubbles because there are more wedges than neighbors. The HLR mainly uses this test to measure the slope of surfaces at contact points to decide whether or not an object will slide.

2.3. The underlying diagram

We began with the view that the retina is a special purpose parallel processor designed to detect diagrammatic features without saying anything about the precise nature of the diagrams themselves. With the retinal processor in hand, we can now see that the representation of the diagrams is unimportant as long as each bubble receives its correct input. This is analogous to a program which issues a READ command without caring whether the input is coming from a card reader, a file, or a terminal. The method of mapping from the diagram to the retinal bubbles' input must be fast, however, because the retina is re-filled everytime it is fixated at a new diagram location.

There are at least two different types of representing media for the underlying diagram. The first is the conventional medium of visible marks on a two-dimensional surface, usually paper. The map from diagram to human retina is accomplished by the lens of the eye focusing the incoming light. Since there is simultaneous stimulation of the receptors, it is a very fast process.

The second possible type of diagram representation is similar to that used in generating computer graphics. The diagram is specified as a list of primitive elements (in graphics applications, usually line segment equations). In a similar vein, Kosslyn (1975) proposes that human visual imagery is in some ways analogous to the storage and display of graphics images. The parallel processing capacity of WHISPER's retina can be used to quickly map each primitive element into the proper bubble inputs. To mark all bubbles lying on line segment, S, the retinal supervisor directs every bubble to determine independently if it is on S, and if so, to mark itself. Since this simple test—do a circle and a line segment intersect?—is performed by all bubbles simultaneously, the time required is independent of the length of S. The same method can mark all bubbles within any simple shape such as a circle, square or triangle in time independent of its area. Regardless of the type of primitive element, the time taken to "draw" the diagram on the retina is, however, proportional to the number of primitives in its description. They must be processed sequentially.

Due to the lack of true parallel processing, neither of the above two types of diagram representations is used in WHISPER. Instead, the diagram is implemented as a square array. Each array cell denotes a point on a real world, pencil and paper diagram.

2.4. The re-drawing transformations

The re-drawing transformations are the procedures the HLR can call to change the underlying diagram. In WHISPER there are transformations for adding and removing lines, and for rigidly translating and rotating shapes. Other non-linear transformations could be added if required. These re-drawing procedures are of course dependent upon the representation of the diagram they modify, and the ease and efficiency with which they can be implemented could affect the choice of diagram representation.

3. WHISPER in Operation

With the basic mechanisms for interaction with the diagram now understood, it is appropriate to see how they are used in the course of solving a problem. We will consider problems of the type: predict the sequence of events occurring during the collapse of a "blocks world" structure. The structure will be a piled group of *arbitrarily* shaped objects of uniform density and thickness. If the structure is stable, there are no events to describe; if it is unstable, then the events involve rotations, slides, falls, and collisions. WHISPER accepts a diagram of the initial problem state, and produces a sequence of diagrams, called *snapshots*, as its qualitative solution. A quantitative solution specifying precise locations, velocitites, and times is not found; however, deriving one from a qualitative solution should not be too difficult (deKleer (1975)).

Fig. 6 is a typical example of WHISPER's input diagrams. They all depict a side view of the structure. Each object is shaded a different "color" (alphanumeric value) so it can be easily distinguished and identified. Objects' boundaries are also distinctly colored. The diagram depicts a problem, called the "chain-reaction problem", which is particularly interesting because the causal connection between objects B and D must be discovered.

3.1. The Qualitative HLR

The HLR is the top level of the WHISPER system. It is solely responsible for solving each problem; the diagram and retina are simply tools at its disposal. It consists of procedural specialists which know about stability, about the outcome of different varieties of instability, how to interpret each perceptual primitive, and how to call the transformation procedures to produce the solution snapshots. There are two types of instabilites—rotational and sliding. For clarity, sliding instabilities will not be discussed for the present. Operation of the system follows the steps:

Step 1. Determine all instabilities.
Step 2. Pick the dominant instability.
Step 3. Find pivot point for rotation of unstable object.
Step 4. Find termination condition of rotation using retinal visualization.
Step 5. Call transformation procedure to modify diagram as determined in Step 4.

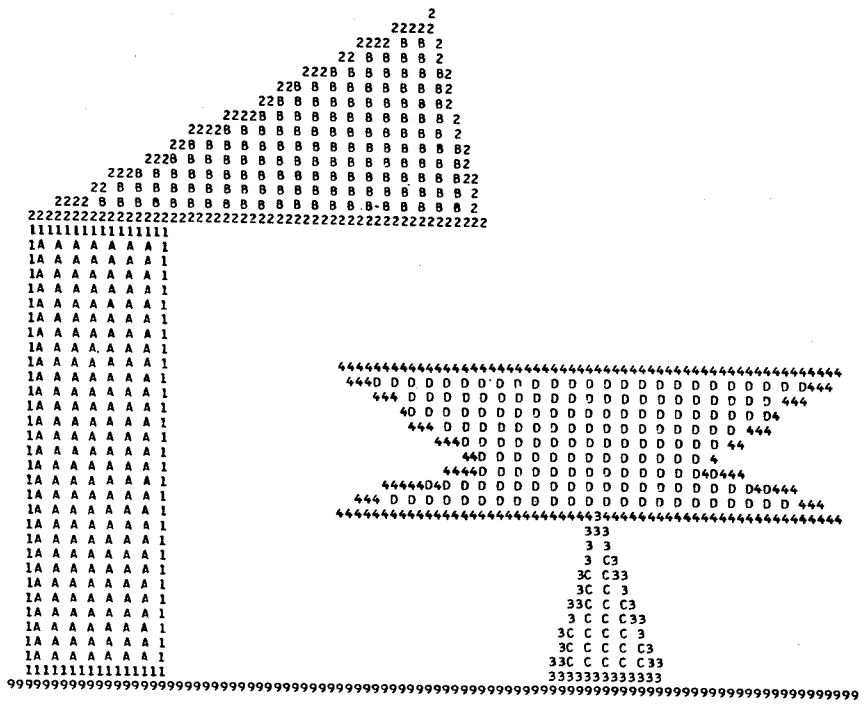

FIG. 6. Chain reaction problem.

Step 6. Output modified diagram as a solution snapshot.

Step 7. Use snapshot from Step 6 as input and restart from Step 1.

In what follows we elaborate on each of these steps.

The diagram and retina are an invaluable aid to the HLR in discovering what stops an object's motion, and in accomplishing the necessary state change. The chain reaction problem demonstrates this. The stability specialist directs the retina to fixate at numerous locations while perusing the diagram, and from an analysis (discussed below) of the visible support relationships determines that *B* is the only unstable object. *B* will pivot about the support point closest to its center of gravity. The retina is fixated there (the top right corner of *A*), so *B*'s rotation can be visualized. As the object rotates, two events are possible. It may collide with another object, or it may begin to fall freely. The conditions under which either of these occur are monitored during the visualization. From this simulation of *B*'s rotation, its collision with *D* is discovered, and its angle of rotation and location of first contact with *D* are found. Because of the coarseness of the retinal resolution, this angle of rotation is only approximate. This approximate value is used in conjunction with feedback from the diagram to refine the angle of rotation as follows. First the redrawing transformations are called to produce a new diagram (Fig. 7) in which *B*

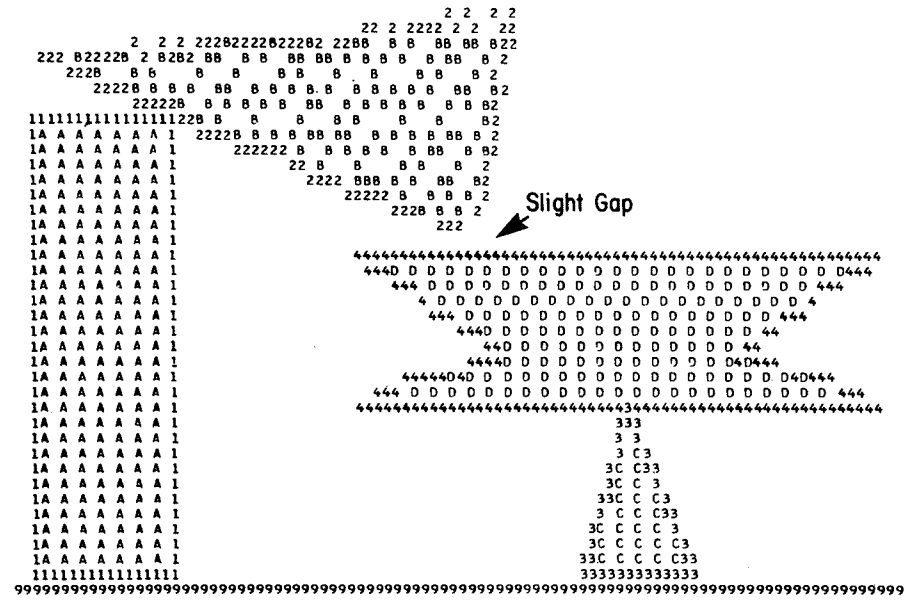

FIG. 7.

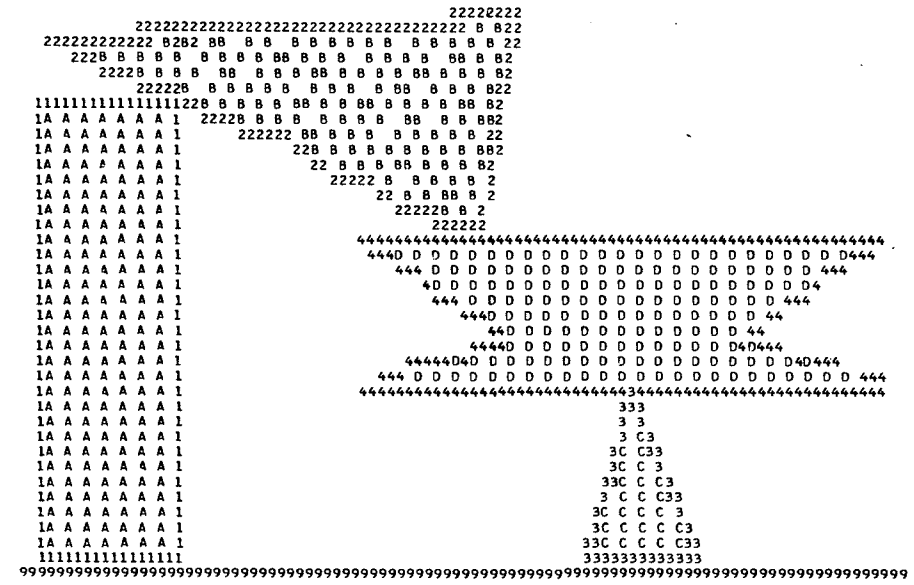

FIG. 8. First snapshot.