Appears in *Diagrammatic Representation and Inference*, Alan Blackwell, Kim Marriott and Atsushi Shimojima, Editors, Lecture Notes in Artificial Intelligence 2980, Berlin: Springer-Verlag, 2004, pp. 151-165.

An Architecture for Problem Solving with Diagrams[1]

B. Chandrasekaran[+], Unmesh Kurup[+], Bonny Banerjee[+], John R. Josephson[+] and Robert Winkler[++]

[+]Dept of Computer & Information Science
The Ohio State University
Columbus, OH 43210, USA
{Chandra|kurup|banerjee|jj}@cis.ohio-state.edu, winkler@arl.army.mil

[++]U. S. Army Research Lab
2800 Powder Mill Rd
Adelphi, MD 20783 USA

Abstract. In problem solving a goal/subgoal is either solved by generating needed information from current information, or further decomposed into additional subgoals. In traditional problem solving, goals, knowledge, and problem states are all modeled as expressions composed of symbolic predicates, and information generation is modeled as rule application based on matching of symbols. In problem solving with diagrams on the other hand, an additional means of generating information is available, viz., by visual perception on diagrams. A subgoal is solved opportunistically by whichever way of generating information is successful. Diagrams are especially effective because certain types of information that is entailed by given information is explicitly available – as emergent objects and emergent relations – for pickup by visual perception. We add to the traditional problem solving architecture a component for representing the diagram as a configuration of diagrammatic objects of three basic types, *point*, *curve* and *region*; a set of perceptual routines that recognize emergent objects and evaluate a set of generic spatial relations between objects; and a set of action routines that create or modify the diagram. We discuss how domain-specific capabilities can be added on top of the generic capabilities of the diagram system. The working of the architecture is illustrated by means of an application scenario.

1 Introduction

Diagrams are ubiquitous as aids to human reasoning. They are potentially useful representations either when the domain is intrinsically spatial, as in the case of maps for route planning, or when there is a mapping from domain information to two-dimensional space that preserves certain properties, as in the case of pie charts, where

[1] This paper was prepared through participation in the Advanced Decision Architectures Collaborative Technology Alliance sponsored by the U.S. Army Research Laboratory under Cooperative Agreement DAAD19-01-2-0009.

the ratios of wedge angles are the same as ratios of certain quantities in the target domain. Some of the ways diagrams help are not unique to diagrams as spatial representation. For example, like many external representations, diagrams are a memory aid. They also enable certain abstractions to stand out, by omitting details that are not relevant. Maps work this way – in contrast to an aerial photo of the same region of space, only certain objects are shown, and many of them are iconized rather than spatially veridical, such as a church icon to represent the location of a church, instead of the actual spatial extent of the church. But diagrams enable deeper use of their spatiality in comparison to textual forms external representation. Diagrams help by enabling perception to pick up information corresponding to what one might call *emergent objects* and *emergent relations*. An emergent relation is exemplified in Fig.1. Similarly, if one were to draw a diagram consisting of two intersecting curves, visual perception can pick up a number of emergent objects, such as their intersection points and the curve segments. If the curves represent roads, a problem solver can make use of these new objects in, say, route planning. To the extent that recognition of such objects and relations is easy for the perceptual system of a particular agent, the agent can avoid complex sequences of reasoning or calculation that might otherwise be required in deliberative problem solving to obtain the same information.

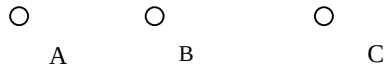

Fig. 1. The figure is generated from description, "A is to the left of B, and B is to the left of C," and relation "A is to the left of C" is emergent, and can be picked up human perception.

We distinguish the work in this paper from two other research streams in diagrammatic reasoning. In one stream, the focus is on algorithms that operate on representations of diagrams to produce new diagrams that represent some information of interest. The authors of such algorithms usually contrast them with symbolic reasoning, and propose that their algorithms exploit the structure of space in their data structures, e.g., local operations on pixel arrays. In another stream, the focus is on the logical aspects of diagrams, in particular on how proofs may be thought of as transformations of diagrams e.g., Venn diagrams as they are often used in informal set theory. The goal is to give diagrams and their manipulations the same logical status as sentences and their transformations in traditional logic. Our aim, however, is to understand reasoning and problem solving activities that combine representation and inferences in both diagrammatic and symbolic modes so that different subproblems are solved by the mode that is best suited for them. Research in this area may make use of algorithms and ideas in the other streams. For example, some of the inferences from diagrams – we call them Perceptual Routines and discuss them later in the paper –- may make use of algorithms in the first stream, and some decisions to transform diagrams in certain ways may relate to symbolic reasoning. However, the focus in our research is an architecture that best deploys each of the modes.

Our goal in this paper is to propose and develop a representation for diagrams that is as domain-independent as possible. The goal is to provide a diagrammatic represen-

tation that can help a computational problem solver, much as external diagrams help human problem solvers. We intend to present a certain intuition about a diagram as an abstract, but still spatial, notion. We also extend existing goal-directed problem-solving frameworks in such a way that diagrammatic representations as we conceive them can be used as a basic part of problem state representations, complementing the traditional predicate representations. Though we hope that much of what we say is consistent with important aspects of the human cognitive architecture, our motivation is not to propose a psychological theory, but rather to expand the options available to designers of AI and human-machine systems.

2 Problem Solving: Information Generation for Goals and Subgoals

Problem solving as goals and subgoals. On a standard account in AI of problem solving (see [1] for the most evolved version), problem solving is a *means-ends*, or a *subgoaling*, process. The agent uses general and domain knowledge to decompose the goal into subgoals, subgoals into further subgoals, and so on, as needed. The task of problem solving then is to solve enough subgoals so that the main problem can be solved. Each important step in problem solving then is to solve a (sub)goal, or to decompose it into additional subgoals. For our current purpose, it is useful to think of goals in information terms: achieving a goal (whether the main or a subgoal) corresponds to having information of a certain description. Solving a (sub)goal corresponds to inferring the information needed from background general and domain knowledge, information defining the problem, and information generated in earlier steps in problem solving. Thus, as each subgoal is solved, it may provide information that may help infer solutions to other subgoals, and finally the main goal. As information is generated, it results in change in the representation of problem state.

Let us consider an information generation step in the above problem solving process, say I_2 from I_1. For I_1 to be support the generation of information I_2, I_1 must entail I_2 in some sense. Information generation usually involves *representing* information in some representation framework. In general, if I_1 is represented by a representation R_{I_1} in a representation system $\mathbf{R}$, not all the information entailed by I_1 will be explicit in R_{I_1}. The task then becomes one of transforming R_{I_1} into a representation R_{I_2} such that R_{I_2} makes I_2 explicit. However, as has been often remarked in the literature on diagrammatic reasoning (e.g., [2]), representation systems differ in the degree to which the representation of information I also explicitly represents other information entailed by I. In diagrammatic reasoning, two different representation systems are employed:

- *Information generation by rule-based transformation of symbol strings.* In most standard accounts, the goals, subgoals, and information are all represented in terms of symbolic predicates, such as the familiar `On(A,B)` in the blocks domain. In this representation, only I is explicit in R_I. Entailed information is generated by apply-

ing rules to such expressions composed of predicates. For our purposes, AI representations such as frames, rules, etc., still belong in this category. The relevant point is that the representation is composed of predicates, and explicit rules determine what information can be generated from what predicate expressions.

- *Information generation by perception from spatial representations*. Diagrams (more generally visual representations) are a class of representation whose instances automatically make some of the entailed information available, provided suitable pickup capabilities are present. The medium constrains the representation in such a way that it is not possible to represent I without representing a range of entailed information. In human use of diagrams, visual perception can pick up some of the entailed information, as in the example in Fig. 1

The architecture that we describe has both of these representational and inference generation modalities, working together. The problem solving and the representation are *bi-modal*.

Flexible integration of information generation modes. As mentioned earlier, as information is generated, the problem state representation changes. When diagrams are used, the change may be to either or both diagrammatic and predicate components. Change to the diagram may involve adding or deleting diagrammatic elements, or changing spatial specifications of the existing elements. As indicated, such changes give rise to emergent objects and relations in the diagram, which information may be picked up by perception in problem solving steps further down the line. In general, a source of power in problem solving is the way information generated at one step creates potential for further information generation in future steps. In diagrammatic reasoning, the combination of the two types of information generation operations can be especially powerful: information might be obtained in one step from the diagram by perception; the information so obtained might be combined in the next with other information in memory to support applying an inference rule to information in symbolic form, which might result in changes to the diagram, which might in turn give rise to emergent objects and relations that can be picked up by perception, and so on. Diagrams add a whole new level of power to this opportunistic goal-directed behavior that is problem solving.

Diagrams can be internal or in a computer memory. Notwithstanding the debates about the nature of mental images, there is a functional level at which for many problems a mental image plays a role similar to that of diagrams on external surfaces. That is, problem solving proceeds *as if* information is being obtained from a mental image, and the process of problem solving is similar to what happens when the diagram is external. The essential steps in the reasoning of a human problem solver are the same whether or not she draws an external diagram such as in Figure 1, or imagines some version of it. In either case, a step in the problem solving sequence is to assert that the problem solver *sees* that A is to the left of C in the instance the solver has created or imagined, and to generalize the conclusion from the instance to the general case. This sequence of steps is in marked contrast to the following sequence: Represent the given information in Figure 1 as `Left-of(A,B)` ^ `Left-of(B,C)`, represent knowledge about transitivity as $\forall x \forall y$ {`Left-of(x,y)` ^ `Left-of(y,z)`} → `Left-of(x,z)`}, and apply the predicate system's inference capability to generate the information, `Left-of(A,C)`. In the former, spatial rela-

tional information is "seen" and asserted for an instance, and implicitly or explicitly generalized. In the latter, the reasoning starts with a general rule – in this case about the transitivity of Left-of – , instantiates it, and infers the required information by applying the rule. Whether the diagram is external or the agent claims to solve it by means a mental image, the structure of reasoning is essentially similar, but different from that corresponding to the symbolic predicate-rule-based inference process. In this sense, whatever the true underpinning of a mental image, it is a functional equivalent of an external diagram with respect to set of information extraction operations.

3 Diagrammatic Representation

A representation system is characterized functionally by (i) a repertoire of representational primitives, (ii) ways of creating/modifying a representation to represent given information by composing primitives, and allowed ways of transforming them to make desired information explicit, and (iii) ways of reading information that is explicit in a representational instance. How these three functional characterizations apply to predicate-representations is well-known. For our diagram representation system, dubbed DRS, we describe below the basic set of primitive objects, creation/modification operations that we call *Action Routines* (ARs), and information reading capabilities that we call *Percepual Routines* (PRs). The diagram so constructed, along with a set of ARs and PRs, is intended to be functionally equivalent to an agent obtaining information from a diagram – external or internal – during problem solving. Defining DRS functionally avoids many of the contentious issues surrounding whether a computational representation is "really" a diagram, just as defining a mental image functionally at the end of the previous section was a way to avoid getting entangled with the question of whether mental images are "really" images.

Three Dimensions of Perceptual Experience. We view the experience corresponding to visually perceiving a scene as having three dimensions. The first and most basic dimension is that of seeing the world as composed of objects – the figure, a set of objects, is discriminated from ground. The objects are perceived as shapes [3], an experience like that evoked by a Henry Moore abstract sculpture. With diagrams, this dimension corresponds to perceiving it as composed of objects – or figures – as 2-d shapes. The second dimension is that of *seeing as*, e.g., recognizing an object as a telephone, or a figure in a diagram as a triangle. The recognition vocabulary comes in a spectrum of domain-specificity. The third dimension is relational – seeing that an object is to the left of another object, taller than another, is part of another, etc. The result of these latter two dimensions of perception is not the spatiality of the objects as such, but symbols and symbol structures that assert and describe, albeit in a vocabulary using spatial terms.

3.1 Diagrammatic Objects

Our basic representation of a diagram is intended to capture the first dimension in the preceding discussion – in this case 2-d shapes of diagrammatic objects. Perceptual

routines then act on this to recognize object types and relations. Conversely, action routines may construct or modify aspects of the diagram, as we will discuss shortly.

A Diagram is a pair (**I**, DDS) where **I** is the image, defined as a specification – implicit or explicit – of intensity values for points in the relevant regions of 2-D space, and DDS is the *Diagram Data Structure*, which is a list of labels for the diagrammatic objects in the image; associated with each object label is a specification of the subset of **I** that corresponds to the object. A diagrammatic object can be one of three types; *point*, *curve*, and *region*. Point objects only have location (i.e., no spatial extent), line objects only have axial specification (i.e., do not have a thickness), and region objects have location and spatial extent. The labels are internal to DDS. External labels such as A, B, etc., in Figure 1 are additional features of the objects that may be associated with them.

I is any description from which a specification of intensity values for the relevant

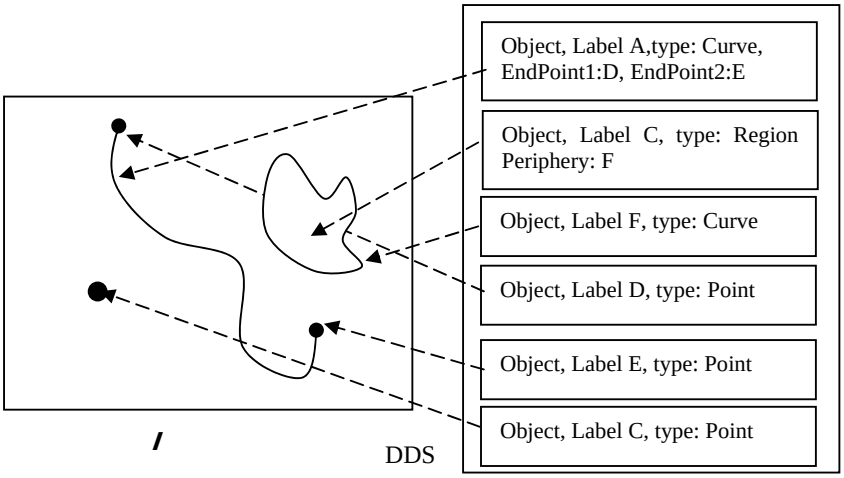

Fig. 2. A DRS for a simple diagram composed of a curve and a region. However, note that there are other objects: distinguished points such as end points, the closed curve defining the perimeter of the region, etc.

points in the 2-D space can be obtained. It can be extensional, such as a specification of a 2-D array and the intensity values of the elements of the array. It can be intensional, such as algebraic expressions or equations that describe point, line or region objects. Fig. 2 is an example DRS for a simple diagram.

There are several important distinctions between an external diagram as an array of marks on paper – a raw image – and DDS. The first distinction is that DDS is the result of a figure-ground discrimination already made, the array *interpreted* as objects, along with their spatial specifications. The second distinction is that DDS involves a particular type of abstraction. In external diagrams, points and curves are really regions, with some convention that signals to the user which of the regions are to be interpreted as abstract points and curves, and which to be taken as regions. DDS incorporates the abstraction. Thus, in Figure 1, point A is shown as a circular region. The

DDS corresponding to point A, however, has as its spatial specification just the intended coordinate of the point. Similarly, a curve in a physical diagram would have a non-zero thickness, whereas the corresponding representation in DRS will be as an abstract curve, i.e., as a type of object with no thickness. Next, those marks or visual elements in physical diagrams that have purely iconic roles are not represented spatially in DDS, even though in the physical diagram they have spatial extent. Examples are alphabetical symbols that are often placed next to diagrammatic elements to name them, icons such as that of a church on a map at a location to indicate the type of building, coloring that has an iconic role, e.g., red and blue objects standing for enemy and friendly units in a battlefield map, and the cross-hatching of a curve on a map that might indicate that it is a railroad. Object representations in DDS have fields that can hold these abstract symbols; e.g., the railroad will be represented by a curve in DDS, with an appropriate symbol in a field of the object representation. The problem solver would interpret it as a railroad. Next, DDS is generic. Information that is domain-specific, e.g., that a certain curve is a defensive perimeter, is simply treated as abstract symbolic information, and incorporated as labels attached to the objects to be interpreted by the problem solver using domain-specific conventions. At the level of DDS, there is no recognition of a curve as a straight line, or a region as a polygon. These are recognitions that will be made further downstream by perceptual routines. Also, how curves and regions are actually represented in a computational realization is not relevant at the level of DDS. For example, in a specific implementation, a curve may in fact be represented as a sequence of line segments (as we do in our implementation), or as a set of pixels in an array-based representation. These are representational details below the level of the abstraction of DDS, in which such an object is still a curve.

Finally, in addition to specification of spatial extent, external diagrams may have visual aspects that are of representational significance beyond iconic. For example, grayscale intensity gradients across a region object in a diagram may represent the values of a real variable in a domain, such as a weather map in which the grayscale intensity at a point represents the temperature at that location. Perceptions such as "the intensity is least in this subregion," and "the intensity increases from the center to the periphery of the region" convey information in an assertional form that may be useful for making further inferences. The DDS framework can be extended to capture the gradation of intensity values over space for the various objects.

Diagrams are constructed by placing diagrammatic objects on a 2-D surface in specific configurations to represent specific information. DDS will initially consist of the labels of objects so placed and their spatial specifications. As objects in DDS are created, deleted or modified, new objects might emerge, objects may change their spatial extents, and existing objects might be lost. Perceptual routines will make these determinations and DDS will be updated to reflect these changes. DDS mediates the interaction of the problem solver with **I**. For example, given a question such as "Is A to the left of C?", the information in DDS is used to identify the image descriptions for A and C as arguments for the perceptual routine `Left-of(X, Y)`. In principle, the same image **I** might correspond to different DDS's depending on which subsets are organized and recognized as objects, such as in the well-known example of a figure that might be perceived as a wine glass or profiles of two faces.

In keeping with the functional notion, DDS can represent only a diagrammatic instance (not a class), and its spatial specifications must be complete, just as in the case of an external diagram, and unlike in the case of predicate-based representations that can provide partial specification of a situation, or of classes of situations (such as *for all*). This does not mean that the agent is committed to all the spatial details in the DDS – it is the task of the problem solver to keep track of what aspects are intended to represent what.

Representation in Memory. Just as problem state is bimodal, we think representation of knowledge in memory is in bimodal as well, for agents that use diagrammatic representations. We have route fragments in mind that we can bring to our consciousness as diagrams, and compose these fragments into larger routes. DRS fragments can be stored in memory, just as knowledge in symbolic predicate form.

3.2 Perceptual Routines

Ullman [4] proposed a set of elementary visual routines, such as visual search, texture segregation and contour grouping, that might be composed to perform complex visual tasks. Our notion of perceptual routines[2] (PRs) is based on a similar notion of composable and extensible primitives, but more oriented to the needs of problem solving with diagrams. Because of our interest in generic objects, aspects of our proposals are intended to be domain-independent.

A PR takes specified objects in a diagram as arguments and produces information corresponding to a specific perception. PRs can be categorized into two classes, the first set producing objects with spatial extent as their output, and the second set producing symbolic descriptions. The first includes PRs that identify emergent objects -- points, curves, and regions -- that are created when a configuration of diagrammatic objects is specified or modified, and similarly objects that are lost. The PRs of the first class are domain-independent in that the point, curve, region ontology is completely general. The PRs of the second class produce symbolic descriptions belonging to one of three kinds: (i) specified properties of specified objects (e.g., curve C has length of m units), (ii) relations between objects (e.g., point A is in region R, curve C1 is a segment of curve C2, object A is to the left of object B, values of the angles made by intersection of curves C1 and C2), and (iii) symbols that name an object or a configuration of objects as an instance of a class, such as a triangle or a telephone.

The PRs of the second class come in different degrees of domain specificity. Properties such as length of curve, area of a region, and quantitative and qualitative (right, acute, obtuse, etc.) values of angles made by intersections of curves are very general, as are *subsumption relations* between objects, such as that curve C1 is a segment of curve C2. Relations such as Inside(A,B), Touches(A,B), and Left-of(A,B) are also quite general. PRs that recognize that a curve is a straight line, a closed curve is a triangle, etc., are useful for reasoning in Euclidean geometry, along with relations such as Parallel (Line 1, Line2). The PRs of the second class are open-ended in the sense that increasingly domain-specific perceptions may be conceived: e.g., an L-shaped re-

[2] We call them perceptual routines because one of our long-term goals is to extend the notion of the cognitive state to multiple perceptual modalities, and vision is just one modality.

gion, Half-way-between(point A, point B, point C). Our goal for the current set of PRs is what appears to be a useful general set, with the option for additional special purpose routines later on. The following is a list of perceptual routines of different types that we have currently identified and implemented as being generally useful.

Emergent Object Recognition Routines. Intersection-points between curve objects, region when a curve closes on itself, new regions when regions intersect, new regions when a curve intersects with a region, extracting distinguished points on a curve (such as end points) or in a region, extracting distinguished segments of a curve (such as those created when two curves intersect), extracting periphery of a region as a closed curve. Reverse operations are included – such as when a curve is removed, certain region objects will no longer exist and need to be removed.

Object Property Extraction Routines. Length-of(curve) (this will return a number), Straightline(Curve) and Closed(Curve) will return True or False. Additional property extraction routines can be added as needed.

Relational Perception Routines. Inside (I1,I2), Outside, Left-of, Right-of, Top-of, Below, Segment-of (Curve1, Curve2), Subregion-of (Region1, Region2), On (Point,Curve), and Touches(object1, object2), angle-value(curve AB, curve BC). Subsumption relations are especially important and useful to keep track of as object emerge or vanish.

Abstractions of groups of objects into higher level objects. Objects may be clustered hierarchically into groups, such that different object abstractions emerge at different levels.

When to invoke PRs. As a diagram is created or modified, and given a repertoire of PRs, when a specific PR should be applied is a practical computational issue. It would be impractical to apply all of them and generate all the perceptions. In our current implementation, the only PRs that are immediately applied as a diagram is created or changed is the set of emergent object routines, with all other PRs applied only in response to a problem solving need. Even in the case of emergent objects, the number of these objects can grow exponentially. For example, as a region is intersected by two curves, one horizontal and one vertical, we not only have four quadrant regions, NE, NW, SE and SW, but also regions that are composed of them, such NW+SW+SE and NE+NW. In addition, for each of the curves, we have a large number of segments as curve objects and emergent point objects as well. So our strategy has been to detect all *first-order* emergent objects, with other objects identified on an as-needed basis. By first-order, we mean emergent objects that do not have other emergent objects of the same type as subparts. In the example, we just considered, the four quadrant regions would be first-order emergent regions, and each of the curves will give rise to four first-order curve objects.

Domain-specificity. Perceptions may be domain-specific because they are of interest only in some domains, e.g., "an L-shaped region." They may also be domain-specific in that they combine pure spatial perception with domain-specific, but non-spatial, knowledge. For example, in a military application, a curve representing the motion of a unit towards a region might be interpreted as an attack, but that interpretation involves combining domain-independent spatial perceptions – such as extending the line of motion and noting that it intersects with the region – with non-spatial domain knowledge – such as that the curve represents the motion of a military unit, that the region's identity is as a military target belonging to a side that is the enemy of the

unit that is moving, etc. In our current implementation, it is the task of the problem solver to combine appropriately the domain-independent perceptions with domain-specific knowledge to arrive at such conclusions, but in application-dependent implementations of the architecture, some of these perceptions might be added to the set of perceptual routines.

3.3. Action Routines

The problem solving process may modify the diagram – create, destroy or modify objects. Typically, the task – the reverse of perception in some sense – involves creating the diagram such that the shapes of the objects in it satisfy a symbolically stated constraint, such as "add a curve from A to B that goes midway between regions R1 and R2," and "modify the object O1 such that point P in it touches point Q in object O2." Constructing or modifying diagrammatic elements that satisfy such constraints involves a set of Action Routines parallel to Perception Routines. Again similar to PRs, ARs can vary in generality. Deleting named objects that exist in the diagram, and adding objects with given spatial specifications, e.g., Add point at coordinate, Add curve <equation>, etc., are quite straightforward. Our ARs include translation and rotation of named objects for specified translation and rotation parameters. Because of the specific needs of our domain, we have ARs that find one or more *representative paths* from point A to point B, such that intersections with a given set of objects are avoided. A representative path has the right qualitative properties (avoid specific regions, e.g.), but is a representative of class of paths with those qualitative properties, members of which may differ in various quantitative dimensions, such as length. The set of ARs includes routines that adjust a path in this class to be shorter, longer, etc., in various ways. Extending lines indefinitely in certain directions so that a PR can decide if the extended line will intersect with an object of interest is one that we need in our domain. Other researchers have found specific sets of ARs that are useful for their tasks, such as the AR in [5], "Make a circle object that passes through points A, B, and C." An AR that we haven't yet used, but we think would be especially valuable, is one that changes a region object into a point object and vice versa as the resolution level changes in problem solving, such as a city appearing as a point in a national map, while it appears as a region in a state map.

Underspecification of spatial properties of objects. More generally, each of the PRs can be reversed and a corresponding AR imagined. For example, corresponding to the PR Inside(R1, R2) is the AR, "Make region R2 such that Inside(R1,R2)," (assuming region R1 exists); and corresponding to Length(curve C1) is the AR, "Make curve C1 such that Length(C1) < 5 units." In most such instances, the spatial specification of the object being created or modified is radically underdefined. Depending on the situation, random choices may be made, or certain rules about creating of objects can be followed. However, the problem solver needs to keep track of the fact that the reasoning system is not committed to all the spatial specification details. For example, in the example in Fig. 1, the points are placed in response to the Left-of information regarding A, B and C. The problem solver needs to keep track of the fact that there is no commitment to the actual distances between the points in the figure.

4 The Architecture

Figure 3 is a schematic of the problem solving architecture that is the subject of current experimentation. The architecture is a functional diagrammatic problem solver even in the absence of an external diagram. This is because the perceptual routines work off DRS, and DRS may be constructed either from an external representation (by that part of visual perception that breaks the image array into objects, or manually, as in our experiments), from memory, or a combination.

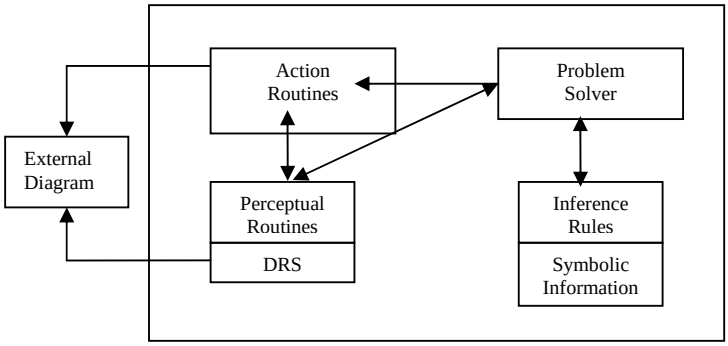

Fig. 3. The architecture showing the perceptual and the symbolic parts as co-equal information sources.

The problem solver sets up subgoals, and the representational system – the symbolic or the diagrammatic component – that can provide the solution responds. The problem solver may create or modify the diagram during problem solving. The modification is achieved by activating action routines that interact with the external representation, or it can simply change DRS – functionally corresponding to imagining changes. If a modification corresponds to a commitment to some part of a solution, say, a route in a route-planning problem, it would be appropriate to change the external representation; otherwise, such as when solutions are being hypothesized before being critiqued, a change to DRS is sufficient. These decisions are under the control of the problem solver.

Figure 3 doesn't bring this out explicitly, but the problem state is really bimodal. Using a route planning example for illustration, suppose a certain state corresponds to, say, "Road A being taken eastwards." The symbolic component of this information – e.g., `route-segment: (Road A, direction: East, Property1(A): 2-lane, Property2(A): toll, ….)`— is in the Symbolic Information box of the architecture. The corresponding diagrammatic component, the curve object corresponding to Road A, and the other roads and intersection points will be in the DRS part. If the next subgoal is to calculate the cost of travel on Road A, the calculation will be done by the symbolic box by making use of additional information about costs from its background knowledge. On the other hand, if the next subgoal is to decide if point B is on the way on Road A going east, the DRS and the perceptual routines will be best positioned to solve this subgoal.

A research goal is to see how far we can go in making the perceptual component co-equal with the symbolic component in problem solving. To take an example, there is no intrinsic reason why a problem solving goal has to be represented solely in symbolic terms, i.e., in terms of predicates that we wish to hold in the goal state, as in `ON(A,B)` in the Blocks world. In principle, goals may also be partly or wholly described in spatial terms, such as "Make an object whose shape is <this>," where the referent of <this> is a shape, represented in DRS. Similarly, the output of a perceptual routine may be perceptual, such as when emergent objects are recognized. Or it may be symbolic, as when a relational routine returns a yes or no answer (such as to the question "Is point A to the left of region B?"), or a predicate (such as "A is inside B"). This information will be added to the store of symbolic information, and thus made available for supporting additional inferences. Conversely, the result of a symbolic rule application, e.g., "Attach component A to component B at point C," might result in a DRS component being created representing the spatial configuration corresponding to this situation. This would make it possible for perceptual routines to be applied to the representation in DRS.

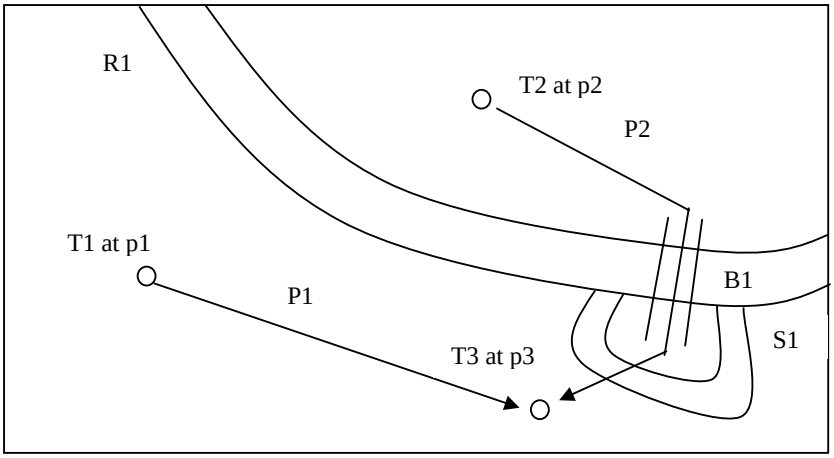

Fig. 4. A scenario. T1, T2 and T3 are tanks, and p1, etc. are the corresponding locations. R1 represents a river, B1 a bridge, S1 is a sensor field and P1 and P2 are paths.

5 Application Example

The US Army's All Source Analysis System or ASAS provides a picture of the enemy to its forces. The ASAS is continually updated by incoming intelligence reports such as sightings of enemy vehicles at different locations. Integrating incoming data into the system is done by human operators. One of the areas we are applying DRS to is in automating this process. In the example below, we intend to highlight the contri-

butions that DRS makes rather than give the reader a comprehensive account of the problem solving process involved.

The current situation is represented in Fig. 4. T1, T2 and T3 are three tank sightings at locations p1, p2 and p3 at time instants t1, t2 and t3 respectively. Initially ASAS has reported sightings of the two tanks T1 and T2. Sometime later, a new report of a sighting of a tank at location p3 comes in. At this point, tank T3 could be the result of either T1 or T2 having moved to the new location or a completely new tank that was previously unreported. The system tries to eliminate some of these possibilities in order to narrow down the hypothesis space. It tries to construct a path from positions p1 and p2 to p3 over which a tank could have traveled within the given time span. This process involves a complex interaction between the problem solver and the diagrammatic reasoner.

Given the map, the emergent object recognition routine of the DRS identifies three subregions of the region object corresponding to the river. The problem solver (PS) retrieves from its knowledge base the information that a bridge is a region over which a river can be crossed. It marks the river sections as barriers, and the section covered by the bridge as crossable. These labels are domain-specific inferences made by the PS using domain knowledge. When the report of the new sighting comes in, the problem solver's task is to figure out whether T3 is either of T1 or T2 that has moved since their original sightings, or a completely new tank. PS calls the DRS function – one of the action routines – to construct representative paths (see Sec. 3.3), one from p1 to p3 and the other from p2 to p3, that avoid the barrier regions. The PS calculates if the representative paths are short enough for the tank to have moved from their starting points to p3 within the given time span; if yes, the corresponding tank is accepted as a potential candidate. If the problem solver decides that the constructed path is too long, it can go back and ask the DRS to construct shorter paths, and the process repeated. If the distance covered by the shortest possible path is still too long to be covered in the time allotted, the PS discards the corresponding tank as a possible candidate. Let us look at another reasoning sequence. In the case of Tank 2, any path has to go over the bridge and hence over the sensor field at its foot. The problem solver asks the DRS whether the path from p2 to p3 intersects any regions labeled S (for sensor – DRS doesn't know anything about sensors, but the objects in DRS have labels to mediate the interactions between PS and DRS). When the DRS comes back and reports that the path goes over region S1, the PS goes back to the DRS asking it to find an alternative path from P2 to P3 (it can do this by simply setting the sensor field as a barrier). If no such alternative path exists, the PS can conclude that if T2 has moved to p3, then it should have gone over the sensor field and there should a corresponding report about it somewhere in the database. The absence of such a report while not completely ruling out the possibility of T3 being T2 (due to the possibility of sensor failure), it does substantially decrease the possibility and in turn, increases the possibility that T3 is either T1 or a new tank.

6 Related Work

This work is an evolution of earlier general proposals made in [6]. The following is a sampling of work in diagrammatic research on mixing the symbolic and perceptual modes. Geometry theorem proving has especially been a task that has seen much research in this area, with [5] and [7] being well-known recent examples. Diagrammatic and symbolic representations in device behavior understanding are considered in [8]. While there are many points of contact between the representation of the diagram in these systems and DRS, our overall goal is a generic architecture for bimodal problem solving in contrast to the above research whose proposals for primitive objects and perceptions are tailored to the needs of their domains. In contrast, we have little to say about the specific ways in which diagrams work in their domains, but our more general representations may be specialized to their needs. Reference [9] describes a program that recognizes domain-specific objects (such as tea kettles) from diagrams composed of straight line segments, and extracts spatial relations between the objects, but does not deal with problem solving in general. Pineda [10] treats diagrams as sentences constructed from diagrammatic primitives and focuses on the semantics of such sentences. In an informal sense, DRS is such a sentence, with the semantics of the symbols for diagrammatic objects being directly represented by its spatial specification. Heterogeneous representations from a logic perspective are considered in [11], and in database retrieval in [12], neither of which is concerned with general issues in problem state representation. In work on general cognitive architectures such as ACT-R [13] and Soar [1], there has been significant recent emphasis on systems that deal with perceptual input, but the degree of integration of perception that we are proposing has not been attempted in those research areas.

7 Discussion

Diagrammatic reasoning is not magic slayer of the computational dragon – it so happens that humans have perceptual systems that works in parallel with deliberation, so it makes sense for humans to make use of that computational capacity to reduce the burden on deliberation, which is slow and memory-limited. What interest is diagrammatic reasoning to AI, which is not necessarily restricted to the contingencies of the design of the human architecture?

We propose three reasons. The first is that robotic systems would necessarily have, like humans, a perceptual system with a number of basic perceptual capacities built in. Thus, for robots solving problems, drawing an external diagram, or imagining one in its equivalent of DRS, and applying perceptual routines might offer similar benefit. Such systems may also find it useful to have the equivalent of visual memories of their experiences, in contrast to the current approach where the robot extracts some information about the world in a sentential form and stores it in it memory. DRS has suggestive implications for the design of such memories. The second reason applies even for AI systems more limited in their intended capabilities than integrated robotic systems. It is true that for a specific simple problem such as that in Fig. 1, it would make no sense to build the general purpose machinery such as that described in

this paper. However, there are AI applications that have to interact with human users, with the human and the machine sharing the problem solving responsibilities in a flexible way. Planning and situation understanding problems of this sort with a large spatial component are generic enough that investment in general perceptual reasoning would make sense. A third reason is that much human knowledge in various domains has diagrammatic components. Exploiting such knowledge would benefit from an ability to use diagrams in reasoning.

References

1. Newell, A., *Unified Theories of Cognition*. 1990, Cambridge, MA: Harvard University Press.
2. Lindsay, R., Imagery and Inference, in *Diagrammatic Reasoning: Cognitive and Computational Perspectives*, J. Glasgow, N.H. Narayanan, and B. Chandrasekaran, Editors. 1995, AAAI Press/MIT Press: Menlo Park, CA. p. 111-136.
3. Marr, D. and H.K. Nishihara, Representation and Recognition of the Spatial Organization of Three Dimensional Shapes, in *Proceedings of the Royal Society*. 1978. p. 200, 269-294.
4. Ullman, S., Visual routines. *Cognition*, 1984. **18**: p. 97-159.
5. Lindsay, R.K., Using diagrams to understand geometry. *Computational Intelligence*, 1998. **14**(2): p. 238-272.
6. Chandrasekaran, B. Diagrammatic Representation and Reasoning: Some Distinctions. in *AAAI Fall 97 Symposium Series, Diagrammatic Reasoning*. 1997. Boston, MA: American Association for Artificial Intelligence.
7. Koedinger, K.R. and J.R. Anderson, Abstract Planning and Perceptual Chunks: Elements of expertise in geometry. *Cognitive Science*, 1990. **14**: p. 511-550.
8. Narayanan, N.H., M. Suwa, and H. Motoda, Hypothesizing behaviors from device diagrams, in *Diagrammatic Reasoning: Cognitive and Computational Perspectives*, N. Glasgow, N. H. Narayanan, and B. Chandrasekaran, Editors. 1995, The MIT Press: Cambridge, MA. p. 501-534.
9. Ferguson, R.W. and K.D. Forbus. GeoRep: A flexible tool for spatial representation of line drawings. In *Proc. 18th National Conference on Artificial Intelligence*. 2000. Austin, Texas: AAAI Press, pp. 510-516.
10. Pineda, L., Graflog: a Theory of Semantics for Graphics with Applications to Human-Computer Interaction and CAD Systems. 1989, Ph. D. Thesis, University of Edinburgh: Edinburgh, UK. p. 197+vii.
11. Barwise, J. and J. Etchemendy, Heterogeneous Logic, in *Logical Reasoning with Diagrams*, G. Allwein and J. Barwise, Editors. 1996, Oxford University Press: New York. p. 179-200.
12. Anderson, M. and B. Andersen, Heterogeneous Data Querying in a Diagrammatic Information System, in *Diagrammatic Representation and Inference*, M. Hegarty, B. Meyer, and N.H. Narayanan, Editors. 2002, Springer: New York. p. 109-111.
13. Anderson, J.R. and C.J. Lebiere, *The Atomic Components of Thought*. 1998: Lawrence Erlbaum Associates.