Multi-modal Cognitive Architectures: A Partial Solution to the Frame Problem

Unmesh Kurup (Kurup@cse.ohio-state.edu)
Computer Science and Engineering, Rm 395, 2015 Neil Ave
Columbus, OH 43210 USA

B. Chandrasekaran (chandra@cse.ohio-state.edu)
Computer Science and Engineering, Rm 591, 2015 Neil Ave
Columbus, OH 43210 USA

Abstract

Since its definition by McCarthy in 1969, the Frame Problem (FP) has been one of the more heavily debated problems in AI. Part of the debate has been on the exact definition of what the FP really is. The computational aspect of the FP can be thought of as reasoning about what changes and what doesn't change in a dynamic world. The "sleeping dog strategy" is considered to be a viable solution to this aspect of the FP. We intend to show that this strategy has a weakness that can be partially solved using diagrammatic reasoning, under certain conditions. A related and equally important problem, called the Ramification Problem, is to be able to reason about the indirect effects of an action in the world. Our proposal provides a more efficient solution to the Ramification Problem when reasoning about spatial relations. To illustrate our solution, we introduce a problem solving architecture based on Soar that is augmented with a diagrammatic reasoning component. A problem state in this augmented Soar is bi-modal in nature, one part being symbolic and the other diagrammatic. We describe its use in certain problems and show how the use of diagrams can handle the frame and ramification problems with respect to spatial relations.

Keywords: Multi-modal; Cognitive Architectures; Diagrammatic Reasoning; Frame Problem

Introduction

In their 1969 paper, McCarthy and Hayes discuss a number of philosophical issues in artificial intelligence. One of the issues they identified is the Frame Problem. As formulated by McCarthy and Hayes (now referred to as the logical or technical aspect of the FP), the FP is the problem of axiomatizing the *Common Sense Law of Inertia* - the understanding that an action is assumed to not have changed a property of a situation unless there is evidence to the contrary (Shanahan, 2004). A different problem involves how to reason about an action's consequences without having to go over the entirety of its (the agent's) knowledge. McDermott calls this the computational aspect of the FP (McDermott, 1987). Later discussions have identified other distinct, but related problems, including the Philosopher's aspect of the FP and the Ramification Problem, lurking where just one problem was first seen. There is broad consensus today that the frame problem in its logical guise has been solved (Shanahan, 1997) (Reiter, 2001). However, the computational aspect still remains problematic. For the remainder of the paper, when we refer to the FP, we mean its computational aspect.

In the logic framework, information about the world and the objects in it is represented as sentences in a symbolic language. When a problem solving system is built in this framework, the various actions that the agent can take in the world are represented as rules that have pre-conditions that decide when the rule applies and post-conditions that explicitly capture the changes. While various heuristics and strategies might mitigate the problem, there is a consensus that the FP and its variants are unavoidable in the sentential knowledge representation framework. This has resulted in suggestions that perhaps alternative representative frameworks might exist that avoid the problem. One such proposal by Janlert (1996) is that analog representations such as pictorial representations might be the answer. In such systems, a rule only encodes the basic action. The other changes are implicitly captured by the representation (as in the external world). Unfortunately, analog representations suffer from problems too, one of which is their tendency to over-specify. Due to this, the possibility exists that conclusions drawn within the framework could be wrong. To avoid this, agents using such representations usually perform additional reasoning that verifies any conclusions. As Pylyshyn (1996) has mentioned, when this additional reasoning is combined with the problem of converting problems into equivalent ones that have a spatial character, we end up trading one problem for another equally difficult problem.

Our proposal is two-fold. While reformulating every problem into a spatial one has prohibitive costs and is not always possible, we believe there is a small subset of problems, namely those that are already spatial or have a natural spatial analog that have no need for conversion. Analog representations can indeed be of help in these cases. Secondly, analog representations have another advantage. The sleeping dog strategy, the preferred technique for handling the problem in the logic framework, has the drawback that any addition to the vocabulary of operations in the world results in changes to the existing knowledge in the system. A diagrammatic representation does not have the same problem. We take an existing, well established general problem solving architecture – Soar (Laird *et al.*, 1987) – and augment it with a diagrammatic reasoning component. We then compare how traditional Soar and this

new Soar (called bi-Soar) perform in the blocks world domain when new relations are introduced in the world.

The Frame Problem

As mentioned earlier, the FP is faced by an agent reasoning in a dynamic world. Consider the blocks world domain. When a block A, resting on a block B, is moved to a block C, not only does the new world have A on top of C, it also no longer has A on top of B. Further, there could be other relations that have changed, like A may have been to the left of other blocks but after the move it could be to their right. To complicate things, there may also be a lot of relations that don't change as a result of the move, such as A's color, size, shape etc. The relations between B and C don't change either. Hayes (1987) provides a more comprehensive introduction to the frame problem. Without getting into too much detail about the history of the problem, it is easy to see that one possible way of handling it is to keep track of what changes are made by each action and that anything not explicitly mentioned as changing is assumed to not have changed. McDermott (1987) calls this the "sleeping dog" strategy. One way to implement such a strategy is using suitably parameterized add and delete lists to keep track of the consequences of actions. When the precondition of a rule was met, everything in the add list is added to memory and everything in the delete list is removed from memory. In order to control the lists from becoming too big, relations in the modeled world were divided into primitive and non-primitive relations. Non-primitive relations are those that can be inferred from primitive relations and hence, add and delete lists need contain only the changes to primitive relations.

Add and delete lists do indeed provide a solution to the FP, but have certain drawbacks. Consider the blocks world example described above. Each add and delete list describes which primitive relations are changed by an action. As more and more primitive relations are added to the world, the number of entries in the lists also increases. At some point, the lists will grow so large that the agent spends a significant amount of time simply updating the state of the world using these lists. Another problem is that the number of inferences required to derive a non-primitive relation from the primitive relations may turn out to be expensive and repeated application of such inferences could slow down the system. These drawbacks are well documented in the FP literature.

We believe that there is yet another drawback not identified by earlier discussions on the topic. Assume we want the representation to capture the new (primitive) relation right-of. This would naturally involve a new predicate right-of and a new action with add and delete lists that allows the agent to move blocks to the right of other blocks. But, that is not enough because moving a block to be on top of another block can also change its right-of relation to other blocks. Thus, to fully capture the effects of the new relation, the agent's add and delete lists for the action to move a block to be on top of another has to be modified too. In the worst case, adding a new relation to the agent would involve changing every add and delete list.

External Representations

Consider the same blocks world scenario as before. Except that the agent now has a piece of paper and a pencil and the ability to draw and erase shapes on the paper. Instead of representing blocks A, B and C using predicates, the agent instead draws them as blocks on the paper. If the agent needs to know the relationship between any of the blocks, it simply looks at the diagram and extracts the required information from it. If the agent has to move block A from B to C, it simply erases block A from its previous position above B in the diagram and redraws it on top of C. One can see how the FP does not exist in this representation and consequently there are no add and delete lists to update after a move. Now consider the example of adding a new relation *right-of* to the vocabulary of the world. This would involve adding perception and action routines that tell the agent how to check for the *right-of* relation and how to move something to the *right-of* another respectively. And that's all. There is no need to modify any other relation.

What makes this form of representation so powerful is a combination of factors – One, the nature of the problem allows the use of a spatial representation. There are many problems that can't be transformed into such a representation and cannot be solved using this technique. Two, the structure of the physical world ensures that the causality of space is applied to the diagrammatic representation. Three, the perceptual abilities of the agent are capable of carrying out the tasks that are required for perceiving, creating and modifying diagrams. This ability of diagrammatic representations (and spatial representations in general) to make explicit certain implicit consequences of an operation, has been referred to variously as free rides and emergent properties (Chandrasekaran *et al.*, 2004).

While the use of external analog representations is non-controversial, there has been much debate about the presence and availability of internal analog representations for reasoning. Without getting into the debate, our internalization of this representation can be justified simply as an AI solution to an AI problem. The idea of diagrammatic representations as a solution to the frame problem has been proposed before, most notably by Lindsay (1995). But, while Lindsay does lay out his vision of such a diagrammatic system, he merely mentions that "One may view perception as offering a solution to the frame problem by allowing "the world" to make appropriate inferences which are then "read" by the brain/mind." Our work takes a closer look at the frame problem space and identifies exactly where diagrammatic representations can make a contribution. Also, we propose diagrammatic representations as a solution to the drawback of the sleeping dog strategy and lastly, we show how a general purpose reasoner, namely Soar, can be augmented with diagrammatic representations to create a multi-modal

cognitive architecture that can be used to solve a variety of problems.

While the proposal is for an agent that is multimodal, we restrict our attention in the rest of the paper to bi-modal agents in which the predicate-symbolic component is augmented with a visual component. We next describe a representational system that is the functional equivalent of an external diagram.

The Diagrammatic Representation System (DRS) and the Bi-modal State

We make use of a data type called DRS, proposed in (Chandrasekaran *et al.*, 2004), to represent a diagram. A physical diagram (on a screen or on paper) is an image that contains diagrammatic objects, each to be interpreted as a *point*, *curve* or a *region*, that represent objects of interest in the domain of discourse. That is, the diagram is viewed not as an un-interpreted image, but as a configuration of diagrammatic objects. Note too that while in the physical diagram all the objects are regions, so that they can be perceived, DRS captures the intended diagram. If an object in the physical diagram appears as a circle, in DRS it would be treated as a Euclidean point object with just location to characterize it. DRS is domain-independent – the only objects are points, curves or regions. Interpreting them in domain terms is the job of the user of the representation. The objects in DRS have associated with them information about their spatiality -- locations for point objects, and representations that are functionally equivalent to the sets of points that constitute the objects for curves and regions. Associated with the DRS are a set of perception and diagram construction/modification capabilities; following Ullman (1984), these are called *routines*. All these routines are visual, but we use the more general term so that it will apply to the more general multi-modal view.

Perception Routines take diagrammatic elements as arguments and return information about specified spatial properties or spatial relations. There are two types of perception routines: the ones in the first type re-perform the figure-ground separation on the image – rather than on the DRS – perceiving emergent objects (e.g., the two sub-regions that emerge when a curve intersects a region.) Routines of the second type return specified spatial properties of objects, e.g., the length of a curve; and evaluate specified spatial relations between objects, e.g., Inside(Region1, Region2). These routines work from descriptions in DRS. DRS thus is an intermediate representation that supports reconstituting the image, a capability needed for emergent object identification, and also the perceptual routines that perceive properties of and relations between objects.

Routines that help in constructing or modifying the diagram are *action routines.* They create diagrammatic objects that satisfy specific perceptual criteria, such as "a curve object that intersects a given region object," and "a point object inside the region object." The sets of routines are open-ended, but routines that are useful across a number of domains are described in Chandrasekaran (2004), which also contain more information on DRS.

DRS is the functional equivalent of a diagram in the sense that it has the same information that a diagram has – objects and their spatiality – and can be operated on by routines that are equivalent to perception on external diagrams. DRS will provide us the representational medium for the visual modality of the bi-modal state. The symbolic component in the standard architectures may be transformed in two ways: by rules as in traditional symbolic representations and by the relational predicates generated by perceptual routines operating on DRS. The visual component of the state may be transformed by action routines invoked during problem solving to create or modify aspects of the diagram. Functionally, the bi-modal states are exemplified by Figures 1 and 2, both of which represent the same world state. The predicate-symbolic representations on the left of each of the figures are the usual state descriptions. The diagrams on the right are the visual modality. The diagrammatic part is represented in DRS. The fact that diagrams are the same in the figures, while the descriptive components are different leads us to an important point about how perceptual representations partially help with the Frame Problem.

With respect to the spatial aspects of the problem, the diagrammatic component is complete in a way that the symbolic component is not, and cannot be. We can apply different perceptions and extract different descriptive pieces from the diagrammatic component. Diagrammatic representations (and similar things such as scale models) can often provide a partial solution to the Frame Problem for the spatial components, provided perceptions are available for the information of interest. Thus, while contemplating the world state corresponding to Fig. 1, the agent can simply check the diagrammatic component – or the external world – perceptually to see if the required spatial relations are satisfied, e.g., Above(A, C), at the point when such information is needed. In fact, there is no real reason to carry the complete set of symbolic descriptions from state to state.

Bi-modal Augmentation of Cognitive State in Soar

Soar

Soar is an architecture for constructing general cognitive systems (Laird et al., 1987) that perform a wide variety of tasks. For achieving this goal, Soar provides representations for short and long-term memory, mechanisms for interacting with the external world, a subgoaling strategy that is independent of the problem and a learning mechanism that allows Soar to learn as a result of success in solving subgoals. The Soar architecture also provides a rule-based programming language that can be used to program the intelligent agent. Soar's long-term memory is in the form these rules or productions of the language. The agent's cognitive state is called Working Memory (WM) in Soar. For our immediate purposes, we do not need many of the

details about WM, which we will simply model as containing any goal state, description of the state of the world it is solving a problem about, and active operators. Soar's design belief is that all deliberate goal-oriented behavior can be cast as the selection and application of operators to the current problem state; and a goal is the desired outcome of the problem solving activity. All state representations in Soar make use of predicate-symbolic descriptions.

For comparison purposes, we constructed *biSoar*, that is a Soar with a diagrammatic component available to represent various states. The resulting problem state is a bi-modal problem state with part (or whole) of the information represented symbolically and part diagrammatically. The

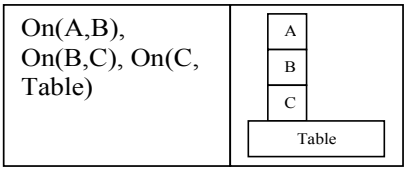

Fig 1. A bi-modal state

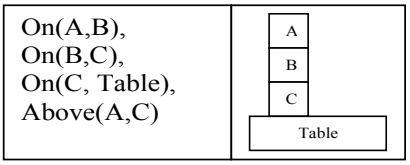

Fig.2. Alternate symbolic description of same world state as in Fig. 1.

diagram or diagrams used in the solver can be initialized as part of the initialization production of biSoar. A perceptual routine can be executed on a diagram by calling the routine in the RHS of a biSoar production. Due to the generic domain-independent nature of the DRS, the problem solver in biSoar needs to translate from the domain dependent nature of the perceptual questions to the generic ones supported by DRS.

For example, if biSoar asks a question "Is block A inside of box B1?", the question is translated into "Is region A inside of region B1?" During problem solving, the biSoar problem solver can modify the diagram by invoking the action routines, and modify the symbolic components by applying perceptual routines to the diagram.

In biSoar, when solving problems concerning some external situation, WM may contain several elements each of which may be augmented with a diagrammatic component. It is useful to distinguish between world state, goal state, and cognitive state. World state is simply the state of the world which is the subject of problem solving. Goal state is a state in the world that we wish to achieve. Cognitive state is the contents of the WM of the agent. WM may contain the goal state the agent is working towards, and the world state that is the result of any action being contemplated by the agent. In traditional implementation, each of these components of WM would be represented in a form similar to the left sides of Figures 1 and 2. In the augmented version, these parts of WM will each be

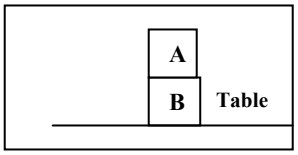

Fig 3: A simple blocks world example

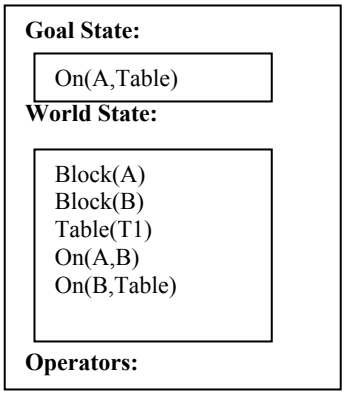

Fig 4: Initial contents of Soar's WM

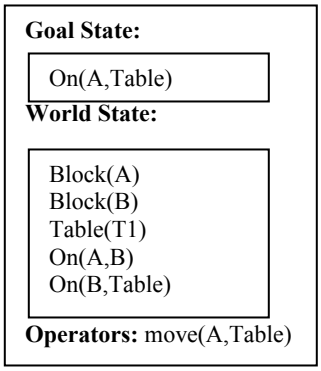

Fig 5: Soar's WM after operator proposal

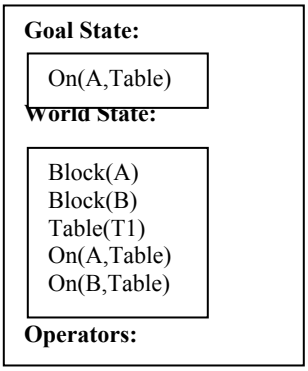

Fig 6: Soar's WM after Move applied

augmented with the corresponding diagrammatic component when such a representation is appropriate and available. In the rest of the paper, the term Soar will refer to the traditional symbolic version, while biSoar will refer to the augmented version.

Blocks World in Bi-modal Soar

Example 1

Let us start with an extremely simple example, Fig 3, to illustrate the basic ideas and issues. The situation has only two blocks – A and B and a Table, one relation *on-top-of* and a *move-on-top-of* operator. The goal is to create a domain state that satisfies the description ON(A,Table).

We will run through the representations in Soar, describing its problem space and working memory at each point in problem solving, and repeat the sequence for biSoar[1].

Fig 4 shows the starting state of working memory in Soar. It contains a description of the world state, and the current goal. During the proposal phase, the production for proposing a move operator fires and a move operator is proposed to move A onto the Table. Fig 5 shows the state of

[1] For our purposes, a content description of Soar's WM is all that is required. This is what the figures represent and should not be mistaken for an exact replica of Soar's WM.

Soar after the operator proposal phase. During the application phase, shown in Fig 6, the rule for applying the move operator fires and removes On(A,B) from the state and adds On(A,Table) to it.

Fig 7 shows the starting state for biSoar. There are two blocks – A and B and a table T. The problem state now has a diagram, represented in DRS, attached to it. In the Figure, the goal to be achieved is represented both symbolically and diagrammatically, but either alone might be sufficient[2]. Unlike in standard Soar, in biSoar there is no requirement for the symbolic part of the state to contain predicates describing the initial state world state, if the diagrammatic component depicts the situation. During the proposal phase, the rule that proposes the *Move* operator fires (this state is not shown in any of the figures). During the application phase, instead of updating the symbolic part, the rule calls the action routine to update the diagram to reflect Move(A,Table). Checking for preconditions can be done directly by the relevant perceptual routines. Fig 8 shows the final state after the move operator has been applied. Unlike standard Soar, biSoar does not need add or delete lists to keep track of the state of the world. The diagrammatic part does it instead.

Example 2

Let us add the following new relations to the world: *under*, *above*, *below*, *imm-right-of*, *imm-left-of*, *right-of*, *left-of* and *inside-of*. These relations are interpreted in their natural meanings, so we forego formal descriptions of them. The goal state to be achieved is described in terms of *above* and *right-of* relations.

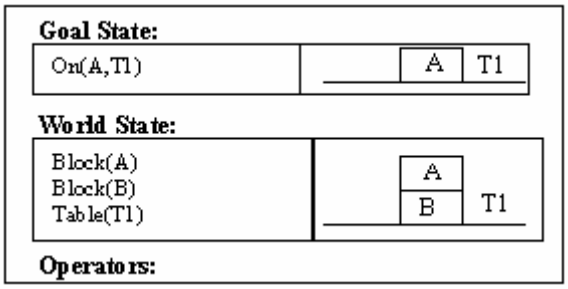

Figure 7: Initial contents of biSoar's WM

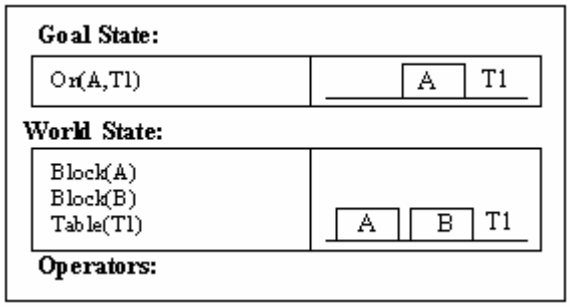

Figure 8: biSoar's WM after Move applied

[2]because of the ambiguity inherent in diagrammatic representations about what is intended, we only use symbolic goal descriptions.

For our list of relations, *on-top-of*, *under*, *imm-right-of*, *imm-left-of* and *inside-of* are the primitive relations while *above, below, right-of* and *left-of* are the non-primitive relations. In Soar, the primitive relations, from which all other relations may be derived, are updated after each change in the world. If a non-primitive relation is needed, the solver performs inference to find the answer.

Fig 9 shows the initial state for our blocks world problem. The final state is laid out as a sequence of goals to be achieved by the problem solver, while the initial state is simply a DRS representation of Fig 9. The goals to be achieved are: B *inside-of* B1, E *above* A, F *above* A, H *above* A, D *above* B, G *above* A. The problem solving sequences for the standard Soar problem solver and bi-modal Soar are shown in Fig 10. The solvers try to achieve each goal in the sequence in which it is presented. We examine one slice of this sequence. Consider the final sub-goal of the problem "G above A". To achieve this sub-goal, standard Soar first checks if Block A is clear. Since it's not, the solver sets up a sub-goal to find the topmost block above A. In order to find the topmost block, the solver performs inference by moving up the stack starting with block A. It finds that E is on-top-of A, that F is on-top-of E, H is on-top-of F and that there is nothing on-top-of H. Instead of just 3 blocks above A, if the stack had 20, the solver would have had to go through 20 such steps to find the topmost block. Consider the same sub-goal being solved in bi-modal Soar. The sequence does not vary from any of the other sub-goals. The solver calls a perceptual routine to check if A is clear. Since A is not, it calls the perceptual routine *topmost* to find the topmost block above A. The routine returns H and the solver calls the *move* routine to move G onto H. This sequence of problem solving steps is independent of the number of blocks in the stack. If there were 20, the solver would still call the topmost routine just once to find the topmost block.

Adding these new relations also means that we have to add the corresponding move operators for each of these relations as well as modify the existing *move-on-top-of* operator. For example, consider adding the *imm-right-of* relation. The corresponding *move-imm-right-of* operator will update the state of the world by adding and deleting the appropriate imm-right-of() predicates. It will also have to maybe add and delete some ON predicates depending on whether the block being moved was on or is being moved on to a block. But this is not enough. We also need to modify the existing *move-on-top-of* operator, because now, moving a block on top of another block could change its *imm-right-of* relations with other blocks. Similarly, now adding imm-left-of means that we need to modify both *move-on-top-of* and *move-imm-right-of* operators.

In biSoar, instead of modifying the symbolic content, we add perceptual and action routines corresponding to the *imm-right-of* and *imm-left-of* to the diagrammatic component. The *move-on-top-of* operator however was left untouched. According to Janlert (1996) "A sign that the frame problem is under proper control is that the

representation can be incrementally extended: A conservative addition to the furniture of the world would involve only a conservative addition to the representation." In our case, the world is the blocks world and an addition to

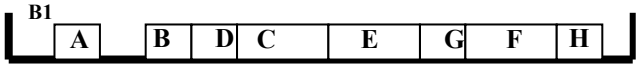

Figure 9. The initial state for Example 2.

the world can be in the form of objects and/or relations. The examples show that biSoar handles both additions well without exponential additions to the agent or modifications to its knowledge of existing objects and relations.

Using a diagrammatic component does come with its share of costs. Though each access to a routine is presented as taking only a single Soar cycle, the cycle itself could be considerably longer than normal. It is also worth mentioning

Standard Soar	Multi-modal Soar
Check if Box B1 clear	Check if box B1 clear
Move B to inside of B1	Move B to inside of B1
Check if A is clear	Find topmost block above A
Move E onto A	Move E onto A
Check if A is clear.	Find topmost block above B
No. Find block on-top-of A	Move F onto A
Check if E is clear	Find topmost block above A
Move F onto E	Move H onto F
Check if A is clear	Find topmost block above B
No. Find block on-top-of A	Move D onto F
Check if E is clear	Find topmost block above A
No. Find block on-top-of E	Move G onto H
Check if F is clear	
Move H onto left-half of F	
Check if B is clear	
No. Find block on-top-of B	
Check if E is clear	
No. Find block on-top-of E	
Check if F is clear	
Move D onto right-half of F	
Check if A is clear	
No. Find block on-top-of A	
Check if E is clear	
No. Find block on-top-of E	
Check if F is clear	
No. Find block on-top-of F	
Check if H is clear	
Move G onto H	

Fig 10: The problem solving sequences for standard and multi-modal Soar for the problem in Fig 9.

that while the complexity of perceptual routines is independent of the number of blocks and relations only up to a certain limit. There are also, important conceptual issues that remain. One relates to the aforementioned over-specificity of perceptual representations. While it seems to be a daunting task, we seem to be able to use diagrams without falling into the trap of over-specificity. It doesn't seem too hopeful to assume that agents could perform the same way and in comparable time. At the very least it is possible to specify what can be trusted for a particular domain.

Conclusion

Most of the approaches to solving the frame problem have been to find clever heuristics to restrict the explosion of causal effects in a dynamic world. A smaller section of these problems, namely those that depend on the causal structure of the world can be solved by the use of diagrams. There are serious drawbacks to this approach. However, it also has the advantage that changes to the vocabulary of the world can be incrementally added without a reworking of the existing parts of an agent. We have presented a multi-modal architecture that combines the predicate symbolic reasoning power of Soar with a diagrammatic component and reasons both symbolically and diagrammatically and shown that the agent is capable of dealing with a change in the number of/type of objects and relations without having to modify the existing knowledge of the agent.

Acknowledgements

Advanced Decision Architectures Collaborative Technology Alliance sponsored by the U.S. Army Research Laboratory under Cooperative Agreement DAAD19-01-2-0009.

References

Chandrasekaran, B., Kurup, U., Banerjee, B., Josephson, J.R., Winkler, R. (2004). *An architecture for problem solving with diagrams. Proceedings of the Diagrammatic Representation and Inference conference* (pp. 151-165): Springer-Verlag.

Hayes, P. J. (1987). What the frame problem is and isn't. In Z. W. Pylyshyn (Ed.), *The robot's dilemma: The frame problem in artificial intelligence* (pp. 123-138): Ablex Publishing.

Janlert, L.-E. (1996). The frame problem: Freedom or stability? With pictures we can have both.

Laird, J. E., Newell, A., & Rosenbloom, P. S. (1987). Soar: An architecture for general intelligence. *Artificial Intelligence, 33*(1), 1-64.

Lindsay, R. (1995). *Images and inference*. In Glasgow, J. et. al. (Eds), *Diagrammatic Reasoning: Cognitive and Computational Perspectives* (pp. 112-135): AAAI Press

McDermott, D. (1987). We've been framed: Or, why ai is innocent of the frame problem.

Pylyshyn, Z. W. (1996). The frame problem blues: Once more, with feeling.

Reiter, R. (2001). *Knowledge in action: Logical foundations for specifying and implementing dynamical systems*. Cambridge, MA.: The MIT Press.

Shanahan, M. (1997). *Solving the frame problem*: The MIT Press.

Shanahan, M. (2004). The frame problem. *The Stanford Encyclopedia of Philosophy*, 2004

Ullman, S. (1984). Visual routines. *Cognition, 18*, 97-159.