A Hybrid Connectionist and BDI Architecture for Modeling Embedded Rational Agents*

Deepak Kumar
Department of Math & Computer Science
Bryn Mawr College
Bryn Mawr, PA 19010
(610)526-7485
dkumar@cc.brynmawr.edu

Lisa Meeden
Computer Science Program
Swarthmore College
Swarthmore, PA 19081
(610) 328-8618
meeden@cs.swarthmore.edu

Abstract

In this paper, our ongoing work on a hybrid, connectionist and belief-desire-intention (BDI) based, rational agent architecture is described. The architecture makes specific commitments in order to achieve a harmony among the tasks of reasoning and acting. The architecture uses a bottom-up learning strategy to acquire rules for reactive behavior.

Embedded Agents: From Reactive to Planful Behaviors

It has been the goal of most research on planning to embody the computational models in a physical robot. From the early attempts based on SHAKEY the robot, it has been clear that the richness and unpredictability of the real world poses significant challenges to the design of robust physical agents (Nilsson 1984). The intractability of classical representation and reasoning models only adds to the problem. Thus, while the classical approach to reasoning and planning facilitates good goal-directed behavioral models, it falls short on two fundamental issues: the issue of reactivity and the resources required to carry out real-time deliberative behavior. In order to drive efficient goal-directed behavior, it often becomes necessary to sacrifice high-level models.

There has also been the introduction of a diametrically opposite approach to goal-directed behavior. This approach, called subsumption architectures, explicitly denies any high-level representation of knowledge as well as planning and acting modules (Brooks 1991). Complex behavior is achieved by a combination of very simple, low-level behaviors. While, not necessarily, goal-directed, these models exhibit promising reactive capabilities. It has since been argued that the synergy of both, conceptual as well as non-conceptual, representational paradigms may be needed in order to develop more robust, more powerful, and more versatile architectures (Sun & Alexandre 1997). These considerations have led us towards the development of a new architecture. We recognize the utility of both approaches: deliberative goal-directed behavior and reactive, learning behavior as facilitated by low-level behavior-based models. We are developing a hybrid, anytime architecture that tries to accommodate the benefits derived from both types of models in the context of physical mobile robots.

It has been recognized that the design of autonomous agents have the following properties: (a) there should be a relatively direct coupling between perception and action; (b) control should be distributed and decentralized; and (c) there should be a dynamic, closed-loop interaction between the environment and the agent (Maes 1991). While the desiderata of autonomous agent design has been laid out, there is little consensus about the nature of the computational models employed to satisfy these criteria (Kirsh 1991; Sun 1995; 1996). We are currently focusing on exploring a connectionist model coupled with a BDI architecture. Our research is explicitly concerned with implemented theories and systems. The knowledge representation and reasoning formalism used is called SNePS, Semantic Network Processing System (Shapiro & Group 1989; Shapiro & Rapaport 1992). The learning component is a recurrent artificial neural network developed through reinforcement learning. The physical agents are implemented on Khepera robots (Mondada, Franzi, & Ienne 1993). Each of these components will be described below.

BDI Architectures

One objective of the work described here is to contribute to the evolution of belief-desire-intention (BDI) architectures that can be used to model embedded rational agents capable of reasoning, acting, learning, and interacting based on unifying underlying principles. Specifically, the work is driven by the following:

*From Proceedings of 1998 AAAI Fall Sympoisum on Cognitive Robotics, AAAI Press

1. In the classical approaches to knowledge representation and planning, what are the relationships between beliefs, desires, and intentions?
2. What roles do these relationships play in the modeling of agents that are capable of reasoning, acting, as well as planning, in a unified framework?
3. Is it possible to apply a BDI approach to model agents that are physically embedded in the world?
4. What role can/does learning play for agents physically embedded in the world?

In this paper, we will describe some of the results we have obtained in exploring these questions.

Planning, Acting, and Inference: The Teleological Gap

We begin with the properties of the attitudes of belief, desires, and intention, the way they interrelate, and, most importantly, the ways they determine rational behavior in a unified fashion on a single, implemented platform. At the core of this approach are several knowledge representation issues, as well as issues relating to the architectures of planning and acting agents. We have observed that in most formalisms, it is somewhat awkward to do acting in reasoning systems, and it is awkward to study reasoning and representational issues in systems designed for acting/planning. The most successful planning/acting systems are relatively unsuccessful at being knowledge representation and reasoning (KRR) systems and vice versa. Even if a formalism were to use a common representation for beliefs, acts, plans, it would still require separate modules/processes for planning, acting, and inference.

Consequently, it is typically the case that a KRR system is a subordinate module to planning, and acting systems. It is the planning and acting modules that use the reasoning modules. Reasoning seems to be carried out in service of planning and acting. This results in a gap between the acting, planning, and reasoning processes. We call this the *representational-behavioral gap* or, the *teleological gap*. An agent ought to be able to act in service of inference. In our earlier work, we have illustrated this using the following blocksworld example:

```
All red colored blocks are wooden.
If you want to know the color of a block,
look at it.
```

In this example, *looking* is an action the agent can perform on an object and it results in the agent knowing the color of the object. The agent is able to exhibit the following behavior:

`Is A wooden?`
I wonder if A is wooden.
I wonder if A is colored red.

I wonder if A is a block.
I know A is a block.

Since A is a block I infer
If you want to know the color of A look at it.

I intend to do the act look at A.
I wonder if the act look at A has any preconditions.
...
Now doing: Look at A.
Sensory-add: *A is colored red.*

Since A is a block and A is colored red
and all red colored blocks are wooden
I infer A is wooden.

Notice that a backward chaining query lead the agent to perform an action in order to answer the query. Thus, acting was performed in service of inference. In what follows, we first provide an overview of the SNePS BDI architecture.

The SNePS BDI Architecture

The belief-desire-intention architecture we have developed is based on our analysis of the relationship between beliefs, plans, acts, and the process of reasoning and acting. This has led us to make several commitments.

Semantic Commitments

Let us look closely at the mechanism of inference. Reasoning is the process of forming new beliefs from other beliefs using inference rules. The connectives and quantifiers of the inference rules govern the derivation of new beliefs. Reasoning can be looked at as a sequence of *actions* performed in applying inference rules to derive beliefs from other beliefs. Thus, an inference rule can be viewed as a rule specifying an *act*—that of believing some previously non-believed proposition—but the "believe" action is already included in the semantics of the connective. Thus, another way of characterizing an inference engine is as a *mental actor* or a *mental acting executive*. During backward chaining, the mental acting executive forms the intention of believing the (queried) consequents of a rule if its antecedents are satisfied (i.e., preconditions are fulfilled). Similarly for forward chaining. McCarthy has also suggested that inference can be treated as a mental action (McCarthy 1986).

Alternatively, plans can be viewed as rules for acting. Reasoning rules pass a *truth* or a *belief* status from antecedent to consequent, whereas acting rules pass an *intention* status from earlier acts to later acts. In order to exploit this relationship between inference and acting we must make an architectural commitment.

Architectural Commitments

The above discussion suggests that we may be able to integrate our models of inference and acting by eliminating the acting component of the architecture. While it may sound appealing to redefine all the inference mechanisms as a bunch of explicit plans (under the new interpretation, this is theoretically possible), we have refrained from doing so. The trade-off here is that of the long-standing tradition of inference being a basic primitive in an AI system as well as the optimized implementation of inference (where previous deductions are not repeated, if valid), which is a necessity. The resulting unified acting and reasoning engine, which we are calling a *rational engine*, has to operate on beliefs as well as acts (Kumar 1993b). This poses a challenge to the underlying knowledge representation scheme, which leads us to the epistemological commitments described in the next section.

The SNePS Rational Engine, called SNeRE (Kumar 1993a; 1996), is an integrated reasoning and acting module that uses a logic called SWM (Martins & Shapiro 1988). It is the module responsible for the agent's reasoning processes. It is also the module responsible for the agent's acting and planning behavior. It employs an assumption-based truth maintenance (ATMS) system (Martins & Shapiro 1988). Thus, inferences, once drawn, are retained by the agent as long as their underlying support persists. The ATMS is also employed for implementing the extended STRIPS assumption (Georgeff 1987) for acting (Kumar & Shapiro 1993). Moreover, the rational engine is capable of modeling reactive as well as belief acquisition behavior (cases where inference can lead to acting).

Epistemological Commitments

The key to success lies not only in making the above semantic and architectural commitments but also an important Epistemological commitment: all knowledge required by the agent for reasoning, planning, and acting should be represented in a single formalism. In our previous work, we imposed an additional requirement that the modeled agent be capable of interaction using natural language.

The modeled agent's beliefs, plans, acts, and rules are represented in the SNePS semantic network formalism (Shapiro & Rapaport 1992). SNePS is an intentional, propositional semantic network system. Nodes in the semantic network represent conceptual entities—individuals, and structured individuals. Structured individuals can be propositions, which are used to represent beliefs, or acts and plans. Representing beliefs as well as acts as conceptual entities provides the central uniform framework for the architecture. Any conceptual entity represented in the system can be the object of a belief, plan, or act. By the same token, it can be reasoned about (or acted upon, as the case may be) and discussed by the agent representing it.

Acts can be primitive or complex (ones that will have to be decomposed into a plan) and are classified as *physical, mental*, or *control* acts. Physical acts are domain specific acts (like `PICKUP` or `PUT`). Mental acts are the acts of believing (or disbelieving) a proposition (i.e., they bring about changes in the agent's belief space). Control acts are used to structure plans (i.e., they control the agent's intentions). Our repertoire of control acts includes acts for sequencing (linear plans), conditional acts, iterative acts, nondeterministic choice and ordering acts, and qualifier acts—acts whose objects are only described and not yet fully identified (see (Kumar 1993a; 1996)).

Transformers In addition to standard beliefs that an agent is able to represent, we also define a special class of beliefs called *transformers*. A *transformer* is a propositional representation that subsumes various notions of inference and acting. Being propositions, transformers can be asserted in the agent's belief space; they are also beliefs. In general, a transformer is a pair of entities—$(\langle\alpha\rangle, \langle\beta\rangle)$, where both $\langle\alpha\rangle$ and $\langle\beta\rangle$ can specify beliefs or acts. Thus, when both parts of a transformer specify beliefs, it represents a reasoning rule. When one of its parts specifies beliefs and the other acts, it can represent either an act's preconditions, or its effects, or a reaction to some beliefs, and so on. What a transformer represents is made explicit by specifying its parts. When believed, transformers can be used during the acting/inference process, which is where they derive their name: they transform acts or beliefs into other beliefs or acts and vice versa. Transformations can be applied in forward and/or backward chaining fashion. Using a transformer in forward chaining is equivalent to the interpretation "after the agent believes (or intends to perform) $\langle\alpha\rangle$, it believes (or intends to perform) $\langle\beta\rangle$." The backward chaining interpretation of a transformer is, "if the agent wants to believe (or know if it believes) or perform $\langle\beta\rangle$, it must first believe (or see if it believes) or perform $\langle\alpha\rangle$." There are some transformers that can be used in forward as well as backward chaining, while others may be used only in one of those directions. This depends upon the specific proposition represented by the transformer and whether it has any meaning when used in the chaining process. Since both $\langle\alpha\rangle$ and $\langle\beta\rangle$ can be sets of beliefs or an act, we have four types of transformers—*belief-belief, belief-act, act-belief,* and *act-act*.

Belief-Belief Transformers: These are standard reasoning rules (where $\langle\alpha\rangle$ is a set of antecedent belief(s) and $\langle\beta\rangle$ is a set of consequent belief(s)). Such rules can be used in forward, backward, as well as bidirectional inference to derive new beliefs. For example, a class of transformers that represent antecedent-consequent rules is called `AntCq` transformers. In this paper, rather than drawing semantic networks, we will use the linear notation

$$\langle\alpha\rangle \rightarrow \langle\beta\rangle$$

to write them. For example "All blocks are supports" is represented as

$$\forall \mathtt{x}[\mathtt{Isa(x, BLOCK)} \rightarrow \mathtt{Isa(x, SUPPORT)}]$$

In addition to the connective above (which is also called an or-entailment), our current vocabulary of connectives includes and-entailment, numerical-entailment, and-or, thresh, and non-derivable. Other quantifiers include the existential, and the numerical quantifiers (see (Shapiro & Group 1989)).

Belief-Act Transformers: These are transformers where $\langle\alpha\rangle$ is a set of belief(s) and $\langle\beta\rangle$ is a set of acts. Used during backward chaining, these can be propositions specifying preconditions of actions, i.e. $\langle\alpha\rangle$ is a precondition of some act $\langle\beta\rangle$. For example, the sentence "Before picking up A it must be clear" may be represented as

$$\mathtt{PreconditionAct(Clear(A), PICKUP(A))}$$

Used during forward chaining, these transformers can be propositions specifying the agent's desires to react to certain situations, i.e. the agent, upon coming to believe $\langle\alpha\rangle$ will form an intention to perform $\langle\beta\rangle$. For example, a general desire like "Whenever something is broken, fix it" can be represented as

$$\forall \mathtt{x}[\mathtt{WhenDo(Broken(x), FIX(x))}]$$

Act-Belief Transformers: These are the propositions specifying effects of actions as well as those specifying plans for achieving goals. They will be denoted `ActEffect` and `PlanGoal` transformers respectively. The `ActEffect` transformer will be used in forward chaining to accomplish believing the effects of act $\langle\alpha\rangle$. For example, the sentence, "After picking up A it is no longer clear" is represented as

$$\mathtt{ActEffect(PICKUP(A), \neg Clear(A))}$$

It can also be used in backward chaining during the plan generation process (classical planning). The `PlanGoal` transformer is used during backward chaining to decompose the achieving of a goal $\langle\beta\rangle$ into a plan $\langle\alpha\rangle$. For example, "A plan to achieve that A is held is to pick it up" is represented as

$$\mathtt{PlanGoal(PICKUP(A), Held(A))}$$

Another backward chaining interpretation that can be derived from this transformer is, "if the agent wants to know if it believes $\langle\beta\rangle$, it must perform $\langle\alpha\rangle$," which is represented as a `DoIf` transformer. For example, "Look at A to find out its color" can be represented as

$$\mathtt{DoIf(LOOKAT(A), Color(A, ?color))}$$

Act-Act Transformers: These are propositions specifying plan decompositions for complex actions (called `PlanAct` transformers), where $\langle\beta\rangle$ is a complex act and $\langle\alpha\rangle$ is a plan that decomposes it into simpler acts. For example, in the sentence, "To pile A on B first put B on the table and then put A on B" (where piling involves creating a pile of two blocks on a table), piling is a complex act and the plan that decomposes it is expressed in the proposition

$$\mathtt{PlanAct(SEQUENCE(PUT(B, TABLE), PUT(A, B)), PILE(A, B))}$$

The Rational Engine As shown above, we are able to represent beliefs, acts, reasoning rules, and plans using the same knowledge representation formalism. The formalism makes appropriate semantic distinctions between various conceptual entities. A single operating module, the rational engine, carries out inference as well as acting. The abstract interface of the resulting system is that of a *tell-ask-do* architecture, where one can either *tell* the modeled agent a fact, *ask* a query about one, or request the agent to form an intention to *do* something. Typically, a *tell* leads to forward chaining inference and *ask* leads to backward chaining. The satisfying of intentions is carried out by an inference engine.

Modeling Embedded Rational Agents

While we were experimenting with the design of SNeRE and considering various commitments that lead to it, we were mainly using software agents operating on controlled software environments (blocksworld, geographical information systems). The agent, using natural language interactions, was told about the domain, and how to act in it. In our more recent work, we have been working with a Khepera robot. In one of the basic experiments, the modeled agent exhibits simple navigational behavior (move around without colliding into obstacles): that of a *Braitenberg Vehicle* (Braitenberg 1984; Hogg, Martin, & Resnick 1991). The behavior can be easily described by the following procedure:

```
DO forever
   IF left is blocked THEN turn right
   IF right is blocked THEN turn left
   ELSE go straight
```

Considered one of the simplest robotic behaviors, it reveals several issues as well as options when modeled in the BDI architecture described above. First, the agent needs to have beliefs that describe the world adequately enough for it to carry out the behavior: `Blocked(LEFT)`, `Blocked(RIGHT)`, and `Clear(FRONT)`. We could use three primitive actions: `TURNLEFT`, `TURNRIGHT`, and `GOSTRAIGHT`. One may even decide to have the preconditions: `Blocked(RIGHT)`, `Blocked(LEFT)`, and `Clear(FRONT)`, respectively, for the three actions. The effects of the three actions could simply be to eliminate their preconditions, in the spirit of traditional planning models.

The issue of sensing/perception comes next. In this simple example, we could do with a simple `PERCEIVE` action, the result of which would be the assertion of one of three beliefs described above, depending on the current situation of the robot. However, one has to decide if `PERCEIVE` is going to be an act we would like the agent to explicitly intend. We realize that, as far as sensing goes, our model should accommodate *synchronous* as well as *asynchronous* sensing facilities in an agent. By synchronous sensing, we mean an explicitly modeled act of sensing (there may be several). Here again there are at least two possibilities: the agent performs the explicit sensing act as a part of its acting schema (as in a *perceive-reason-act* cycle); alternatively, the agent explicitly intends to perform a perception act (as in the blocksworld example, above). For the latter case, we could model the behavior of a Braitenberg vehicle as follows:

```
ITERATE(true,
  SEQUENCE(PERCEIVE,
    IF((Blocked(LEFT), TURNRIGHT),
       (Blocked(RIGHT), TURNLEFT),
       (Clear(FRONT), GOSTRAIGHT))))
```

If we were to incorporate the preconditions of actions as described above, the specifications in the conditional control action would be redundant. One could, instead, use the plan:

```
ITERATE(true,
  SEQUENCE(PERCEIVE,
    DOONE(TURNRIGHT, TURNLEFT, GOSTRAIGHT)))
```

If, on the other hand, the action schema were such that sensing was performed as a part of the schema itself, we could write:

```
ITERATE(true,
  DOONE(TURNLEFT, TURNRIGHT, GOSTRAIGHT))
```

If one were to employ an asynchronous sensing module, the agent should expect a flurry of assertions to which it would be expected to react. In this case, the following rules would suffice:

```
WhenDo(Blocked(LEFT),TURNRIGHT)

WhenDo(Blocked(RIGHT),TURNLEFT)

WhenDo(Clear(FRONT),GOSTRAIGHT)
```

As you can see, decisions on the kind of sensing involved impacts the way the agent's plans or desires are encoded. In the case of the simple Braitenberg Vehicle, one could get the desired behavior by employing several representations. Some would employ control actions, and some would make use of transformers (most notably, `WhenDo`), and some would need both. It is our belief that an embedded BDI architecture would facilitate all of these models. Within our hybrid system, the behavior developed at the lower level by the learning component should constrain the representational choices at the higher level.

Learning to Act in the Real World

Conducting learning on physical robots is a time consuming process. Often, for reasons of practicality, the robots used are quite small. This allows the task environment to be set up on a desktop with the robot tethered to a computer for data collection and tethered to an electrical outlet for power. One popular platform for conducting learning experiments is the Khepera robot. Khepera is circular in shape and miniature (diameter 55 mm, height 30 mm, and weight 70g). It has two motors which control two wheels that can be powered from -10 (full reverse) to 10 (full forward). It's standard sensory apparatus consists of eight infra-red proximity sensors which also measure ambient light. Six of these sensors are arranged in an arc across the front of the robot while the other two sensors are in the back. The proximity sensors measure reflectance; when this measure is high an obstacle is close. The range of these sensors depends on the reflexivity of the obstacle, but is generally about 30–40 mm.

Given this platform, we can now reconsider the navigation task discussed above in more detail. Dealing with a physical robot rather than a simulated model requires you to make some very specific choices. For perceiving, what does it mean for the Khepera to be blocked? We could designate `Blocked(LEFT)` to mean *the front leftmost sensors are reading the maximum value* or *the front leftmost sensors are reading above some threshold*. For acting, what does it mean for the Khepera to turn? We could designate `TURNLEFT` to be *motors(-10,10)*, *motors(-5,10)*, or *motors(0,10)*. Once we make these decisions we have restricted the possible solutions to the task at hand.

Rather than making these choices apriori, we can allow a learning system to have direct access to the raw sensor readings and the motor controls. Then given general feedback about how to solve the task, the learning system will discover when it is appropriate to turn and by how much.

In initial experiments on learning this task, feedback was based on three factors: speed, straightness of motion, and obstacle avoidance. Ultimately we want the robot to explore as much of the accessible environment without getting stuck. By rewarding speed and straight motion, in addition to obstacle avoidance, this could be achieved (Floreano & Mondada 1996). Each of these factors was normalized to be between 0 and 1 (with high values indicating good behavior) and multiplied together to produce an overall fitness score on each time step. For speed, the value was based on the average speed of the two wheels. For straightness, the value was based on the difference between the speed of the two wheels. For obstacle avoidance, the value was based on the maximum proximity reading.

Using this bottom-up reinforcement technique, the resulting behavior is more varied than one might expect when considering the problem only from the top-down. In one solution, as the Khepera approached an obstacle, it reacted to it when one of its proximity sensors was maxed out and another was beginning to increase. Then it would maintain a sharp turn (*motors(-9,9)*) for several time steps, eventually switching to a much more gradual turn (*motors(8,9)*). Finally, it would return to straight motion (*motors(9,9)*), but would sometimes have to make another small turning adjustment depending on the width of the obstacle. The reactions of the robot were smooth and appropriate to the environment.

In a second set of experiments, the learning system was given only abstracted features of the environment (on input, whether the robot was blocked on its left or right; on output, whether to turn left, turn right, or go straight). The resulting behavior was not as successful, receiving less reward overall and exhibiting a tendency to get stuck.

These preliminary findings seem to indicate that allowing a learning system to explore the full range of possible behaviors is essential to finding robust solutions. Once a learning system has discovered useful distinctions about the world, it then becomes fruitful to pass on these findings to a higher-level reasoning system.

Connecting Reacting and Planning

Meeden has shown that by using a connectionist framework adapted through reinforcement learning, it is possible to build physical, as well as simulated autonomous agents that, after a series of training sessions, exhibit plan-like, goal-directed behavior (Meeden, McGraw, & Blank 1993; Meeden 1996). The control networks displayed a large repertoire of navigational strategies for accomplishing the given goals. In fact, the hidden layer representations differentiated each strategy, even those with only subtle differences. This was proved by doing cluster analysis of the hidden layer activations that corresponded to specific behaviors. The hidden layer activations of the recurrent network controller can potentially contain a compressed history of the past, including goals, perceptions, and actions taken. The strategies embodied within the hidden layer activations were termed *protoplans*. These protoplans can serve as building blocks for planning at the deliberative level (Meeden 1994).

Our exercise of explicating the issues of deciding the best BDI representations for a simple Braitenberg Vehicle can now be combined with an examination of the kinds of protoplans such a vehicle would learn, if it were to be purely driven by a connectionist network. It is our working hypothesis that the goal of automatic transfer of protoplans into higher-level symbolic representations would provide the needed direction for chosing amongst the various options.

We are currently working on the integration of this model with the higher-level symbolic BDI architecture. It is expected that the combination of a deliberative architecture, good for goal-directed reasoning and acting behavior modeling, with a reactive and learning behavior would be a viable attempt to overcome the shortcomings of both styles of architectures. There is also a deeper scientific motivation for this work—the learning exhibited by the connectionist architecture can form the basis for the emergence of cognitive concepts at the deliberative level. This is a promising new direction. So far, most attempts at hybrid architectures have concentrated on a top-down flow of information (Hexmoor, Kortenkamp, & Horswill 1997). That is, goals and behaviors at the symbolic level are transferred to lower-level, motor activities. Other models provide a bottom-up path for information, but focus on basic situation-action rules. In our work, we are proposing a bottom-up, learning-based model that will construct concepts of goal-directed, sequential behavior at the higher level purely based on the agent's physical interactions with its environment.

References

Braitenberg, V. 1984. *Vehicles: experiments in synthetic psychology.* Cambridge, MA: MIT Press.

Brooks, R. 1991. Intelligence Without Representation. *Artificial Intelligence* 47:139–159.

Floreano, D., and Mondada, F. 1996. Evolution of Homing Navigation in a Real Mobile Robot. *IEEE Transactions of Systems, Man, and Cybernetics-Part B: Cybernetics* 26(3):396–407.

Georgeff, M. P. 1987. Planning. In *Annual Reviews of Computer Science Volume 2.* Palo Alto, CA: Annual Reviews Inc. 359–400.

Hexmoor, H.; Kortenkamp, D.; and Horswill, I. 1997. Software Architectures for Hardware Agents. *JETAI* 9(1).

Hogg, D.; Martin, F.; and Resnick, M. 1991. Braitenberg Creatures.

Kirsh, D. 1991. Today the earwig, tomorrow man? *Artificial Intelligence* 47:161–184.

Kumar, D., and Shapiro, S. C. 1993. Deductive efficiency, belief revision and acting. *Journal of Experimental and Theoretical Artificial Intelligence (JETAI)* 5(2).

Kumar, D. 1993a. *From Beliefs and Goals to Intentions and Actions— An Amalgamated Model of Acting and Inference.* Ph.D. Dissertation, State University of New York at Buffalo.

Kumar, D. 1993b. Rational engines for BDI architectures. In Lansky, A., ed., *Proceedings of The 1993 AAAI Spring Symposium on Foundations of Automated Planning.* AAAI Press. 78–82.

Kumar, D. 1996. The SNePS BDI architecture. *Journal of Decision Support Systems* 16:3–19.

Maes, P. 1991. Guest editorial. In Maes, P., ed., *Designing Autonomous Agents.* The MIT Press.

Martins, J. P., and Shapiro, S. C. 1988. A model for belief revision. *Artificial Intelligence* 35(1):25–79.

McCarthy, J. 1986. Mental situation calculus. In Halpern, J. Y., ed., *Theoretical Aspects of Reasoning about Knowledge—Proceedings of the 1986 Conference*, 307.

Meeden, L. A.; McGraw, G.; and Blank, D. 1993. Emergence of control and planning in an autonomous vehicle. In *Proceedings of the Fifteenth Annual Meeting of the Cognitive Science Society*, 735–740. Hillsdale, NJ: Lawrence Erlbaum Associates.

Meeden, L. A. 1994. *Towards planning: Incremental investigations into adaptive robot control.* Ph.D. Dissertation, Indiana University.

Meeden, L. A. 1996. An incremental approach to developing intelligent neural network controllers for robots. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics* 26(3):474–485.

Mondada, F.; Franzi, E.; and Ienne, P. 1993. Mobile robot miniaturization: A tool for investigation of control algorithms. In *Proceedings of ISER93.*

Nilsson, N. J. 1984. Shakey the robot. Tech Note 323, AI Center, SRI International.

Shapiro, S. C., and Group, T. S. I. 1989. *SNePS-2 User's Manual.* Department of Computer Science, SUNY at Buffalo.

Shapiro, S. C., and Rapaport, W. J. 1992. SNePS considered as a fully intensional propositional semantic network. In Burkholder, L., ed., *Philosophy and the Computer.* Boulder, CO: Westview Press. 75–91.

Sun, R., and Alexandre, F., eds. 1997. *Connectionist-symbolic integration : from unified to hybrid approaches.* Mahwah, NJ: Lawrence Erlbaum Associates.

Sun, R. 1995. Robust Reasoning: Integrating Rule-Based and Similarity-Based reasoning. *Artificial Intelligence* 75:241–295.

Sun, R. 1996. Hybrid Connectionist-Symbolic Modules. *Artificial Intelligence Magazine* 17(2):99–102.