THE OK BDI ARCHITECTURE*

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

Stuart C. Shapiro
Department of Computer Science
State University of New York at Buffalo
Buffalo, NY 14260
(716) 645-3181
shapiro@cs.buffalo.edu

Abstract

The design of a belief-desire-intention (BDI) architecture is presented. The architecture is defined using a unified object-based knowledge representation formalism, called the OK formalism, and a unified reasoning and acting module, called the OK rational engine. Together they form the OK BDI architecture for modeling rational agents endowed with beliefs, desires, and intentions.

Keywords: BDI Architectures, Knowledge Representation and Reasoning, Acting, and Planning.

1 Introduction

A survey of AI systems would reveal that it is somewhat awkward to do acting in reasoning (or logic-based) systems (but it is convenient to talk about representational and reasoning issues using them), and it is awkward to study reasoning and representational issues in systems designed for acting/planning. Thus, most "good" planning/acting systems are "bad" knowledge representation and reasoning (KRR) systems and *vice versa*. For example, in a recent symposium on "Implemented KRR Systems" [24] out of a total of 22 KRR systems presented only four systems had capabilities for representation and reasoning about actions/plans (RHET [1], CYC [17],

*From IJAIT, International Journal of Artificial Intelligence Tools, Volume 3 Number 3 (1994) 349–366, World Scientific Publishing

CAKE [23] and SNePS [27]). The work presented in this paper presents an approach that bridges this "representational/behavioral gap."

In traditional planning/acting systems the inference engine is employed in service of planning and acting. In other words, it is the planning and acting processes that invoke the modeled thought (reasoning) processes. In the architecture presented in this paper, it is also possible for inference to use planning and acting. I.e., it is possible for thought to lead to action. We describe an integrated approach to reasoning and acting that is based on the following considerations:

- The modeled agent reasons, plans, as well as acts based on its beliefs, desires, and intentions.
- The modeled agent's beliefs, desires, actions, plans, etc., are represented in a single knowledge representation formalism.
- The modeled agent's reasoning and acting behavior is implemented using an amalgamated model of inference and acting.

The above considerations are satisfied by an implemented BDI (for Belief-Desire-Intention) architecture. In doing so, the architecture makes some semantic, epistemological, as well as operational commitments. The architecture makes a semantic clarification between inference and acting—that they are the same. Inference is a kind of acting: *mental acting*. This leads to an operational commitment—an architecture's reasoning and acting behavior should be carried out by a single module (as opposed to traditional approaches that employ two or more separate modules: an *inference engine*; a *planner*; and an *acting executive*). We call such a module a *rational engine* as it is solely responsible for the agent's reasoning, planning, and acting behavior. Not only does the modeled agent use inference in service of acting, it also employs acting in service of inference. In this paper, we describe the architecture that results when one makes the above set of commitments.

The work presented here has evolved from research involved in extending a semantic network-based KRR system, SNePS (whose rational engine called SNeRE is described in [11, 10, 12, 13]), into a BDI architecture. In this paper we use an object-oriented approach to describe the architecture. The resulting architecture is independent of, yet isomorphic to, the SNePS formalism. The resulting architecture enjoys all the advantages of object-oriented design—the ontology is easily extendible, as is the underlying logic, and amenable to a concurrent implementation [8, 12].

2 Motivation

Consider a modeled agent that has the set of beliefs acquired via an understanding of the following sentences about a blocksworld domain:

Blocks are supports. Red colored blocks are wooden. Picking up is a primitive action. Putting is a primitive action. Before picking up a block the block must be clear. After picking up a block the block is not clear and it is held. If a block is on a support then after picking up the block the support is clear and the block is not on the support. Before putting a block on a support the block must be held and the support must be clear. After putting a block on a support the block is not held, the block is clear, and the block is on the support. After putting a block on another block the latter block is not clear.

There is a table. The table is a support. A, B, and C are blocks. A is clear and it is on the table. B is clear and it is on the table. C is clear and it is on the table.

Now, consider posing the following queries (or requests) to the agent holding the above set of beliefs:

1. Is A a support?
2. Is A wooden?
3. What blocks are wooden?
4. Pick up A.
5. Put B on C.
6. Pick up a wooden block.
7. Put A on a wooden block.

Query 1 can be answered using standard backward chaining inference (italicized text is system response):

Is A a support?
I wonder if A is a support.
I wonder if A is a block.
I know A is a block.
Since A is a block and blocks are supports
I infer A is a support.

Queries 2 and 3 are similar to query 1 except in both cases, the inference will fail to produce an answer since the agent does not have any beliefs about blocks being wooden or red:

Is A wooden?
I wonder if A is wooden.
I wonder if A is a block.
I wonder if A is Red.
I know A is a block.
I don't know if A is wooden.

Of course, the agent could, as in the paragraphs above, acquire beliefs about the colors of blocks. However, our motivation here is to model behavior that would lead the agent to perform a belief acquisition act (like looking at an object) that would lead to its knowing the color of the block. This is a situation where an agent forms an explicit intention to act purely driven by its reasoning processes. In other words, the agent is said to be acting in service of inference. This is vastly different from intentions that an agent forms by an explicit request (as in 4–6 above). The latter being the norm for traditional planning/acting systems (i.e., inference, if used, is always in service of acting)[6]. For instance, for request 4, the agent will typically exhibit the following behavior:

Pickup A.
I intend to do the act Pickup A.
I wonder if the act Pickup A has any preconditions.
I wonder if A is a block.
I know A is a block.
Since A is a block
and before picking up a block it must be clear
I infer before picking up A it must be clear.
The act Pickup A has the following precondition:
A is clear.
I wonder if A is clear.
I know A is clear.
It is satisfied.
Now doing: Pickup A
. . . etc.

Clearly, inference is being used in service of acting. In the case of the failed query, the agent does have to know about the act of looking at objects. However, in the traditional planning and reasoning architectures, there is no way to relate backward chaining queries to actions that, if performed, may bring about the beliefs necessary in order to facilitate an answer. Requests 6 and 7 are explicit requests for performing acts where the agent would benefit by being able to act in service of inference which is in service of acting!. In the next section, we present the set of commitments required in order to design an architecture that provides a solution to the above problem.

3 Commitments

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.

3.1 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 propositional 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 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 [19]. 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.

3.2 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 [9]. This poses a challenge to the underlying knowledge representation scheme, which leads us to the epistemological commitments described in the next section.

3.3 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. We impose an additional requirement that the modeled agent be capable of interaction using natural language.

3.4 The SNePS BDI Architecture

Based on the above commitments, we have extended the SNePS KRR system to the SNePS BDI architecture. The SNePS BDI architecture has the following components:

SNePS BDI Architecture = SNePS Formalism + SNePS Rational Engine

The modeled agent's beliefs, plans, acts, and rules are represented in the SNePS semantic network formalism [28]. 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.

The SNePS Rational Engine, called SNeRE [11, 12], is an integrated reasoning and acting module that uses a logic called SWM [18]. 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 [18]. 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 for acting [6, 15]. Moreover, as will be evident in the section below, the rational engine is capable of modeling reactive as well as belief acquisition behavior (cases where inference can lead to acting). We have extracted the core ideas of the SNePS BDI architecture and incorporated them into the design of a SNePS-independent architecture, which is described in the next section.

4 The OK Architecture

We have defined the OK[1] architecture to have the following constituents:

OK BDI Architecture = OK Formalism + OK Rational Engine

The OK formalism is a conceptual, extendible, object-oriented hierarchy of classes that correspond to the various representational components of a knowledge representation system—individuals, propositions, and acts. While several object-based AI systems are already in existence, this is the first one

[1] OK stands for *O*bject-based *K*nowledge.

that makes an object-oriented commitment to the entities of a KR formalism itself. The OK rational engine is defined using methods inherited or specialized by the classes of the OK formalism.

4.1 The OK Formalism

The representational formalism is described as a conceptual object-oriented hierarchy. This is depicted in Figure 1. In an intensional representational framework, anything a cognitive agent can think about is termed a "mental concept" or a conceptual entity. More specifically these can be individuals, beliefs (propositions), or acts. 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 (as in requests 6 and 7 in Section 2) (see [11, 12]).

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

Figure 1: An Object-Oriented KR Formalism

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. We will use the 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 [26, 11]).

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

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

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

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

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

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

Our present model of acting is based upon a state-change model (see [14]). We identify three types of states—external world states, mental states (belief space), and intentional states (agent's current intentions). Accordingly, we identify three classes of actions—physical actions, mental actions, and control actions that bring about changes in their respective states. Thus `PICKUP` is a physical action, we have `BELIEVE` and `DISBELIEVE` as mental actions whose objects are beliefs, and control actions are described below. Acts can be *primitive* or *complex* (not shown in the figure). A primitive act has an effectory procedural component which is executed when the act is performed. Complex acts have to be decomposed into plans.

Plans, in our ontology, are also conceptual entities. However, like acts, we do not define a separate class for them as they are also acts—albeit control acts. Control acts, when performed, change the agent's intentions about carrying out acts. Our repertoire of control actions includes *sequencing* (for representing linear plans), *conditional, iterative*, *disjunctive* (equivalent to the OR-splits of the Procedural Net formalism [25, 29]), *conjunctive* (AND-splits), *selective*, and *achieve* acts (for goal-based plan invocation).

Sequencing Act: $\texttt{SEQUENCE}(\texttt{a}_1, \texttt{a}_2)$
The acts $\texttt{a}_1$ and $\texttt{a}_2$ are performed in sequence. For example:

$$\texttt{SEQUENCE(PICKUP(A),PUT(A,TABLE))}$$

is the act of first picking up A and then putting it on the table.

Disjunctive Act: $\mathtt{DoONE}(\mathtt{a_1}, \ldots, \mathtt{a_n})$
This act represents a nondeterministic choice. One of the acts $\mathtt{a_1}, \ldots, \mathtt{a_n}$ is performed. For example:

$$\mathtt{DoONE(PICKUP(A), PICKUP(B))}$$

is the act of picking up A or picking up B.

Conjunctive Act: $\mathtt{DoALL}(\mathtt{a_1}, \ldots, \mathtt{a_n})$
All of the acts $\mathtt{a_1}, \ldots, \mathtt{a_n}$ are performed in some order. For example:

$$\mathtt{DoALL(PICKUP(A), PICKUP(B))}$$

is the act of picking up A and picking up B.

Conditional Act: $\mathtt{IF}((\mathtt{p_1}, \mathtt{a_1}), \ldots, (\mathtt{p_n}, \mathtt{a_n}))$
Some act $\mathtt{a_i}$ whose $\mathtt{p_i}$ is believed is performed. For example:

$$\mathtt{IF((Clear(A), PICKUP(A)), (Clear(B), PICKUP(B)))}$$

is the act of either picking up A (if A is clear) or picking up B (if B is clear).

Iterative Act: $\mathtt{ITERATE}((\mathtt{p_1}, \mathtt{a_1}), \ldots, (\mathtt{p_n}, \mathtt{a_n}))$
Some act $\mathtt{a_i}$ whose corresponding $\mathtt{p_i}$ is believed is performed and the act is repeated. For example:

$$\mathtt{ITERATE((Clear(A), PICKUP(A)), (Clear(B), PICKUP(B)))}$$

is the act of picking up A (if A is clear) and picking up B (if B is clear).

Achieve Act: $\mathtt{ACHIEVE(p)}$
The act of achieving the proposition p. For example:

$$\mathtt{ACHIEVE(Clear(A))}$$

is the act of achieving that A is clear.

Single-object Qualifier Act: $\mathtt{WITHONE(x, y, ...)(p(x, y, ...), a(x, y, ...))}$
Find some x, y, etc. that satisfy p and perform act a on them For example:

$$\mathtt{WITHONE(x)(Held(x), PUT(x, TABLE))}$$

is the act of putting on the table something that is being held.

Multiple-object Qualifier Act: $\mathtt{WITHALL(x, y, ...)(p(x, y, ...), a(x, y, ...))}$
Find all x, y, etc that satisfy p and perform the act a on them. For Example,

$$\mathtt{WITHALL(x)(Held(x), PUT(x, TABLE))}$$

is the act of putting on the table everything that is being held.

These control acts are capable of representing most of the existing plan structures found in traditional planning systems (and more). We should emphasize, once again, that since plans are also conceptual entities (and represented in the same formalism) they can be represented, reasoned about, discussed, as well as followed by an agent modeled in this architecture.

4.2 The OK Rational Engine

Next, we will outline details of the integrated reasoning and acting module— called a *Rational Engine* (as opposed to an inference engine that only performs inference). A Rational Engine is the 'operational' component of the architecture (the interpreter) that is responsible for producing the modeled agent's reasoning and acting (and reacting) behavior. It is specified by three types of methods (or messages)—

Believe– A method that can be applied to beliefs for assertional or querying purposes. Consequently there are two versions—

- *Believe!(p)*– where p is a proposition, the method denotes the process of asserting the proposition, p, in the agent's belief space. It returns all the propositions that can be derived via forward chaining inference/acting.
- *Believe?(p)*– where p is a proposition, it denotes the process of querying the assertional status of p. It returns all the propositions that unify with p and are believed by the modeled agent either explicitly or via backward chaining inference/acting.

Intend– that takes an act as its argument (*Intend(a)*) and denotes the modeled agent's intention to perform the act, a.

Transform– These methods enable various transformations when applied to transformers. Corresponding to backward and forward chaining interpretations there are two versions— *Transform?* and *Transform!*, respectively.

Notice that the first two also correspond to the propositional attitudes of belief and intention. The methods *Believe* and *Intend* can be invoked by a user interacting with the agent. New beliefs about the external world can be added to the agent's belief space by using *Believe!* and queries regarding agent's beliefs are generated using *Believe?*. These methods, when used in conjunction with transformers lead to chaining via the semantics of the transformers defined above. The architecture also inherently provides capabilities for consistency maintenance. Each specific object that is a belief can have slots for its underlying support. The support is updated and maintained by the *Believe* methods as well as the mental actions `BELIEVE` and `DISBELIEVE` (together they form the TMS). The effectory procedures for `BELIEVE` and `DISBELIEVE` are implemented as belief revision procedures. We have found that such an integrated TMS facility simplifies several action and plan representations (see [15] for details). The *Intend* method is used to specify the fulfillment of agent's intentions by performing acts. All these methods can be specified (and specialized) for the hierarchy as well as inherited. Thus, domain specific acts (physical acts) will inherit the standard method for the agent to accomplish its intentions (i.e. the specific theory of intentionality employed), where as specializations of the *Intend* method can be defined for mental and control acts (to implement the semantics of respective acts).

Thus, an object-oriented design not only provides a uniform representational formalism, it also facilitates an extendible ontology. The semantics of representations is described by reasoning and acting methods that can be either individually specified or inherited and further specialized, as the case may be. Further, we would also like to claim that the representational formalism is 'canonical' in that its user interface (which is mainly defined via 'print methods') is also extendible. For example, the same object (say, a belief proposition) can be displayed as a frame, a predicate, a semantic network, or some other communicational entity (*ala* KIF) (see [11]). The next section presents how the example of Section 2 can be solved by such an architecture.

5 Example

In order for the failed inference of Section 2 to succeed, the agent only needs to have the following set of beliefs:

> Looking is a primitive action. If you want to know the color of something, look at it.

The primitive act of looking at something is modeled so that, when performed, it will result in the addition of a belief about the color of the entity being looked at. Thus, in a case where the color of the block A is indeed red, the agent, may 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 how, in the above example, 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.

6 Related Work

Our use of the term 'BDI Architectures' comes from Georgeff [6], that mentions the challenges of designing rational agents capable of goal-directed as well as reactive behavior based on the attitudes of beliefs, desires, and intentions. Georgeff specifically mentions that, 'the problem that then arises is specifying properties we expect of these attitudes, the way they interrelate, and the ways they determine rational behavior in a situated agent.' As explained in Section 1, we have taken the task of designing BDI architectures by defining a unified intensional representational formalism; identifying the semantic interrelationships between beliefs, desires, and intentions; capturing these into the idea of transformers; and finally designing a rational engine that brings about rational behavior based on these entities.

There has been work describing formal BDI models [3, 21]. There are also architectures that have been proposed that address various issues relating to rational agency. For instance [2, 20] describe a high-level BDI architecture that specifically focuses on issues relating to resource boundedness of rational agent behavior. Their work explores the hypothesis that plans, once committed, in addition to guiding the agent's actions, also constrain the agent's reasoning behavior. Rao and Georgeff [21, 22] have also studied formally the nature of intention and commitment in the context of rational agent behavior. The architecture reported in [22] provides a very simplistic representation of beliefs (thus suffering from some of the concerns mentioned in Section 1) together with a transition network-like formalism for plans. It is a (limited, though successful) attempt towards bridging the their earlier work on PRS [5, 7] and their later work on formal foundations of rational agents [21]. The work presented here complements these models.

It provides a general representational framework which these models lack. At the same time, it can facilitate easy incorporation of their ideas by virtue of the extendibility of the design.

We have taken a unified approach to representations. Drummond expresses the need for a single unified formalism for representing beliefs, acts, and plans [4]. This facilitates a single reasoning module to be able to reason about beliefs, acts, and plans. We have taken this approach a step further by explicitly identifying the semantic relationship between inference and acting so that a single module, a rational engine, in addition to reasoning, is also responsible for carrying out physical acts and plans (see [16] for examples). In our formalism, act representations are different from standard operator-based representations of classical planning/acting systems. Elsewhere [15], we have also shown how even simple act representations can benefit from an integrated TMS. In the presence of a TMS even the simplest acting model (that of adding and deleting the act's effects) implements the extended STRIPS assumption. As a result, ours is a deductive approach to acting. While this leads to tractability concerns, we feel that it provides consistency in the modeled agent's belief space and forms the basis for rational behavior. This also facilitates a deductive approach to hierarchical plan decomposition (specific `PlanAct` and `PlanGoal` propositions can be deduced in order to find plan decompositions). Search during reasoning/acting/plan decomposition is focused by means of some KR principles, the Uniqueness Principle being one (there is a one-to-one correspondence between instances and intensional entities) [10]. The Uniqueness Principle helps focus the chaining (method/message propagation) through a restricted set of entities.

The object-oriented approach provides a promising approach to building BDI architectures. It can be used to implement a unified representational formalism that bridges the gap between classical approaches to representation/acting/planning and the emerging paradigms for designing and implementing integrated intelligent architectures.

7 Summary

By making a semantic clarification, an operational commitment, and an ontological commitment that maintains proper distinctions between beliefs, plans, acts, reasoning rules, and reacting rules, we are able to arrive at an integrated BDI architecture that is capable of reasoning as well as acting. The design of the resulting BDI architecture was presented. The architecture was defined using a unified object-based knowledge representation formalism, called the OK formalism, and a unified reasoning and acting module, called the OK rational engine. Together they form the OK BDI architecture for modeling rational agents endowed with beliefs, desires, and intentions.

References

[1] James Allen. The RHET System. In Charles Rich (Guest Editor), editor, *SIGART BULLETIN Special Issue on Implemented KRR Systems*, pages 1–7, June 1991.

[2] Michael E. Bratman, David J. Israel, and Martha E. Pollack. Plans and Resource-Bounded Practical Reasoning. *Computational Intelligence*, 4(4), 1988.

[3] P. R. Cohen and H. J. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42(3), 1990.

[4] Mark E. Drummond. A representation of action and belief for automatic planning systems. In Michael P. Georgeff and Amy L. Lansky, editors, *Reasoning about Actions and Plans - Proceedings of the 1986 Workshop*, pages 189–212, Los Altos, CA, 1987. AAAI and CSLI, Morgan Kauffmann.

[5] M. P. Georgeff, A. Lansky, and P. Bessiere. A procedural logic. In *Proceedings of the 9th IJCAI*, 1985.

[6] Michael P. Georgeff. Planning. In *Annual Reviews of Computer Science Volume 2*, pages 359–400. Annual Reviews Inc., Palo Alto, CA, 1987.

[7] Michael. P. Georgeff and Amy. Lansky. Procedural knowledge. Technical Note 411, AI Center, SRI International, 1987.

[8] Deepak Kumar. An AI architecture based on message passing. In James Geller, editor, *Proceedings of The 1993 AAAI Spring Symposium on Innovative Applications of Massively Parallel Architectures*, pages 127–131. AAAI Press, March 1993.

[9] Deepak Kumar. Rational engines for BDI architectures. In Amy Lansky, editor, *Proceedings of The 1993 AAAI Spring Symposium on Foundations of Automated Planning*, pages 78–82. AAAI Press, March 1993.

[10] Deepak Kumar. A unified model of acting and inference. In *Proceedings of the Twenty-Sixth Hawaii International Conference on System Sciences*. IEEE Computer Society Press, Los Alamitos, CA, 1993.

[11] Deepak Kumar. *From Beliefs and Goals to Intentions and Actions—An Amalgamated Model of Acting and Inference*. PhD thesis, State University of New York at Buffalo, 1994.

[12] Deepak Kumar. The SNePS BDI architecture. *Journal of Decision Support Systems—Special Issue on Logic Modeling*, 1994. Forthcoming.

[13] Deepak Kumar, Susan Haller, and Syed S. Ali. Towards a Unified AI Formalism. In *Proceedings of the Twenty-Seventh Hawaii International Conference on System Sciences*. IEEE Computer Society Press, Los Alamitos, CA, 1994.

[14] Deepak Kumar and Stuart C. Shapiro. Architecture of an intelligent agent in SNePS. *SIGART Bulletin*, 2(4):89–92, August 1991.

[15] Deepak Kumar and Stuart C. Shapiro. Deductive efficiency, belief revision and acting. *Journal of Experimental and Theoretical Artificial Intelligence (JETAI)*, 5(2), 1993.

[16] Deepak Kumar and Stuart C. Shapiro. Acting in Service of Inference (and *vice versa*). In Douglas D. Dankel II, editor, *Proceedings of The Seventh Florida AI Research Symposium (FLAIRS 93)*. The Florida AI Research Society, May 1994.

[17] Douglas B. Lenat and Ramanathan V. Guha. The Evolution of CYCL, The CYC Representation Language. In Charles Rich (Guest Editor), editor, *SIGART BULLETIN Special Issue on Implemented KRR Systems*, pages 84–87, June 1991.

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

[19] John McCarthy. Mental situation calculus. In Joseph Y. Halpern, editor, *Theoretical Aspects of Reasoning about Knowledge—Proceedings of the 1986 Conference*, page 307, 1986.

[20] Martha E. Pollack. Overloading Intentions for Efficient Practical Reasoning. *Noûs*, XXV(4):513–536, September 1991.

[21] Anand S. Rao and Michael P. Georgeff. Modeling Rational Agents within a BDI-Architecture. In *Principles of Knowledge Representation and Reasoning— Proceedings of the Second International Conference (KR91)*, pages 473–485. AAAI, IJCAI, CSCSI, April 1991.

[22] Anand S. Rao and Michael P. Georgeff. An Abstract Architecture for Rational Agents. In Bernhard Nebel, Charles Rich, and William Swartout, editors, *Proceedings of the 2nd Conference on Principles of Knowledge Representation and Reasoning*, pages 439–449, San Mateo, CA, 1992. Morgan Kaufmann Publishers.

[23] Charles Rich. CAKE: An Implemented Hybrid KR and Limited Reasoning System. In Charles Rich (Guest Editor), editor, *SIGART BULLETIN Special Issue on Implemented KRR Systems*, pages 120–127, June 1991.

[24] Charles Rich. Special Issue on Implemented Knowledge Representation and Reasoning Systems—Letter from the Guest Editor. *SIGART Bulletin*, 2(3), June 1991.

[25] Earl D. Sacerdoti. *A Structure for Plans and Behavior.* Elsevier North Holland, New York, NY, 1977.

[26] S. C. Shapiro and The SNePS Implementation Group. *SNePS-2 User's Manual.* Department of Computer Science, SUNY at Buffalo, 1989.

[27] Stuart C. Shapiro. Case studies of SNePS. *SIGART Bulletin*, 2(3):128–134, June 1991.

[28] Stuart C. Shapiro and William J. Rapaport. SNePS considered as a fully intensional propositional semantic network. In Leslie Burkholder, editor, *Philosophy and the Computer*, pages 75–91. Westview Press, Boulder, CO, 1992.

[29] David E. Wilkins. *Practical Planning–Extending the Classical AI Planning Paradigm.* Morgan Kaufmann, Palo Alto, CA, 1988.