COGNITIVE SCIENCE 11, 65–99 (1987)

Why a Diagram is (Sometimes) Worth Ten Thousand Words

JILL H. LARKIN
HERBERT A. SIMON
Carnegie-Mellon University

We distinguish diagrammatic from sentential paper-and-pencil representations of information by developing alternative models of information-processing systems that are informationally equivalent and that can be characterized as sentential or diagrammatic. *Sentential* representations are sequential, like the propositions in a text. *Diagrammatic* representations are indexed by location in a plane. Diagrammatic representations also typically display information that is only implicit in sentential representations and that therefore has to be computed, sometimes at great cost, to make it explicit for use. We then contrast the computational efficiency of these representations for solving several illustrative problems in mathematics and physics.

When two representations are informationally equivalent, their computational efficiency depends on the information-processing operators that act on them. Two sets of operators may differ in their capabilities for recognizing patterns, in the inferences they can carry out directly, and in their control strategies (in particular, the control of search). Diagrammatic and sentential representations support operators that differ in all of these respects. Operators working on one representation may recognize features readily or make inferences directly that are difficult to realize in the other representation. Most important, however, are differences in the efficiency of search for information and in the explicitness of information. In the representations we call diagrammatic, information is organized by location, and often much of the information needed to make an inference is present and explicit at a single location. In addition, cues to the next logical step in the problem may be present at an adjacent location. Therefore problem solving can proceed through a smooth traversal of the diagram, and may require very little search or computation of elements that had been implicit.

According to *Bartlett's Quotations,* "a picture is worth 10,000 words" is a Chinese proverb. On inquiry, we find that the Chinese seem not to have heard of it, but the proverb is certainly widely known and widely believed in our culture. In particular, problem solvers in domains like physics and engineering make extensive use of diagrams, a form of pictures, in problem

The first author was supported in part by grant #MDR-8470 166 from the NSP Directorate for Science and Engineering Education and by the John Sima Guggenheim Foundation.

Correspondence and requests for reprints should be sent to Jill H. Larkin, Department of Psychology, Carnegie-Mellon University, Pittsburgh, PA 15213.

solving, and many distinguished scientists and mathematicians (e.g., Einstein, Hadamard) have denied that they "think in words." To understand why it is advantageous to use diagrams—and when it is—we must find some way to contrast diagrammatic and non-diagrammatic representations in an information-processing system.

When they are solving problems, human beings use both internal representations, stored in their brains, and external representations, recorded on a paper, on a blackboard, or on some other medium. Some investigators (e.g., Pylyshyn, 1973) have argued that all internal representations are propositional, while others (e.g., Anderson, 1978) have argued that there is no operational way in which an internal propositional representation can be distinguished from a diagrammatic one. Although our discussion may be relevant to this current controversy about the distinguishability of different internal representations, our analysis explicitly concerns external representations.

We consider external problem representations of two kinds, both of which use a set of symbolic expressions to define the problem.

1. In a *sentential* representation, the expressions form a sequence corresponding, on a one-to-one basis, to the sentences in a natural-language description of the problem. Each expression is a direct translation into a simple formal language of the corresponding natural language sentence.
2. In a *diagrammatic* representation, the expressions correspond, on a one-to-one basis, to the components of a diagram describing the problem. Each expression contains the information that is stored at one particular locus in the diagram, including information about relations with the adjacent loci.

The fundamental difference between our diagrammatic and sentential representations is that the diagrammatic representation preserves explicitly the information about the topological and geometric relations among the components of the problem, while the sentential representation does not. A sentential representation may, of course, preserve other kinds of relations, for example, temporal or logical sequence. An outline may reflect hierarchical relations.

We consider problems presented in these two representations and ask about the relative difficulty of solution. We start with the assumption that the problem is solved using the given representation (sentential or diagrammatic). In fact, of course, one way to solve a problem in a poor representation is to translate it into a better one. One may be able to use the information in a verbal description to draw or image a diagram or use a diagram to infer verbal statements. But in order to understand what makes a good representation, we ask what is required for solution without such translation.

1. FORMALIZING THE QUESTION

To compare diagrams with sentences, we need to define what we mean by representation and by a "better" representation.

1.1 What Does a "Better" Representation Mean?

1.1.1 Informational and Computational Efficiency. At the core of our analysis lie the wholly distinct concepts of *informational* and *computational* equivalence of representations (Simon, 1978). Two representations are informationally equivalent if all of the information in the one is also inferable from the other, and vice versa. Each could be constructed from the information in the other. Two representations are computationally equivalent if they are informationally equivalent and, in addition, any inference that can be drawn easily and quickly from the information given explicitly in the one can also be drawn easily and quickly from the information given explicitly in the other, and vice versa.

"Easily" and "quickly" are not precise terms. The ease and rapidity of inference depends upon what operators are available for modifying and augmenting data structures, and upon the speed of these operators. When we compare two representations for computational equivalence, we need to compare both data and operators. The respective value of sentences and diagrams depends on how these are organized into data structures and on the nature of the processes that operate upon them.

1.1.2 Representations. A representation consists of both data structures and programs operating on them to make new inferences. The data structures we consider are node-link structures that include schemas employing attribute-value pairs. (Such structures have been called variously list structures, colored directed graphs, scripts, and frames. The differences, when there are any, are inconsequential for our purposes.) We can think of these structures as being represented in a list-processing language like LISP. Programs are represented as production systems. Each instruction has the form: $C \rightarrow A$, conditions C with associated actions A. The conditions are tests on some parts of the data structures; whenever such tests are satisfied by the appropriate data structures, the actions of the production are executed. Actions modify data structures, that is, they make and record inferences. Although the data structures we shall postulate are stored externally, on paper, the productions that operate on them are in the problem solver's memory.

Since data structures for a problem are complex, we must also provide for an attention management system that determines what portion of the data structure is currently attended to and can trigger the productions of the program. The nature of attention management depends crucially on the linkages provided in the data structure since this is the only information available for guiding shifts in attention.

Later we describe systems for solving physics and geometry problems. In these systems the productions contain knowledge of the laws and principles of physics or geometry, while the data structures contain knowledge about the particular problem being solved. This separation corresponds to the usual division of labor between the knowledge a problem solver holds in memory and the knowledge he or she commits to paper. In both cases we will also need to consider how attention is managed in keeping with the data structure.

In general the computational efficiency of a representation depends on all three of these factors (data structure, program, attention management) and on how well they work together. Whether a diagram (or any other representation) is worth 10,000 words depends on what productions are available for searching the data structure, for recognizing relevant information and for drawing inferences from that information (Simon, 1978). This point has been made again recently by Anderson (1984) in arguing that the distinction between representations is not rooted in the notations used to write them, but in the operations used on them.

1.2 Diagrams and Sentences

We are concerned with contrasting the operation of an inference program, human or computer, when using two different data structures with the same informational content. To assure informational equivalence, we start with a problem stated in natural language, translate it first into a sequence of more formal sentences, and then translate it into a diagram.

1.2.1 Data Structures. Producing a formal sentential representation from a verbal problem statement is relatively straightforward, using simple analogs of propositional coding like that of (Kintsch & Van Dijk, 1978). But how can we produce data structures that capture important features of diagrammatic problem representations? Consider a situation described by a sequence of ordinary English sentences (e.g., a verbal problem statement). Associate with each sentence a location (perhaps x and y coordinates in a two-dimensional reference frame). Now sentences are indexed by location—a program using this data structure can choose to "look" at a particular location and thereby access all information present there (i.e., all information elements indexed by that location). In short we make the following definitions:

- A data structure in which elements appear in a single sequence is what we will call a *sentential representation.*
- A data structure in which information is indexed by two-dimensional location is what we call a *diagrammatic representation.*

Of course, when a sentence or diagram is analyzed internally it may acquire different linkages, (e.g., a person may form a "mental picture" upon reading a sequence of sentences), but here we are concerned with the external representations.

1.2.2 Programs. The program operating on the data structure employs the following kinds of processes: (1) *Search* operates on the node-link data structures, seeking to locate sets of elements that satisfy the conditions of one or more productions. This process requires attention management. (2) *Recognition* matches the condition elements of a production to data elements located through search. Recognition depends on a match between the elements in the data structure and the conditions of the productions in the program. (3) *Inference* executes the associated action to add new (inferred) elements to the data structure.

How do sentential and diagrammatic representations, respectively, affect the three components of information processing mentioned above: search, recognition, and inference?

Search. Consider first the sentential data structure consisting of a simple list of items. Unless an index is manufactured and added explicitly to this list, finding elements matching the conditions of any inference rule requires searching linearly down the data structure. Furthermore, the several elements needed to match conditions for any given rule may be widely separated in the list. Search times in such a system depend sensitively upon the size of the data structure.

Search in a diagram can be quite different. In this representation an item has a location. If the conditions of an inference rule are only satisfied by structures at or near a single location, then the tests for satisfaction can all be performed on the limited set of structures that belong to the current location, and no search is required through the remaining data. Often part of the search process involves identifying multiple attributes of the same items, for example, that a rabbit is both white and furry. Therefore one computational cost of search is the ease with which such attributes can be collected. Of course, some search may be required to find the right location. (As an example of such a system, see the model of chess perception constructed by Simon and Barenfeld, 1969).

The two systems just described are not, in general, computationally equivalent. As we have described them, we would expect the second to exhibit efficiencies in search that would be absent from the first. Differences in search strategies associated with different representations are one major source of computational inequivalence.

Recognition. The effects of different representations on search are at least equaled by their effects on recognition. Human abilities to recognize information are highly sensitive to the exact form (representation) in which the information is presented to the senses (or to memory). For example, consider a set of points presented either in a table of x and y coordinates or as geometric points on a graph. Visual entities such as smooth curves, maxima and discontinuities are readily recognized in the latter representation, but not in the former.

Ease of recognition may be strongly affected by what information is explicit in a representation, and what is only implicit. For example, a geometry problem may state that a pair of parallel lines is cut by a transversal. Eight angles, four exterior and four interior, are thereby created but not mentioned explicitly. Moreover, without drawing a diagram it is not easy to identify which pairs of angles are alternate interior angles—information that may be needed to match the conditions of an inference rule. All of these entities are readily identified from a diagram by simple processes, once the three lines are drawn. The process of drawing the diagram makes these new inferences which are then displayed explicitly in the diagram itself. We will see later how this explicitness facilitates geometry proofs when a diagrammatic representation is used. Of course, the same information can also be inferred from the sentential representation, but these latter inference processes may require substantial computation, and the cost of this computation must be included in any assessment of the relative efficiency of the two representations.

Although our focus here is the contrast between diagrammatic and sentential representations, the human recognition process can often be specific to particular representations within these broad categories. For example, although the Roman, Cyrillic, and Greek alphabets are nearly isomorphic, a person who can read fluently in one of these alphabets cannot generally read the same information readily when it is transcribed into one of the others. The oral Serbian and Croatian languages are essentially identical, no farther apart than the English and American dialects of the English language. Serbian is written in the Cyrillic alphabet, while Croatian is written in the Roman alphabet. As a consequence, Serbs and Croats can read each other's newspapers only with the greatest difficulty.

The difficulty does not disappear for a person who knows both alphabets well, but each only the context of particular languages. For example, someone who reads Russian fluently when it is written in the Cyrillic alphabet and English in the Roman alphabet will have great difficulty in reading Russian transcribed to the Roman alphabet or English to the Cyrillic. (For similar effects of chess notation, see Chase and Simon [1973]).

It follows that we will be unable to recognize knowledge that is relevant to a situation and retrieve it from long-term memory if the situation is not presented in a representation matching the form of existing productions. This specificity of access is presumably remediable by training, but only at the cost of acquiring whole new sets of productions with condition sides

adapted to the specific representation that is employed. While the specificity favors no particular form of representation, it does severely limit the immediate substitutability of one representation for another.

Because a representation is useful only if one has the productions that can use it, we can readily understand the common complaint of physics professors that students "refuse to draw diagrams" or "don't appreciate their value." If the students lack productions for making physics inferences from diagrams, they may not only fail to "appreciate" the value of diagrams, but will find them largely useless.

Inference. In view of the dramatic effects that alternative representations may produce on search and recognition processes, it may seem surprising that the differential effects on inference appear to be less strong. Inference is largely independent of representation *if* the information content of the two sets of inference rules is equivalent—i.e., the two sets are isomorphs as they are in our examples. But it is certainly possible to make inference rules that are more or less powerful, independently of representation.

Examples of this phenomena are suggested by the everyday use of the verb "see" when no explicit visual processes are present. What is this metaphorical "seeing" and how might it connect to information-processing differences between sentential and diagrammatic representations? We speculate that this metaphor refers to inferences that are qualitatively like perceptually "seeing" in that they come about through productions with great computational efficiency. This efficiency might arise from low search and recognition costs, or from very powerful inference rules or from both.

Consider, for example, a physical chessboard which we would represent as a set of squares, each with an (x,y) location and connections to adjacent squares. With each square is associated the name of any piece on it. Any person can "see" on what squares the pieces lie and locate adjacent or nearby squares. These inferences come from primitive production rules that everyone has. But a chess expert may "see" things in the board not evident to the non-expert observer. For example, an important feature of a chess position is an open file: a sequence of squares that are vacant, running from the player's side of the board toward the opponent's side. In what sense is this "seeing" if everyone cannot see it? This recognition could be accomplished by a production that, upon noticing an open square in the first row, would trace this square to the "North" until a piece was encountered, then store this feature in memory, indexed to its location on the board. For the chess player who has such a production in his or her repertory, an open file is "seen," meaning that it is easily recognized. But for a person without such a set of productions only the individual unoccupied squares may be visible, when attended to, but not the open file. In exactly the same sense, a logician, presented with symbol structures for A and $A \rightarrow B$, may "see" the conclusion: B.

2. A SIMPLE EXAMPLE

To make concrete our comments about differences in computational efficiency for two problem representations containing identical information, we work here a simple example from physics. In this example we will see, for the diagrammatic representation, considerable computational advantages in search, plus a subtle advantage due to different needs for labeling. We leave to a second example the more complex, but probably more important, utility of diagrams in aiding recognition.

We begin our comparison of representations by analyzing a simple pulley problem from elementary mechanics. After stating the problem verbally, we provide a formal sentential encoding of the problem statement. Then we write, in a semi-formal notation, inference rules (productions) that embody the laws of statics relevant to solving the problem. Next we construct a diagrammatic interpretation of the system consisting of the problem data structure and the rules. Finally, we compare the computational efficiency of the sentential and diagrammatic representations. In this example, the principal differences between the representations derive from the amounts of search they demand, rather than from differences in recognition or inference.

2.1. The Given Data Structure and Program

Consider a problem given in the following natural language statements. We have three pulleys, two weights, and some ropes, arranged as follows:

1. The first weight is suspended from the left end of a rope over Pulley A. The right end of this rope is attached to, and partially supports, the second weight.
2. Pulley A is suspended from the left end of a rope that runs over Pulley B, and under Pulley C. Pulley B is suspended from the ceiling. The right end of the rope that runs under Pulley C is attached to the ceiling.
3. Pulley C is attached to the second weight, supporting it jointly with the right end of the first rope.

The pulleys and ropes are weightless; the pulleys are frictionless; and the rope segments are all vertical, except where they run over or under the pulley wheels. Find the ratio of the second to the first weight, if the system is in equilibrium.

Presented with the pulley problem, everyone we've observed reaches for pencil and paper, and draws a sketch of the situation somewhat like Figure 1. However, for the moment, we ignore the sketch, and proceed directly from the natural language statement of the problem.

We formalize and simplify the natural language sentences in this problem to produce the elements listed in Table 1a. The labels 1a, 1b, and so forth, refer to the sentence numbers above, with the decimal numbers labeling successively elements produced from a single sentence.

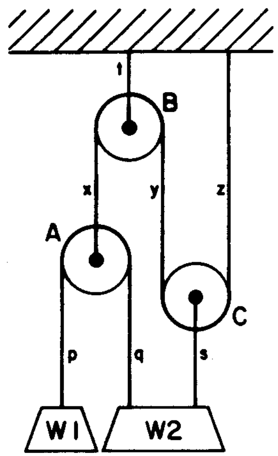

Figure 1. Diagram of the pulley problem.

The first propositions in each sentence associate labels with appropriate objects. We read subsequent propositions, for example, (1a.1) as: the weight *W1* is suspended from the rope *Rp*. We read proposition (1a.2) as: The rope, consisting of the left-hand segment *Rp* and the right-hand segment *Rq*, runs over (or under) the wheel of the pulley *Pa*. Sentences 2 through 3 are similarly translated. At the end, we add element (4.1) giving a specific simple value to the weight *W1* which can then be related to the value of weight *W2* in order to answer the question. We have captured the original problem statement, accurately we believe, in these formal elements that use the relations of *hangs, pulley-system,* and *value.*

In this data structure the various object labels (e.g., *W1,Rx*) are essential. It is only through these labels that one can infer that two different elements (e.g., 1b.1 and 3b.3) both refer to the weight *W2*. In the original problem statement these connections were provided (somewhat obscurely) by a combination of labels and anaphoric, numeric, and locational references (e.g., "supporting it jointly with the right end of the first rope").

We now turn from the given data structure to the program, composed of physics knowledge, that will act on it to solve the problem. This program consists of the following inference (or production) rules based on a few principles of statics. Pointed brackets (< >) indicate variables that refer to particular objects (e.g., ropes, pulleys).

P1. **Single-string support.** Given a weight of known value <*n*> and a rope <*R*> from which it hangs, if there is no other rope from which it hangs (indicated by the symbol ~),
then the supporting rope also have value (tension) <*n*> associated with it.

TABLE 1
Formal (a) Data Structure and (b) Program for the Pulley Problem

(a)

	(Weight W1) (Rope Rp) (Rope Rq) (Pulley Pa)
(1a.1)	(hangs W1 from Rp)
(1a.2)	(pulley-system Rp Pa Rq)
	(Weight W2)
(1b.1)	(hangs W2 from Rq)
	(Rope Rx) (Pulley Pb) (Rope Ry) (Pulley Pc) (Rope Rz)
	(Rope Rt) (Rope Rs) (Ceiling c)
(2a.1)	(hangs Pa from Rx)
(2a.2)	(pulley-system Rx Pb Ry)
(2a.3)	(pulley-system Ry Pc Rz)
(2b.1)	(hangs Pb from Rt)
(2b.2)	(hangs Rt from c)
(3a.1)	(hangs Rx from c)
(3a.2)	(hangs Rs from Pc)
(3b.3)	(hangs W2 from Rs)
(4.1)	(value W1 1)

(b)

P1. **Single-string support.** (weight <Wx>) (rope <Ry>)
(value <Wx> <n>) (hangs <Wx> <Ry>)
~(hangs <Wx> <Rx>)
→ (value <Ry> <W-number>)

P2. **Ropes over pulley.** (pulley <P>) (rope <R1>) (rope <R2>)
(pulley-system <R1> <P> <R2>) (value <R1> <n1>)
→ (value <R2> <n1>)

P3. **Rope hangs from or supports pulley.** (pulley <R1>) (rope <R1>) (rope R2>)
(pulley-system <R1> <P> <R2>) { (hangs <R3> from <P>) or (hangs <P> from <R3>) } (value <R1> <n1>) (value <R2> <n2>)
→ (value <R3> <n1 + <n2>)

P4. **Weight and multiple supporting ropes.** (weight <W1>) (rope <R1>) (rope R2>) (hangs <W1> <R1>) (hangs <W1> <R2>) ~(hangs <W1> <R3>)
(value <R1> <n1>) (value <R2> <n2>)
→ (value <W1> <n1> + <n2>)

P2. **Ropes over pulley.** If a pulley system <*P*> has two ropes <*R1*> and <*R2*> over it, and the value (tension) associated with <*R1*> is <*n1*>, then <*n1*> is also the value associated with rope <*R1*>.

P3. **Rope hangs from or supports pulley.** If there is a pulley system with ropes <*R1*> and <*R2*> over it, and the pulley system hangs from a rope <*R3*>, and <*R1*> and <*R2*> have the values (tensions) <*n1*> and <*n2*> associated with them,
then the value (tension) associated with <*R3*> is the sum of <*n1*> plus <*n2*>.

P4. **Weight and multiple supporting ropes.** If a weight <*W1*> hangs from both ropes <*R1*> and <*R2*>, but hangs from no other ropes, and the values <*n1*> and <*n2*> are associated with <*R1*> and <*R2*>, then the value associated with <*W1*> is the sum of <*n1*> plus <*n2*>.

Table 1b shows these rules, stated then in a formal notation matching that used for the data structure in Table 1a. In this notation symbols with pointed brackets can be matched to any symbol. A "~" indicates an element that may not appear in the current data structure if the conditions of the rule are to be satisfied.

These inference rules are based on two physics principles. (1) The total force on an object at rest is zero. (2) The tensions are equal in all parts of an ideal (massless, frictionless) rope, even if this rope passes over ideal (massless, frictionless) pulleys. The productions are directions for applying these principles in several specific situations relevant to our problem. Principles must be rewritten as active inference rules for any problem solving system. (Indeed students may well be unable to solve problems in part because they learn principles, and do *not* translate them into inference rules.) We might have made the list of productions shorter, but at the cost of more complex explanations.

Our system, now includes a data structure and a program (rules of inference). It remains to consider how the data structure is searched to find information the inference rules can use, how such information is recognized, and how the inferences are made.

2.2 Sentential Representation

If we consider the data structure in Table 1a as a sequential list of elements, then, by our earlier definition, this is a sentential definition of the pulley problem. Using the program in Table 1b, the problem can be solved by a simple non-algebraic procedure. Applying our physics inference rules, we develop sequentially values associated with objects in the problem, ultimately finding the desired value associated with weight *W2*. In English, the steps of this solution are:

1. Because weight *W1* (value 1) hangs from rope *Rp* and no other rope, the value associated with *Rp* is 1.
2. Because *Rp* (value 1) and *Rq* pass over the same pulley, the value of *Rq* is 1.
3. Because *Rp* and *Rq* have values 1, and the pulley *Pa* over which they pass is supported by *Rx*, the value associated with *Rx* is 1 + 1 = 2.
4. Because *Rx* (value 2) and *Ry* pass over the same pulley, the value of *Ry* is 2.
5. Because *Ry* (value 2) and *Rz* pass under the same pulley, the value of *Rz* is 2.

6. Because Ry and Rz have values 2, and the pulley Pc under which they pass is supported by Rs, the value associated with Rs is $2+2=4$.
7. Because weight $W2$ is supported by rope Rq (value 1) and rope Rs (value 4) and by no other ropes, its value is $1+4=5$.

This sequence of inferences is not logically complex, yet the reader has probably already discovered its psychological complexity. The major difficulty is that each inference requires locating simultaneously one or more recently developed values, together with sentences in the original data structure supporting the next inference. For example, consider the inference that the value associated with pulley Pa is 2 (see Figure 1 for relief). In the sentential representation, making this inference requires holding in memory the value associated with rope Rp (and perhaps also Rq), locating in the original data structure the relation (pulley-system $RpPaRq$), and perhaps also sentences identifying these objects appropriately as ropes and pulleys. Throughout the solution there is this continual need to hold on to values while searching for relational information. The search is particularly difficult for step 7 using production *P4*.

> **Weight and multiple supporting ropes.** If a weight <*W1*> hangs from both ropes <*R1*> and <*R2*>, but hangs from no other ropes, and the values <*n1*> and <*n2*> are associated with <*R1*> and <*R2*>, then the value associated with <*W1*> is the sum of <*n1*> plus <<*n2*>.

The reason is that information about two ropes, one mentioned much earlier, must be held in memory or recorded and then discovered by search; and then it must be verified that no other ropes support the weight.

To make these observations quantitative, we assume the following simple linear attention-control mechanism: Attention is initially at the first sentence. After each inference is made, the attention pointer is left at the beginning of the last sentence in which relevant information was found. New elements, added to the data structure are searched first, and in reverse order of their addition. Finally, if a production specifies that some element may *not* appear, then all data-structure elements must be searched to make sure this is the case.

Table 2 shows the seven steps of the solution using the formal representation developed earlier. The original elements of the data structure are listed at the left, and the seven steps, with the production applied, across the top. At the bottom are the new elements added to the data structure in that step. In each column, an *x* indicates an element that must be present for the inference rule to apply, and an *o* indicates an element that must be searched (assuming the simple linear search strategy outlined above) in order to verify that the production applies.

In step 1, for example, rule *P1*, **Single-string support**, is matched by the four *x*'d elements to conclude that the value associated with rope segment

TABLE 2
Solution to the Pulley Problem Using the Sentential Representation.

Original Element	Steps / Productions	P1	1 P2	2 P3	3 P2	4 P2	5 P3	6 P4
2. (Weight W1)		x	o	o	o			o
(Rope Rp)		x	x	x	o			o
(Rope Rq)		o	x	x	o			x
(Pulley Pa)		o	x	x	o			o
(hangs W1 from Rp)		x	o	o	o			o
(pulley-system Rp Pa Rq)		o	x	x	o		o	o
(Weight W2)		o		o	o	o	o	x
(hangs W2 from Rq)		o		o	o	o	o	x
2. (Rope Rx)		o		x	x	o	o	o
(Pulley Pb)		o		o	x	o	o	o
(Rope Ry)		o		o	x	x	x	o
(Pulley Pc)		o		o	o	x	x	o
(Rope Rz)		o		o	o	x	x	o
(Rope Rt)		o		o	o	o	o	o
(Rope Rs)		o		o	o	o	x	x
(Ceiling c)		o		o	o	o	o	o
(hangs Pa from Rx)		o		x	o	o	o	o
(pulley-system Rx Pb Ry)		o			x	o	o	o
(pulley-system Ry Pc Rz)		o				x	x	o
(hangs Pb from Rt)		o					o	o
(hangs Rt from c)		o					o	o
3. (hangs Rz from c)		o					o	o
(hangs Rs from Pc)		o					x	o
(hangs W2 from Rs)		o					o	x
(value W1 1)		x					o	o
New Elements								
1. (value Rp 1)			x	x				x
2. (value Rq 1)				x				o
3. (value Ry 2)					x			o
4. (value Rx 2)						x	x	o
5. (value Rz 2)							x	o
6. (value Rs 4)								x
7. (value W2 5)								o
Total Elements Searched: 138		25	7	20	19	14	22	31

Rp is 1. All other elements must also be searched, however, to assure that weight $W1$ is not supported by any rope other than Rp, one of the conditions of this inference rule. This kind of difficulty arises in using any inference rule based on finding all instances of a particular class, and examples in physics are common. (The net force is the sum of *all* forces acting on the system; energy is conserved if there are *no* dissipative processes.)

2.3. Diagrammatic Representation

A diagrammatic representation permits information at or near one locality to be accessed and processed simultaneously. Table 3 shows our pulley problem with such an indexing system. Each element contains one or more locations (labeled *a–m*).

To use this indexing by location, we need an attentional control mechanism with the following properties: (1) When one location is attended to, all information at that location is automatically available. (2) The system can switch attention to any location mentioned by the elements currently attended to. The attention switching can be handled by a simple inference rule of the form

(<anyobject> <location1> <location2>) (current-attention <location1>)

→

(change the last element to read: current-attention <location2>)

This mechanism is analogous to that proposed for the sentential representation, in which attention switched from one sentence to an adjacent one in the data structure. We define "adjacent" to mean that the two locations are mentioned in a single element of the data structure.

To aid in interpreting Table 3 and the solution based on it, Figure 2 shows the locations *a* through *m* roughly as they would appear in a diagram. The lines connect locations that are adjacent because they appear in the same element in the data structure. Attention moves from one location to another. At most there are two possible adjacent locations, and search is limited to these locations.

TABLE 3

Data Structure for the Pulley Problem, with Locations Indicated by Lower Case Labels a, b, c,

	(Weight a) (Rope b) (Rope c) (Pulley d)
(1a.1)	(hangs a from b)
(1a.2	(pulley-system b d c)
	(Weight e)
(1b.1)	(hangs e from c)
	(Rope f) (Pulley g) (Rope h) (Pulley i) (Rope j)
	(Rope k) (Rope l) (Ceiling m)
(2a.1)	(hangs d from f)
(2a.2)	(pulley-system f g h)
(2a.3)	(pulley-system h i j)
(2b.1)	(hangs g from k)
(2b.2)	(hangs k from m)
(3a.1)	(hangs j from m)
(3a.2)	(hangs l from i)
(3b)	(hangs e from l)
(4.1)	(value a 1)

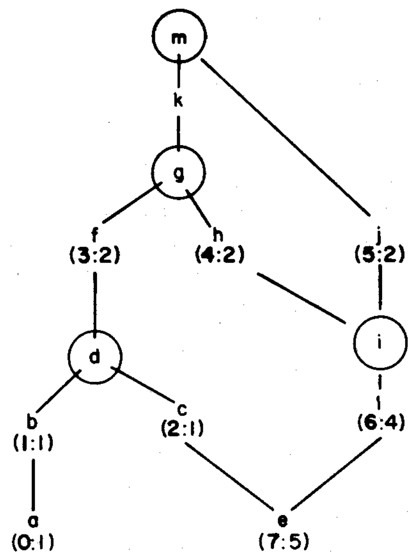

Figure 2. Schematic diagram of locations in the diagramatic representation. "Adjacent" (see text) locations are connected. Steps in the solution are shown in bold, with the value added to the location following the relevant step.)

Based on this data structure, and using the same physics program as before, the solution to the pulley problem is as follows:

1. A weight at *a*, with associated value 1, hangs from something at *b*, which is a rope. Therefore the rope at *b* has associated value 1. (**Single-string support,** No search, Attention now at *b*.)
2. The rope at *b* is in the pulley-system at *d* which also contains the thing at *c* which is a rope. Therefore the rope at *c* has associated value 1. (**Ropes over pulley.** Possible to make false attempt to apply **Rope hangs from or supports pulley** to (b c d f). Will fail because value at *c* is unknown. Attention now at *c*.)
3. The rope at *c* (value 1) is in the pulley-system at *d* which hangs from the thing at *f* which is a rope. The pulley-system at *d* also contains the thing at *b* which is a rope with value 1. Therefore the value associated with the rope at *f* is 2. (**Rope hangs from or supports pulley** Possible alternative step: Apply **Weight and multiple supporting ropes** to (c e l). Will fail because value at *l* is unknown. Attention now at *f*.)
4. The rope at *f* (value 2) is the pulley system at g which also contains the thing at *h* which is a rope. Therefore the value associated with the thing at *h* is 2. (**Ropes over pulley.** Attention now at *h*.)
5. The rope at *h* is in the pulley-system at *i* which also contains the thing at *j* which is a rope. Therefore the value associated with the rope at *j* is 2. (**Ropes over pulley.** Possible alternative step: Apply **Rope hangs from**

or supports pulley to (h i j). Fails because value at *j* is unknown. Attention now at *j*.)

6. The rope at *j* (value 2) is in the pulley-system at *i* which also contains the thing at *h* (a rope, value 2) and suspends the thing at *l* which is also a rope. Therefore the value associated with rope *l* is 4. (**Rope hangs from or supports pulley.** No search alternatives. Attention at *l*.)
7. The thing at *l* (a rope, value 4) suspends the thing at *e* which is a weight, also suspended from the thing at *c* which is a rope with associated value 1. Therefore the value of the weight at *e* is 5. (**Weight and multiple supporting ropes.** No search alternatives. Attention at *e*.)

We assume that change of attention to an adjacent location is a computationally easy process. Therefore, the searches and inferences indicated above comprise essentially all of the work of solving this problem. There are instances where a false effort may be made. These efforts all fail immediately, leaving the solver with only the correct possibilities. Compared with the 138 considerations of elements for the sentential representation (see Table 2), the diagrammatic representation provides a large saving.

Note also that the object labels in the sentential data structure (e.g., *W1,PA*) have been replaced by locations. Thus in addition to cutting search, a diagrammatic representation eliminates the overhead of keeping track of object labels.

Finally, this example illustrates that the program used with the sentential data structure is not immediately applicable to the diagrammatic data structure. The later assumes additional location-based changes in attention.

2.4. Computational Power in Inference Rules

Inference rules can be more or less powerful, whatever the representation (diagrammatic or sentential) on which they operate. In the example considered here, the physics inference rules (used in both representations) are far more powerful computationally than the physics principles on which they are based. For example, when **Single-string support** is used to infer the tension on *Rp* from the weight of *W1*, Newton's Third Law is being assumed implicitly: if the rope is in equilibrium, then the downward pull of *W1* must be exactly balanced by the tension in the rope. Also when **Rope hangs from or supports pulley** is used to infer the tension on *Rq* from the tension on *Rp*, Newton's Third Law is also being assumed implicitly: each small part of the rope is being pulled by a force exerted by the segment adjoining it on one side, which must be exactly balanced by the opposite force exerted by the segment adjoining it on the other side.

Thus, each production represents a "theorem" which could itself be derived by reasoning from the Laws of Motion. But the expert problem solver does not re-create this derivation each time. He or she simply incorporates it in a powerful operator. The analog in theorem-proving systems is a process for storing theorems that have been proved and using them directly as premises in subsequent proofs (Newell & Simon, 1956). In systems like the ones described here, instead of representing these theorems sententially, we embed them in operators represented as productions. When we say that reasoning is carried out "intuitively," we are often referring to just these kinds of procedural replacements of declarative knowledge. Information processing schemes for making such replacements are largely based on the formalism of adaptive production systems (Waterman, 1970). An early example of a program capable of converting declarative statements defining a problem into processes is UNDERSTAND (Hayes & Simon, 1974). This idea has subsequently been developed extensively by Neves and Anderson (1981) and others.

The benefits of more powerful inference rules are, of course, not limited to physics. For example, in the most austere forms of logic, where propositions are viewed as sentences in the predicate calculus (or other formal system), new information is obtained by applying truth-maintaining rules of inference (perhaps in the form of productions) to these sentences, and adding to the store the new sentences that are thus produced. This is the basic representation of all systems that accomplish problem solving by theorem proving. If the rules of inference are very limited in number and power (say, consisting of *modus ponens* alone), the system is likely to be highly inefficient from a computational standpoint. If other knowledge (e.g., physics) is also incorporated in the system in propositional form (as axioms), without addition of inference rules, the inefficiency is compounded.

A simple example will make the point clear. Suppose we have a system whose only inference rule is modus ponens:

If A, and (A *implies* B)$\rightarrow B$.

Suppose, further, that the system contains an axiom stating that xy is commutative; i.e., (xy implies yx). Now if (xy is true, then a search will reveal a match of xy and (xy implies yx) with the conditions of modus ponens, and yx will be produced.

On the other hand if the system contains the inference rule:

If $xy \rightarrow yx$,

then in the presence of xy, the condition of this rule is matched without further search, and xy will be produced immediately. Similarly, a system that has productions allowing it to assign the same tension to all segments of a rope will proceed more expeditiously than one that has to reach this result from general reasoning based on Newton's Third Law.

In general, powerful inference rules will contain information that is specific to a particular task domain: they will be task dependent. A good deal of skill acquisition in any domain can be attributed to the gradual acquisition of domain-specific inference procedures, specifically, acquisition of the corresponding productions, including both actions and the conditions that cause them to be evoked when relevant.

3. A MORE DEMANDING EXAMPLE

We now apply the analysis procedure illustrated in the previous problem to one that is more complex and interesting. Here we will see not only advantages to the diagram in search and labeling, but also a large and important role in facilitating recognition.

As in the preceding section, we will introduce an example, this time from geometry, and consider how it might be solved using a sentential and then a diagrammatic problem representation. In this case, again we find that the diagrammatic representation dramatically reduces search and removes considerable need for labeling. But we will find that, in contrast to the pulley example, the given data structure does not match the given program. Therefore to solve the problem at all, the problem solver must enhance the data structure in ways that prove considerably easier with diagrams than with sentences.

We first introduced the *given data structure and given program.* We then develop an enhanced sentential data structure and corresponding program that allow solution of the problem. Then we alter the data structure and program to see how the solution would proceed with a diagrammatic representation.

3.1. The Given Problem Representation

The given problem representation consists, as stated below, of a verbal statement of the problem (the given data structure), together with textbook statements of the definitions and axioms needed to solve the problem (the given program).

1. Two transversals intersect two parallel lines and intersect with each other at a point *x* between the two parallel lines.
2. One of the transversals bisects the segment of the other that is between the two parallel lines.
3. Prove that the two triangles formed by the transversals are congruent.

Figure 3 provides a diagram for this statement, but for now we will concentrate on the sentential representation.

The given program consists of the following axioms and theorems from geometry.

P1. **Definition of Bisector.** If something is a bisector, then it divides a line segment into two congruent segments.

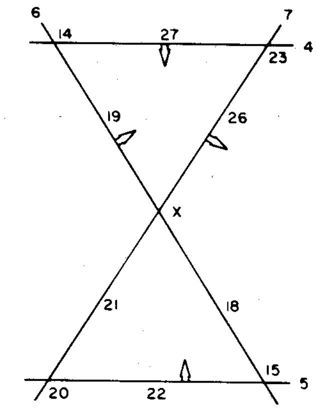

Figure 3. Diagram for the geometry problem. Labels correspond to Table 5.

P2. **Alternate Interior Angles.** If two angles are alternate interior angles, then they are congruent.

P3. **Vertical Angles.** If two angles are vertical angles then they are congruent.

P4. **ASA.** If two angles and the included side of one triangle are congruent to the corresponding two angles and included side of another triangle, then the triangles are congruent.

As in the pulley example, we rewrite the preceding natural-language statements into a sequence of propositions and a set of production rules, each using a limited set of predicates.

Formalizing the initial statement of the problem is straightforward and is given in Table 4a. For sentence 1 we use the element (transversal t1 l1 l2) to mean that line t1 is a transversal cutting parallel lines l1 and l2. Sentence 2 introduces a segment that is part of one of the transversals (t1), between the parallel lines and bisected by the other transversal t2. But then we are asked to prove congruent two triangles that have not been mentioned. This situation already contrasts with that of the pulley problem, where every necessary element was specified in the problem statement. In just interpreting the problem statement we already have significant difficulty in recognizing all mentioned elements.

The given program can also be formalized straightforwardly as shown in Table 4b. When we compare the given program to the given data structure, however, we see further recognition problems. The condition elements in the program mention many terms (e.g., alternative interior, vertical, sides) for which there is no mention in the given data structure. Therefore neither a sentential nor a diagrammatic representation of this problem can be successful until we enhance the data structure and modify the program so that

TABLE 4
The (a) Given Data Structure, and (b) Program in Semi-formal Notation.

```
                 (a)
          (line 11) (line 12) (line t1) (line t2)
(1a.1)    (parallel l1 l2)
(1a.2)    (transversal t1 l1 l2)
(1a.3)    (transversal t2 l1 l2)
(1a.4)    (intersect t1 t2 x)
(1a.5)    (between x l1 l2)

(2a.1)    (segment s1 of t1) (between s1 l1 l2)
(2a.2)    (bisector t2 s1)

(3a.1)    Prove: (congruent tr1 tr2)

                 (b)
(p  bisector
    (bisector <b> segment <s>)
    (forms-subsegments <b> <s> <s1> <s2>)
→
    (congruent <s1> <s2>)

(p  vertical
    (vertical <a1> <a2>) (angle <a1>) (angle <a2>)
→
    (congruent <a1> <a2>))

(p  alt-int
    (alternate-interior <a1> <a2>)
    (angle <a1> <a2>)
→
    (congruent <a1> <a2>))

(p  asa
    (triangle <t1> haspart <s1>) (side <s1>)
    (triangle <t2> haspart <s2>) (side <s2>)

    (triangle <t1> haspart <aa>) (angle <aa>)
    (triangle <t1> haspart <ab>) (angle <ab>)

    (triangle <t2> haspart <ac>) (angle <ac>)
    (triangle <t2> haspart <ad>) (angle <ad>)

    (congruent objects <ac> <aa>)
    (congruent objects <ad> <ab>)
    (congruent objects <s2> <s1>)
→
    (congruent <t1> <t2>))
```


the elements in one can be recognized and operated on by the other. In short, while *search* was the major difficulty in the pulley example, we will find that *recognition* of appropriate elements for inference is the major difficulty here.

3.2. Sentential Representation

Starting with the given data structure, we first develop an enhanced data structure, and then add conditions to the given program so that it can use this data structure.

3.2.1. The Perceptually Enhanced Data Structure. We call this new data structure "perceptually" enhanced because it includes exactly those elements that a person looking at a diagram like that in Figure 3 could immediately recognize. This enhanced data structure includes explicitly the points, segments, angles, and triangles implied by the given problem statement, and evident in the diagram in Figure 3. Although this data structure includes perceptually obvious elements, we initially represent it sententially, as the sequence of elements listed in Table 5. For simplicity we have omitted elements [e.g., (line 4)] that simply assert the existence of an object.

The initial elements shown in part (a) of Table 5 are those from the given problem. Parts (a) and (b) together form the perceptually-enhanced data structure and include all the points, segments, angles, and triangles that a human could readily identify in Figure 3.

This data structure is produced by a set of "perceptual" inference rules that operate on the given data structure to produce the elements of the perceptually enhanced data structure. The first set of these interpret various given data elements in terms of points and segments. Specifically, four productions make the following inferences that interpret the elements described at the left (given problem elements) by adding the elements described at the right. These latter elements interpret the former in terms of perceptually obvious items (segments, points, and lines).

Given element	**Segment and line elements added**
Segment between two lines:	a segment with endpoints that are intersection points with the two lines.
Point between two lines, lying on a line intersecting two lines	the point lies on the segment with endpoints at the intersection points.
Two lines intersect:	There is a point that lies on both lines.

A second set of perceptual productions identify segments, angles, and triangles, elements that are readily seen by a human and that appear in the given program of geometry theorems and definitions. The production inferring segments simply creates a segment with endpoints at any two points

TABLE 5
Elements in the Final Perceptually Enhanced Data Structure, including (a) Original Elements, (b) Elements added by Perceptual Productions, and (c) Elements Added by the Geometry Program.

(a)

(parallel lines 4 and 5)
(transversal 6 of 4 and 5)
(transversal 7 of 4 and 5)
(intersect 6 and 7 in x)
(between x lines 4 5)
(segment 16 of 6)
(segment 16 between 4 and 5)
(bisector 7 of 6)

(b)

(point x on lines 6 and 7)
(point 14 on lines 6 and 4)
(point 15 on lines 6 and 5)
(segment 16 from 14 to 15)
(point x on segment 16)
(segment 18 joining x and 15)
(segment 19 joining x and 14)
(point 20 on lines 5 and 7)
(segment 21 joining 20 and x)
(segment 22 joining 20 and 15)
(point 23 on 4 and 7)
(segment 24 joining 23 and 20)
(point x on segment 24)
(x is between 20 and 23)
(segment 26 joining 23 and x)
(segment 27 joining 23 and 14)
(point 23 in line 6 region 1)
(point 20 in line 6 region 2)
(point 23 in line 5 region 1)
(point 14 in line 5 region 1)
(point x in line 5 region 1)
(point 20 in line 4 region 1)
(point 15 in line 4 region 1)
(point x in line 4 region 1)
(point 15 in line 7 region 1)
(point 14 in line 7 region 2)
(angle 38: vertex 14 line 6 region 1 line 4 region 1)
(angle 39: vertex x line 7 region 2 line 6 region 1)
(angle 40: vertex 23 line 7 region 2 line 4 region 1)
(triangle 41 is: 23 x 14)
(segment 27 is in triangle 41)
(segment 26 is in triangle 41)
(segment 19 is in triangle 41)
(angle 42: vertex 15 line 6 region 2 line 5 region 1)
(angle 43: vertex x 7 line 1 region 6 line 2 region)
(angle 44: vertex 20 line 7 region 1 line 5 region 1)
(triangle 45 is: 20 x 15)
(segment 22 is in triangle 45)
(segment 21 is in triangle 45)
(segment 18 is in triangle 45)

(continued)

TABLE 5 (continued)

(c)

(alternate interior angles 44 40
  angle vertex 20 line 7 region 1 line 5 region 1
  angle vertex 23 line 7 region 2 line 4 region 1)
(vertical angles 39 43
  vertex x line 6 region 1 line 7 region 2
  vertex x line 6 region 2 line 7 region 1)
(alternate interior angles 42 38
  angle vertex 15 line 6 region 2 line 5 region 1
  angle vertex 14 line 6 region 1 line 4 region 1)
(sements 15 x and 14 x are congruent
  definition of bisector)
(congruent triangles
  segments 15 x and 14 x
  angles vertex 15 line 5 region 1 line vertex 14 line 4 region 1
  angles vertex x line 6 region 2 vertex x line 6 region 1)

that lie on the same line. In order to define an angle, however, we need a way of distinguishing the two sides of a line. Each pair of intersecting lines makes four angles depending on which sides of the lines are considered. Therefore three productions define "regions" on the two sides of a line. Initially one point (not on the line) is arbitrarily assigned to lie in region 1 (for that line), and the other to region 2. Thereafter, any point not lying on a line is assigned to one region or the other depending on whether or not a segment joining it to the original point crosses the line. (A segment "crossing" a line means the segment has a point in common with the line) The double arrows in Figure 3 point to region 1 for each line.

With regions defined, four unique angles can be defined for each intersection, although, in order to make the program run faster, we restricted it to angles lying in the two triangles. With angles and segments defined, a final perceptual production collects appropriate triples and identifies the two triangles.

The result is the perceptually enhanced data structure, including the original elements together with the elements shown in part (b) Table 5. Because (for good reason, as we shall see) Table 5 is hard to understand, Figure 3 is labeled to show in diagrammatic form most of the elements of Table 5.

In Table 5, explicit labels are necessary for each element, as they were for the sentential representation of the pulley problem. For example, we have a (triangle 41). It must be labeled, because that label, 41, provides the only connection to other elements concerning it (e.g., segment 27 is in triangle 41).

Looking again at the given program (theorems and definitions) described in Table 4b we find that many of the conditions on the left side of these inference rules still have no corresponding elements in our perceptually enhanced data structure. For example, there are no "alternate interior" or "vertical" angles. We therefore have a choice. We could further enhance the representation to include elements like "alternate interior" used by the program. Alternatively, we can leave the data structure as it is and make changes in the program so that it can recognize elements in the existing data structure. We choose the latter to make a conservative distinction between perceptually obvious elements and elements that must be developed by reasoning. We argue that all elements currently in the perceptually enhanced representation (including points, segments, angles, and triangles) are perceptually obvious to any educated person looking at a diagram like that in Figure 3. Although we could add patterns like alternate interior angles, it is not clear that such patterns are obvious to everyone, and we will develop our program to reason about them explicitly.

In summary, we now have a data structure including all the primitive, visually obvious elements in the problem situation. It remains (a) to modify the given program so that it can interact with this data structure, and (b) to see how these modified inference rules solve the problem.

*3.2.2. **The Program.*** Table 6 lists the four inference rules, corresponding to the original given program, but replacing elements undefined in the existing data structure with their definitions in terms of elements already defined. For example, in the alternate-interior-angles rule, the single predicate "alternate-interior" has been replaced by a set of elements that state explicitly what alternate interior means in the current data representation in terms of "parallel," "between," "region," "side," and so on.

Thus, the angles have different regions with respect to the transversal (they are "alternate"). The angles have overlapping regions with respect to the parallel lines (they are "interior" to these lines). Similar additions are needed for the other inference rules as given in Table 6. These inference rules add to the data structure in Table 5 the final five elements in part (c) which include the desired conclusion that the triangles are congruent.

In summary, Table 5 shows the data structure after the problem is solved. This structure includes the original problem elements (part a), those added by perceptual inference rules (part b), and those added by geometric inference rules (part c), the latter modified to refer to primitive elements like points and segments rather than constructs like alternate interior angles.

3.2.3. Costs of Recognition and Search. The perceptual elements in part (b) of Table 5 and the modified conditions of the geometry inference rules in Table 6 capture what is necessary to allow the given geometry program to interact with the given data structure. Producing this data structure and

TABLE 6

Formal Geometry Program in Terms of Primitive Elements (points, lines, and segments).

```
(p bisector
   (bisector by <b> segment <s>)                          ;original condition
     ;<s> bisected by <b>

   (line <b>)                                             ;definition of bisector
   (point <p> on <b>)
     ;line <b> with <p> on it

   (segment <s>)
   (point <p> on <s>)
     ;segment <s> with <p> on it

   (segment <s> endpoints <p1> <p2>)
     ;endpoints of <s>
   (segment <s1> endpoints <p1> <p>)
   (segment <s2> endpoints <p2> <p>)
     ;subsegments of <s> formed by <p>

   - (congruent <s1> <s2>)
     ; this inference hasn't already been made
→
   (congruent <s1> <s2>)

(p vertical
   (angle <a1> vertex <v> line <l1> region <n1>
                          line <l2> region <n2>)
   (angle { <a2> <> <a1> } vertex <v>
               line <l1> region { <n3> <> <n1> }
               line <l2> region { <n3> <> <n1> })
   - (congruent object <a1> <a2>)
→
   (make congruent> objects <a1> <a2>))

(p alt-int
   (parallel <l2> <l3>)

   (between> point <x> <l2> <l3>)
   (region side <n1> offline <l2> containspoint <x>)
   (region side <n3> offline <l3> containspoint <x>)
     ;these are interior angles

   (angle <a1> vertex <v1> line <l2> side <n2>)
         line <l1> side <n1x>)
   (angle { <a2> <> <a1> } vertex { <v2> <> <v1> }
         line <l3> side <n3>)
         line <l1> side { <nly> <> <n1x> } )
     ;these are alternate angles

   - (congruent object <a1> other <a2>)
→
   (congruent> objects <a1> <a2>))
```

(continued)

TABLE 6 (continued)

```
(p asa
   (triangle <t1> object <s1>)
   (segment <s1> point <pa> other <pb>)
   (triangle { <t2> <> <t1> } object <s2>)
   (segment <s2> point <pc> other <pd>)
     ;corresponding sides

   (triangle <t1> object <aa>)
   (angle <aa> vertex <pa> line <la> side <na>)
   (triangle <t1> object <ab>)
   (angle <ab> vertex <pb> line <lb> side <nb>)
     ;corresponding angles

   (triangle <t2> object <ac>)
   (angle <ac> vertex <pc> line <lc> side <nc>)
   (triangle <t2> object <ad>)
   (angle <ad> vertex <pd> line <ld> side <nd>)
     ;corresponding angles

   (congruent object <ac> other <aa>)
   (congruent object <ad> other <ab>)
   (congruent object <s2> other <s1>)
   - (congruent object <t1> other <t2>)
     ;congruences
→
   (make congruent objects <t1> <t2>))
```

program is the cost of being able to *recognize* useful elements in the data structure. Neglecting the costs of modifying the program (probably a considerable cost to students of geometry), the costs of developing the perceptual elements are evident in Table 5 which includes 40 elements in part (b) (compared with just 5 in part (c) added by the geometry program). The computer-implemented production system language (Ops5, see Brownston, et al., 1985) used to produce this table uses a slightly different system of elements. In this form the given data structure has 15 elements, the perceptual productions add 78 more in 41 steps, and the geometry productions just 10 more in five steps. The overt costs of recognition are clearly large.

Even with a data structure adequate for recognition, *search* is still problematic. If we imagine a sentential representation consisting of the list of elements in Table 5, then to match each element in an inference rule, we must search through the entire list until we find it, an average of $48/2=24$ tests.

In short, in this simple geometry problem, recognition of the conditions for an inference rule, and search for matching conditions are both significant problems in a sentential representation. We discuss now how these difficulties are diminished by using a diagrammatic representation.

3.3. Diagrammatic Representation

In constrast to the pulley problem considered earlier, the major efficiency difference between the sentential and diagrammatic representations in this problem arises from differences in *recognition* rather than from differences in *search* (although here search is also important). The reason is that the original given statement (and its corresponding formal representation) does not include elements that can be recognized by the inference rules in the given program (i.e., in the theorems and definitions of geometry). Therefore, to solve the problem in the sentential representation, we must first do a great deal of perceptual enhancement of the data structure (addition of points, lines, segments, etc.). We chose to limit this enhancement to just those elements that a human viewer can obviously detect unambiguously on a diagram. To make the program work with this data structure required also enhancing the inference rules so that high-level elements like "alternate interior angles" were replaced (using the definition of alternate interior) by sets of elements that appear in the data structure. When we turn to the diagrammatic representation, the perceptual enhancement of the given representation is done by processes that are computationally very cheap, i.e., the processes of drawing and viewing the diagram. Since this phase comprises most of the work in the sentential representation, we already have a major saving, but there are two other advantages for the diagrammatic representation.

The first is a computational difference in *recognition*. With the perceptual work done, the (enhanced) geometry rules can readily recognize their conditions. In the sentential representation, the perceptual work of recognition is explicit and extensive; in the diagrammatic representation, it is automatic and easy.

In addition to these large differences in recognition, however, there are also considerable search differences. How can we take the data structure in Table 5 and the program in Table 6 and interpret it as a diagrammatic representation? As in the case of the pulley problem, let us assume that we have a perceptual control mechanism that allows the system to have instant access to all information at a given location; specifically, we assume that if two elements refer to the same point, then they are at the same location. The spacing in Table 6 groups together (for the first three rules) elements at a single location. For these rules, information is always present at just one or two locations. If we make the primitive assumption that there is no guidance for locating the first object in a group, then search will be required through an average of $(48/2)=24$ elements. But thereafter no further search is required for any elements in a location group. The number of elements that must be checked to satisfy a production referring to n locations is therefore $n*(48/2)$, assuming also that each group is found independently of the last. In contrast, the number of elements checked in the sentential representation is $N*(48/2)$ where N is the number of elements in the production. Therefore the grouping of about 10 elements per production in Table 6 into one or two location groups reflects the search advantage of a diagrammatic representation. The fourth rule (ASA) contains two location groups, each corresponding to one triangle, and each related to three point locations joined by segments. The rule is written to emphasize the pairs of congruent parts, rather than the locations. By our criterion that "same location" means "contains a common point," this rule covers four locations in two adjacent pairs. Furthermore, the points in the two locations are related in perceptually predictable ways (although this organization is not captured by our formal representation).

These estimates of relative search difficulty are extremely conservative. If a person can search for two or three things simultaneously, one search can find the conditions of a production with spatial groupings. Furthermore, two separate location groups in a production are often at predictably related locations (e.g., at opposite ends of a transversal).

3.4. Summary and Comparison of Representations

The major difference in a diagrammatic representation, we believe, is difference in recognition processes. We have seen that formally producing perceptual elements does most of the work of solving the geometry problem. But we have a mechanism—the eye and the diagram—that produces exactly these "perceptual" results with little effort. We believe the right assumption is that diagrams and the human visual system provide, at essentially zero cost, all of the inferences we have called "perceptual." As shown above, this is a huge benefit. If the geometry problem is given verbally, without a diagram, all of these elements must be constructed explicitly (or perhaps in part by some internal imaging process). It is exactly because a diagram "produces" all the elements "for free" that it is so useful.

In this problem, as in the pulley problem, using a diagram removed the need for labelling the objects. Because there are so many objects in the geometry problem, this is a considerable saving. In the formal OPS5 system, all objects are defined in terms of points, that is, in terms of their locations. Informally, it is common to construct a geometry proof simply by making corresponding congruent elements in a diagram that need not include any labels at all. The labels are needed only when a conventional proof is written—and that is, in fact, a sentential representation.

Finally the inference rules contain information that can be used to guide attention when the desired information is not at the current location. For example, suppose the system is attending to one segment on a line while applying the definition of bisector. Within the inference rule is the information that the second segment can be found on the same line and at one end or the other of the currently attended segment.

Although in the geometry problem there are large differences in the recognition and search processes required by the sentential and diagrammatic

representations, there are no differences in inference processes. In each case four geometric inferences are required to solve the problem.

We can, however, imagine at least two ways in which geometric inference rules can be more or less powerful. First, these rules might incorporate powerful recognition capabilities, so that in one step, without any need for search, an entire high-level pattern could be recognized. Examples of such patterns include alternate interior angles, vertical angles, and ASA patterns for triangles. Second, the textbook theorems and definitions could be broken down into special cases that apply recognition. This has already been done for our definition of bisector. The original definition states that a line or segment bisects a segment *if* and *only if* the subsegments produced are congruent. We have divided this single statement into two parts, one of which we have used to make the inference that if there is a bisector, then the segments produced are congruent. In other cases, the traditional inference-rule statements are quite specific. For example, rules for alternate interior and alternate exterior angles could readily be stated as a single general inference rule, but most geometry texts separate them. In all cases, the special-case rules require less search and recognition for their application.

Second, as was also illustrated with the pulley problem, we can imagine inference rules that make more powerful inferences. For example, one rule might apply the alternate-interior-angle theorem twice and also conclude that two sets angles are therefore congruent, eliminating the need for applying the vertical-angles theroem. As observed earlier, independent of representation, such powerful inference rules would increase computational efficiency in either a diagrammatic or a sentential representation.

4. "ARTIFICIAL" DIAGRAMS

The examples considered so far describe systems in real spaces (the pulley system) or ideal space (the geometry problem). Diagrams often, however, do not describe an actual spatial arrangement. Examples include graphs and vector diagrams. These "artificial" diagrams provide a test of the properties we have argued are central to the utility of diagrams. Are these simply properties of the spatial world that are then portrayed in diagrams of that world? Or are they properties having psychological utility in problem solving? If the latter is true, then they should also be seen in the diagrams depicting relations among non-spatial variables that people use in solving problems. To explore this issue, we consider briefly three further examples.

4.1. Graphs in Economics

Diagrams were the principal tool for reasoning in economics throughout much of its modern history. In the preface to the first edition of his famous *Principles of Economics,* which dominated the scene for 60 years after 1890, Alfred Marshall had this to say:

> It is not easy to get a full view of continuity in this aspect without the use of mathematical symbols or diagrams. The use of the latter requires no special knowledge, and they often express the conditions of economic life more accurately, as well as more easily, than do mathematical symbols. . [E]xperience seems to show that they give a firmer grasp of many important principles than can be got without their aid; and that there are many problems of pure theory, which no one who has once learned to use diagrams will willingly handle in any other way. (Marshall, 1890)

The following is a simple and typical example of their use (see Figure 4):

The abcissa of the diagram represents the quantity of a commodity that is produced or demanded, and the ordinate, the price at which that quantity will be supplied or purchased.

The line D is a demand curve, indicating the quantity that would be purchased at each price. It slopes downward to express the assumption that the lower the price, the larger the quantity that will be demanded.

The lines S and S' are supply curves, indicating the quantities that would be offered on the market at each price. They slope upward to express the assumption that the higher the price, the larger the quantity that will be offered. Notice that perceptually obvious features (e.g., slope) are used to represent important information.

We can immediately read off from the diagram the equilibrium price and quantity, as the ordinate and abcissa, respectively, of the intersection, E, of S and D (or E', of S' and D). At this price, supply and demand are equal. The information needed to make this inference is located together and again uses perceptually obvious features (e.g., intersections).

Next, we ask what the effect on the equilibrium will be of imposing a manufacturer's tax of k dollars on a unit of the commodity if the initial supply curve is S and the initial equilibrium E. The price at which any given quantity will be supplied will now be k dollars higher than the price at which it would have been supplied before, moving the supply curve from S to S', and the equilibrium from E to E'.

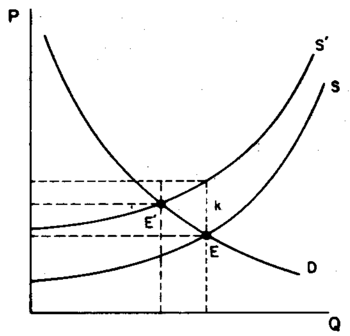

Figure 4. Graph used in an argument from economics.

Again, we see immediately that the equilibrium price will increase, but by less than the amount of the tax, while the quantity exchanged will decrease. By considering demand curves of different slopes in the neighborhood of the equilibrium we would also see directly that the increase in price would be less, and the decline in quantity greater, the flatter the demand curve (the more "elastic" the demand).

As in our other examples, the great utility of the diagram arises from *perceptual enhancement,* the fact that it makes explicit the relative positions of the equilibrium points, so that the conclusions can be read off with the help of simple, direct perceptual operations. It would be a useful exercise for the reader to undertake to reach the same conclusions from the problem statement using either a sentential representation or an algebraic one.

4.2. Free Body Diagrams in Physics

Figure 5a shows a physical situation commonly studied in elementary physics. The solver might be asked to relate the masses of A and B. Figure 5b shows the artificial "free-body" diagrams that are always recommended for solution of such problems (Halliday & Resnick, 1970, Sears, Zemansky & Young, 1981, Heller & Reif, 1984). There are two diagrams, one for car A and one for car B. $\mathbf{F}_{AB}$ is the force on car A due to car B, and $\mathbf{F}_{BA}$ is the force on B due to A. These forces have equal magnitude, but opposite directions. Does this conventionally used, artificial diagram have the properties we have argued are important in our earlier examples?

1. *Localization.* Below each of the free-body diagrams in Figure 5b are the equations central to solving this problem. Each equation comes from one free-body diagram. Information used together is grouped together in the free-body diagrams. This localization is achieved by the conventional rule for free-body diagrams—they include all the forces *on* a particular system. This need not have been the case. Indeed novices often draw their version of "free-body" diagram that places, for example, the force F_{BA}, the force on B due to A, in the A diagram rather than in the B diagram, thus violating the locality property for the free-body diagram.
2. *Minimal labeling.* The only important functions of labels in Figure 5 is to relate parts of the real-world diagram unambiguously to the force diagram. Even this relation is often made unnecessary by experts who simply draw the force diagrams on top of the objects in the original picture.
3. *Use of perceptual enhancement.* In this example, the acceleration of the train is towards the right. Note that the skilled solver has drawn the first force diagram with the right-directed force larger than the left-directed force. This allows a quick, probably perceptual, check of whether the force diagram is consistent with the physical situation. In the vertical

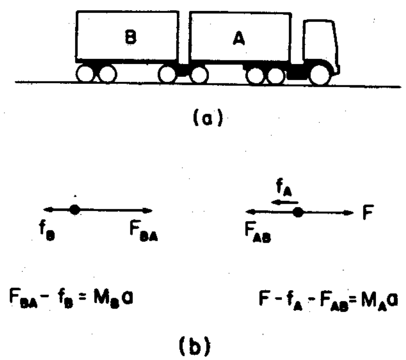

Figure 5. (a) Two cars (masses m_A and m_B) attached to an engine that exerts a force F. (b) Corresponding free-body diagrams and equations of motion for the cars.

direction, the forces are of equal magnitude, and so there is no acceleration. In the horizontal direction, the left-directed forces are shorter or non-existent, and so acceleration is towards the right. Such quick checks can be important, especially in more complex problems.

4.3. Energy-Level Diagram

Figure 6 shows the energy *E* for a hydrogen atom as a function of the strength *B* of a magnetic field surrounding it. When the magnetic field is 0, there are just two possible energies for the atom. But when a field is applied, these levels "split" into four. One can readily see that, for large values of *B* levels *I* and *III*, and levels *II* and *IV* have a constant separation, but this is not the case for levels *III* and *II*. The horiziontal lines represent various possible "energy states" of the atom for the field strength indicated by the dotted line. Consider the properties of this diagram using the following criteria for utility of diagrams developed earlier.

1. *Locality*. This diagram is used to guide computation of the energy released when there is a transition of an electron from one energy state to another. Various possible transitions are indicated in Figure 6 by vertical arrows between the states. As the atom makes this transition, its energy is reduced by the amount *E1-E2*, where *E1* is the energy of the higher state and *E2* the energy of the lower state. This then is the amount of energy released (as a photon). Again we find that all information needed to make this calculation is readily available in predictable locations with the lines representing the two energy levels.
2. *Minimizing labeling*. The labels in Figure 6 serve only to let us talk about the elements there. They are not essential to any of the inferences made.

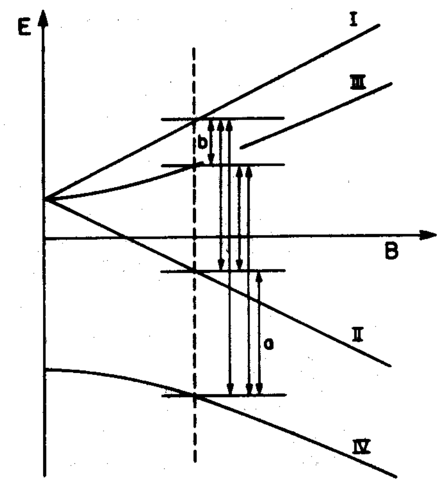

Figure 6. Graph showing energy levels in a hydrogen atom.

3. *Use of perceptual enhancement*. The diagram represents perceptually the qualitative knowledge of the relative sizes of the energy levels. It is also drawn at least roughly to scale, so that the relative size of energies released by transition can be checked against the diagram. For example, transition *a* clearly releases more energy than does transition *b*.[1]

4.4. A Comment on Mental Imagery

In this paper we have limited ourselves to the use of diagrams as a form of external memory. Our only references to psychological processes (other than the subject-matter inference rules) have been to rather obvious phenomena of perception: that people can focus attention on part of a diagram, and that they can detect cues there (e.g., simple objects like pulleys or angles), and use them to retrieve problem-relevant inference operators from memory. These assumptions agree with everything that has been learned in the past two decades about expert performance.

We cannot end our discussion however, without a few comments on mental imagery—the uses of diagrams and other pictorial representations that are not stored on paper but are held in human memory. We have no new evidence to offer, but we do offer the speculation that mental images play a role in problem solving quite analogous to the role played by external diagrams (and that this role is also played in the two memories, internal and external, in concert). By this we mean that mental images, while containing substantially less detail than can be stored in external diagrams, have similar

[1] Figure redrawn from (Feynman et al., 1966).

properties of localization of information and can be accessed by the same inference operators as the external diagrams. This implies also that the creation of a mental imagine (for instance, from a verbal description) employs inference processes like those that make information explicit in the course of drawing a diagram.

Thus, when we draw a rectangle and its two diagonals, the existence of the point of intersection of the diagonals is inferred automatically—the point is created on the paper, accessible to perception. In exactly the same way, when we imagine a rectangle with its two diagonals, we imagine ("see") the point of intersection in memory.

In this paper, we have represented external diagrams symbolically as list structures, and the inference processes as list processes in a production system language. These representations and processes could equally well be interpreted as denoting mental images and imagery processes in the brain. But much difficult psychological research, the exact character of which we can only dimly perceive, will be required to test this hypothesis. Until then it must remain a speculation, albeit one that appears consistent with such psychological evidence as exists.

5. CONCLUSION

For external problem representations, we have provided a simple distinction between sentential representations, in which the data structure is indexed by position in a list, with each element "adjacent" only to the next element in the list, and diagrammatic representations, in which information is indexed by location in a plane, many elements may share the same location, and each element may be "adjacent" to any number of other elements. While certainly not the complete story on this important representational issue, this simple distinction lets us demonstrate the following reasons why a diagram can be superior to a verbal description for solving problems:

- Diagrams can group together all information that is used together, thus avoiding large amounts of search for the elements needed to make a problem-solving inference.
- Diagrams typically use location to group information about a single element, avoiding the need to match symbolic labels.
- Diagrams automatically support a large number of perceptual inferences, which are extremely easy for humans.

None of these points insure that an arbitrary diagram is worth 10,000 of any set of words. To be useful a diagram must be constructed to take advantage of these features. The possibility of placing several elements at the same or adjacent locations means that information needed for future inference can

be grouped together. It does not ensure that a particular diagram does group such information together. Similarly, although every diagram supports some easy perceptual inferences, nothing ensures that these inferences must be useful in the problem-solving process. Failing to use these features is probably part of the reason why some diagrams seem not to help solvers, while others do provide significant help (Paige & Simon, 1966, Larkin, 1983).

The advantages of diagrams, in our view, are computational. That is diagrams can be better representations not because they contain more information, but because the indexing of this information can support extremely useful and efficient computational processes. But this means that diagrams are useful only to those who know the appropriate computational processes for taking advantage of them. Furthermore, a problem solver often also needs the knowledge of how to construct a "good" diagram that lets him take advantage of the virtues we have discussed. In short, the ideas we have presented, not only provide an explanation of why diagrams can be so useful, but also provide a framework for further study of the knowledge required for effective diagram use.

REFERENCES

Anderson, J.R. (1978). Arguments concerning representations for mental imagery. *Psychological Review, 85,* 249–277.

Anderson, J.R. (1984). *Representational Types: A Tricode Proposal* (Technical Report #82-1). Office of Naval Research, Washington, D.C.

Brownston, L., Farrell, R., Kant, E., & Martin, N. (1985). *Addison-Wesley Series in Artificial Intelligence. Programming Expert Systems in OPS5.* Reading, MA: Addison Wesley.

Chase, W.G., & Simon, H.A. (1973). The mind's eye in chess. In W.G. Chase (Ed.), *Visual information processing.* New York: Academic.

Feynman, R.P., Leighton, R.B., & Sands, M. (1966). *The Feynman Lectures on Physics.* Figures 4-12, pp. 11-12. Reading, MA: Addison Wesley.

Hadamard, J. (1945). *The psychology of invention in the mathematical field.* Princeton, NJ: Princeton University Press.

Halliday, D., & Resnick, R. (1970). *Fundamentals of physics.* New York: John Wiley & Sons.

Hayes, J.R., & Simon, H.A. (1974). Understanding written task instructions. In L.W. Gregg (Ed.), *Knowledge and cognition.* Hillsdale, NJ: Erlbaum.

Heller, J., & Reif, F. (1984). Prescribing effective human problem-solving processes: Problem description in physics. *Cognition and Instruction, 1,* 177–216.

Kintsch, W., & Van Dijk, T.A. (1978). Toward a model of text comprehension and production. *Psychological Review, 85,* 363–394.

Larkin, J.H. (1983). *Mechanisms of effective problem representation in physics* (C.I.P. 434). Department of Psychology, Carnegie-Mellon University, Pittsburgh, PA.

Marshall, A. (1890). *Principles of economics.* New York: Macmillan (Preface to the first edition (1980), page x).

Neves, D., & Anderson, J.R. (1981).Knowledge compilation: Mechanisms for the automatization of cognitive skills. In J.R. Anderson (Ed.), *Cognitive skills and their acquisition.* Hillsdale, NJ: Erlbaum.

Newell, A., Shaw, J.C., & Simon, H.A. (1959). Empirical Explorations of the Logic Theory Machine. In *Proceedings of the Western Joint Conference on Artificial Intelligence.* WJCAI,

Paige, G., & Simon, H.A. (1966). Cognitive processes in solving algebra word problems. In B. Kleinmuntz (Ed.), *Problem solving*. New York: Wiley.

Pylyshyn, Z.W. (1973). What the mind's eye tells the mind's brain: A critique of mental imagery. *Psychological Bulletin, 80,* 1-24.

Sears, F.W., Zemansky, M.W., & Young, H.D. (1981). *Physics.* Reading, MA: Addison-Wesley.

Simon, H.A. (1978). On the forms of mental representation. In Savage, C.W. (Ed.), *Minnesota studies in the philosophy of science. Vol. ix: Perception and cognition: Issues in the foundations of psychology*. Minneapolis: University of Minnesota Press.

Simon, H.A., & Barenfeld, M. (1969). Information Processing Analysis of Perceptual Processes in Problem Solving. *Psychological Review, 76,* 473–483.

Waterman, D. (1970). Generalization learning techniques for automating the learning of heuristics. *Artificial Intelligence, 1,* 121-170.

Wertheimer, M. (1959). *Productive Thinking*. New York: Harper & Brothers. Einstein is quoted on p. 228.