Acting in Service of Inference (and *vice versa*)*

Deepak Kumar
Department of Math. & Computer Science
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

We describe an integrated belief-desire-intention (BDI) architecture that is capable of acting in service of inference and *vice versa*. The results are achieved by identifying the relationship between inference and acting, and making certain architectural and ontological commitments. A single module called a rational engine operates on a uniform representation of beliefs, acts, plans, and reasoning rules. The architecture is implemented in the SNePS semantic network processing system.

1 INTRODUCTION

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, ontological, 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.

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

*From the Proceedings of the Seventh Florida Artificial Intelligence Research Symposium (FLAIRS-94), Douglas Dankel II (editor), May 1994.

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)[1]. 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 describe an architecture that provides a solution to the above problem.

3 A 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.

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 [8]. 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[3]. This poses a challenge to the underlying knowledge representation scheme, which leads us to the ontological commitments described in the next section.

The SNePS Rational Engine, called SNeRE[2, 4], is an integrated reasoning and acting module that uses a logic called SWM[7]. 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[7]. 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[1] for acting[6]. 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).

3.3 Ontological Commitments

The key to success lies not only in making the above semantic and architectural commitments but also an important ontological 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.

The modeled agent's beliefs, plans, acts, and rules are represented in the SNePS semantic network formalism[10]. 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 (as in requests 6 and 7 in Section 2) (see [2, 4]).

3.3.1 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 [9]).

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 `DoWhen` transformer. For example, "Look at A to find out its color" can be represented as

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

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

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

Thus, 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. In the next section, we present an example that demonstrates how the agent, using the architecture described, is able to act in service of inference.

4 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. We have also abstracted the main features of our architecture into a general object-oriented BDI architecture (called the OK BDI architecture) which we are in the process of developing[2]. The object-orientedness of the OK architecture is amenable to a concurrent implementation (our present architecture is also implemented using a quasi-concurrent computational paradigm). The architecture is also amenable to extensions in its ontology, its underlying logic, and its underlying action theory [5].

5 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 acting in service of inference (and *vice versa*).

References

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

[2] 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, 1993.

[3] 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.

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

[5] 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.

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

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

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

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

[10] 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.