Implementing BDI-like Systems by Direct Execution

Michael Fisher
Department of Computing
Manchester Metropolitan University
Manchester MI 5GD, United Kingdom

EMAIL: M.Fisher@doc.mmu.ac.uk

Abstract

While the Belief, Desire, Intention (BDI) framework is one of the most influential and appealing approaches to rational agent architectures, a gulf often exists between the high-level BDI model and its practical realisation. In contrast, the Concurrent METATEM language, being based upon executable formal specifications, presents a close link between the theory and implementation, yet lacks some of the features considered central to the representation of rational agents. In this paper, we introduce a hybrid approach combining the direct execution of Concurrent METATEM with elements of rationality from the BDI framework. We show how this system can capture a range of agent behaviours, while retaining many of the advantages of executable specifications.

1 Introduction

The inability of traditional software to handle complex modern applications, together with the rapid expansion of infrastructure such as the INTERNET, has led to the introduction of a new technology, termed *agent-based systems.* An *agent* is a semi-autonomous process, typically communicating via message-passing and cooperating with other agents in order to achieve common goals. This technology has been particularly successful in producing distributed information systems where centralised control is either impractical or undesirable. Here, not only is the ability of agents to act autonomously vital, but such agents are often required to dynamically adapt to unforeseen circumstances and to work cooperatively with other agents in order to overcome problems. These facets make agent-based systems ideal for complex tasks in real-world applications and, consequently, this technology has been applied in a wide variety of areas, from industrial process control to cooperative information retrieval.

In spite of the rapid spread of agent technology, there are, as yet, relatively few high-level programming languages for agent-based systems. Although considerable research has been carried out concerning the development of theories of agency and cooperation, the lack of appropriate high-level logic-based programming languages often means that implemented systems have very little connection with these high-level theories (an exception to this is Shoham's *Agent-Oriented Programming* work [Shoham, 1993]). Traditional programming languages typically lack the flexibility to handle high-level concepts such as an agent's dynamic control of its own behaviour. Consequently, it is widely regarded as important that high-level languages be provided, which can support the principled development of multi-agent systems, from logical theory to implemented system.

In this paper, we consider two approaches to the high-level representation of agent-based systems, namely Rao and Georgeff's BDI model [Rao and Georgeff, 1991] and Fisher's Concurrent METATEM language [Fisher, 1993]. In spite of their differing backgrounds we show that they have a great deal in common and introduce, in §4, a hybrid approach. Whilst not as general as the BDI model, nor as simple as the Concurrent METATEM language, this hybrid approach can capture a range of agent behaviours, while retaining many of the advantages of executable specifications.

2 The BDI Framework

Rao and Georgeff [1991] considered a particular agent framework whereby individual rational agents incorporate certain "mental attitudes" of *Belief, Desire* and *Intention* (BDI). These are used to represent, respectively, the information, motivational and deliberative states of the agent, and together effectively determine the system's behaviour. This framework is both appealing and influential, being used in a number of practical systems [Jennings and Wooldridge, 1995].

2.1 Agents

The core components of the BDI framework are rational agents [Wooldridge and Jennings, 1995]. These are autonomous entities, which execute independently, and have complete control over their own internal behaviour. The core elements of BDI agents are as follows. *Beliefs* correspond to the information that the agent has assimilated about the world; the set of beliefs is typically incomplete and may be incorrect with respect to the true situation. *Desires* are the agent's high-level goals, which need not be achievable simultaneously, but are usually consistent. *Intentions* essentially represent a subset of the agent's desires that it has committed to achieve.

2.2 Executing Agent Descriptions

The key process in executing BDI systems is *deliberation.* This consists of two aspects:

- deciding which desires will become intentions;
- deciding how to achieve those intentions.

Thus, an abstract execution cycle usually involves following steps [Rao and Georgeff, 1991].

1. Update beliefs based upon observations and actions.
2. Based on these beliefs, generate new desires.
3. Select a subset of desires to act as intentions.
4. Select a step to perform based upon intentions and the current state of the agent.

Thus, intentions are only completed one step at a time. The step selected corresponds either to performing an action, modifying a belief, or generating a subgoal and, once this step has been completed, the execution cycle begins again.

2.3 Real BDI Systems

Since its inception, many real-world agent-based systems have been based upon the BDI philosophy, most notably the Procedural Reasoning System (PRS) [Georgeff and Ingrand, 1989], but also systems such as INTERRAP [Fischer *et a*/., 19961. In the PRS, the selection of a step to be performed involves searching a plan library for plans which can achieve the selected intention. This library consists of pre-constructed plans; planning from first principles is not undertaken. In the PRS, *intention structures,* which are essentially partial orders of dependencies, provide linkage between all related intentions. For example, such structures record the fact that some intentions may be suspended, some may be deferred and some depend on further intentions.

Although both PRS and its successor, dMARS, have been successfully used in a number of areas, such as Air Traffic Control [Rao and Georgeff, 1995], the link between such systems and the BDI framework is often tenuous. In particular, while the practical systems incorporate elements termed beliefs and intentions, these are distinct from the formally defined beliefs and intentions of BDI model.

Consequently, there is a requirement for mechanisms linking the high-level BDI model with its low-level realisation. Several different approaches have been considered. For example, a number of BDI-based programming languages have been proposed [Weerasooriya *et al,* 1995; Rao, 1996a]. However, these have either been too low-level or not expressive enough to capture the key elements of BDI systems. More recently, object-oriented development methodologies have been adapted in order to provide, through a form of *agent-oriented* development methodology, a closer link between the model and its realisation [Kinny and Georgeff, 1997]. Since this practical development approach still does not consider the maintenance of formal properties, the need for high-level logic-based languages capturing the key components of the BDI model remains.

3 The Concurrent METATEM Framework

Fisher [1993] introduced Concurrent METATEM, an agent-based programming language comprising two elements:

1. The representation of each individual agent's behaviour using a temporal specification.
2. An operational framework providing both asynchronous concurrency and broadcast message-passing.

Temporal logic provides a means of declaratively specifying agent behaviour. It not only represents the dynamic aspects of an execution, but also contains a mechanism for representing and manipulating the goals of the agent. The operational model of Concurrent METATEM is both general purpose and intuitively appealing. The use of *broadcast* message-passing provides both a general and flexible communication model for concurrent objects [Birman, 1991] and a natural interpretation of distributed deduction [Fisher, 1997b]. These features together provide a coherent and consistent programming model within which a variety of agent applications can be represented [Fisher, 1994].

3.1 Agents

As in the BDI framework, the basic elements of Concurrent METATEM are agents, although these need not necessarily be rational. There are two elements to each agent: its *interface definition* and its *internal definition.*

The definition of which messages an agent recognises, together with a definition of the messages that an agent may itself produce, is provided by the interface definition, which may be given as follows.

```
car ()
  in:   go,stop,turn
 out:   empty,overheat
```

Here, {go, stop, turn} is the set of messages the 'car' agent recognises, while the agent itself is able to produce the messages {empty, overheat}.

The internal definition of each agent is given by a temporal logic specification [Manna and Pnueli, 1992]. Temporal logic is seen as classical logic extended with various modalities for representing temporal aspects of logical formulae [Emerson, 1990]. The temporal logic used here is based on a linear, discrete model of time. Thus, time is modelled as an infinite sequence of discrete 'moments', with an identified starting point, called 'the beginning of time'. Classical formulae are used to represent constraints within individual moments, while temporal formulae represent constraints *between* moments. Examples of temporal operators are given below.

$\Diamond\varphi$ is true now if φ is true at *some* moment in the future.

$\Box\varphi$ is true now if φ is true *always* in the future.

$\varphi\,\mathcal{U}\,\psi$ is true now if φ is true from now *until* a future moment when ψ is true.

$\bigcirc\varphi$ is true now if φ is true at the *next* moment in time.

start is only true at the *beginning* of time.

As an agent's behaviour is represented by a temporal formula, this can be transformed into the temporal normal form, SNF [Fisher, 1997a], This process not only removes the majority of the temporal operators, but also translates the formula into a set of *rules* suitable for cither execution or verification. Each of these rules is of one of the following varieties.

$$\begin{array}{rcl} \textbf{start} & \Rightarrow & \bigvee_{j=1}^{r} m_j \\ \bigwedge_{i=1}^{q} k_i & \Rightarrow & \bigcirc \bigvee_{j=1}^{r} m_j \\ \bigwedge_{i=1}^{q} k_i & \Rightarrow & \Diamond l \end{array}$$

where each k_i, m_j or l is a literal. Note that SNF is just as expressive as the full temporal logic. This logical form provides the core elements for describing basic dynamic execution:

- a description of the current moment;
- a description of transitions that *might* occur between the current and the *next* moment;
- a description of situations that will occur at some, unspecified, moment in the future.

Thus, using this approach, the behaviour of an agent can be represented now, in transition to the next moment in time and at some time in the future [Fisher, 1995].

3.2 Executing Agent Descriptions

An agent's temporal specification can be implemented in a number of ways, for example through refinement to traditional programming languages. However, since temporal logic represents a powerful, high-level notation, a viable alternative, at least for prototyping purposes if not for full implementation, is to animate the agent by directly executing its temporal specification [Fisher, 1996].

In the case of Concurrent METATEM, the set of SNF rules is executed using the *imperative future* paradigm [Barringer *et al,* 1996], Here, a *forward-chaining* process is employed, using information about both the history of the agent and its current set of rules in order to constrain its future execution.

As an example of a simple set of rules which might be part of the car agent's description, consider the following.

$$\begin{array}{rcl} \textbf{start} & \Rightarrow & \neg\texttt{moving} \\ \texttt{go} & \Rightarrow & \Diamond\texttt{moving} \\ (\texttt{moving} \land \texttt{go}) & \Rightarrow & \bigcirc(\texttt{overheat} \lor \texttt{empty}) \end{array}$$

Here, moving is false at the beginning of time and whenever go is true (for example, if a go message has just been received), a commitment to eventually make moving true is given. Similarly, whenever both go and moving are true, then either overheat or empty will be made true in the next moment in time.

The operator used to represent basic temporal indeterminacy is the *sometime* operator, '$\Diamond$'. When a formula such as '$\Diamond\varphi$' is executed, the system must attempt to ensure that φ *eventually* becomes true. As such eventualities might not be able to be satisfied immediately, a record of the outstanding eventualities must be kept, so that they can be re-tried as execution proceeds. The standard heuristic used is to attempt to satisfy as many eventualities as possible, starting with the oldest outstanding eventuality [Fisher and Owens, 1992].

An important consideration with respect to the practical implementation of Concurrent METATEM is that, although it *can* be made into a complete theorem-prover for temporal logic (at least in the propositional case), this is rarely done. Thus, the complexity of full theorem-proving is usually avoided [Fisher, 1996].

3.3 Concurrency, Communication and Grouping

We note that such asynchronously executing agents, communicating via broadcast message-passing, are very useful for developing open systems [Hewitt, 1991], while the notion of agent *groups* [Maruichi *et al.,* 1991] is essential both for restricting the extent of broadcast messages and for structuring the agent space.

3.4 Applications

The combination of executable temporal logic, asynchronous message-passing and broadcast communication provides a powerful and flexible basis for the development of agent-based systems. Consequently, Concurrent METATEM has been applied in a number of areas, including distributed artificial intelligence, concurrent theorem-proving, artificial agent societies and transport systems [Fisher, 1994].

4 A Hybrid Approach

In this section, we show how an extended version of Concurrent METATEM can be used to implement the core facets of the BDI model. As we are keen to represent only *the fundamental* elements, there are a variety of details that we ignore. For example, we will not consider real-time constraints, complex plan structures/libraries, or complex intention structures from the PRS, nor will we consider first-order temporal specifications, concurrent actions, multiple threads, cloning

BDI

```
Initialise-state
repeat
    options = option-generator(event-queue)
    selected-options = deliberate(options)
    update-intentions(selected-options)
    execute-step()
    get-new-external-events()
end repeat
```

METATEM

```
Initialise-state
repeat
    futures = generate-choices(state)
    choice = choose(futures,outstanding)
    outstanding = update(outstanding,choice)
    execute-step(choice)
    state = message-update(choice)
end repeat
```

Figure 1: A Comparison of Interpreter Cycles.

or grouping from Concurrent METATEM. In spite of these restrictions, we will show that the key elements of simple BDI architectures can be concisely and consistently represented and implemented using extended Concurrent METATEM.

4.1 Informational Aspects

As, in practical BDI architectures such as the PRS, beliefs are usually coded by ground instances of first-order predicate calculus, then these can be represented directly as (internal agent) facts in Concurrent METATEM. However, as Concurrent METATEM does not incorporate persistence, then this necessitates the addition of meta-level frame axioms, such as

$$[bel(\varphi) \wedge \neg change(\varphi)] \Rightarrow \bigcirc bel(\varphi).$$

Thus, an alternative approach is to directly add a new modal operator for belief to the logic executed within Concurrent METATEM. AS in BDI, this is a modal dimension satisfying the KD45 axioms and the execution mechanism itself is modified to handle the persistence of beliefs, i.e. both the above frame axiom and the KD45 axioms are 'built in' to the execution mechanism.

Thus, this is the key logical extension provided to Concurrent METATEM. The logic executed is now a multi-modal logic consisting of both temporal and doxastic dimensions. Note that SNF has been extended to such logics elsewhere [Fisher *et al*, 1996], and rules now incorporate the new belief modality, 'B'.

4.2 Motivational Aspects

The key observation we use regarding the BDI framework is that, in practice, there is often little fundamental difference between desires and intentions. The former represent all the goals that the agent wishes to achieve; the latter represent those that it is actively pursuing at present. Indeed, in the logical foundations of the BDI model [Rao, 1996b], both desires and intentions are represented by the same type of modal logic (KD), while beliefs are represented by a different one (KD45). Thus, a very natural (although Bratman [1987] takes a different view) mechanism for representing both intentions and desires is to use temporal eventualities. In particular, these are required to be satisfied eventually (if consistent), can be conflicting (e.g. $\Diamond\varphi$ and $\Diamond\neg\varphi$), and the system must choose between them in order to generate further execution.

4.3 Deliberative/Execution Aspects

Before considering the detailed execution within our approach and its relationship to BDI architectures, it is useful to examine the basic interpretation cycles for both the PRS [Rao and Georgeff, 1995] and core METATEM execution [Fisher and Owens, 1992]. These are presented in Fig. 1. Note the close similarities, the main difference being the more complex (and stratified) deliberation phase in the PRS.

Thus, the key modification of the execution mechanism concerns the implementation of deliberation. The subset of eventualities selected, the order in which they are attempted, and the mechanism for achieving them, are all the concern of deliberation. If, for the moment, we assume that desires and intentions are trivially achievable (this restriction is considered further below), then we can describe the key decision mechanisms of the two approaches as follows. As deliberation is concerned with weighing up each of the possible choices and choosing a subset to actually undertake, then in Concurrent METATEM, the eventualities (desires/intentions) are simply ordered by age and the oldest one is attempted first, while in BDI systems, each is examined in turn and a specific cost function is used to generate an ordering.

In order to provide added flexibility within the language, we allow the user to redefine the priority functions used within deliberation. Thus, the system requires a *desire priority* function, *pd,* which takes as arguments a list of eventualities (desires), and the history of the execution, and returns an ordered list of eventualities (intentions). This is then used to control the part of deliberation that extracts intentions from desires. In standard Concurrent METATEM, this would simply involve comparing the age of each eventuality; in the PRS this function would typically be much more complex, incorporating a variety of aspects of the current and past state of the agent.

If we were to remove the above simplification whereby intentions are trivially achieved, we could also allow a user defined *intention priority* function, p_i, which takes as arguments a list of eventualities (intentions), the history of the execution, and a list of plan definitions, and returns an ordered list of eventualities (intentions) to be executed immediately. In the PRS, the list returned would typically only contain one element (as only one step is taken at each cycle), though this framework can obviously accommodate more complex systems.

Thus, the simple deliberation mechanism in Concurrent METATEM (see Fig. l) is replaced by a two stage process whereby the function *pd* is used to generate a list of intentions and the function *pi* is used to choose appropriate actions, based upon this intention list. The implementation of these priority functions may be carried out in a number of ways, for example via meta-level METATEM rules [Barringer *et al* 1991] directly provided by the user.

In this way a language for implementing simple BDI-like systems can be provided by extending Concurrent METATEM with belief elements and user definable priority functions for use in deliberation. Note that the more complex elements of practical BDI systems may also be represented in this language. For example, in [Mulder et a/., 1996], an implementation of a simplified version of the PRS is provided, incorporating a range of low-level details, such as the manipulation and interpretation of plan components. Simply, dependencies between plans can be provided by asserting that if any sub-plan has not been constructed, the plan itself can not be completed, e.g.

$$\left(\bigvee_i \neg subplan_i\right) \Rightarrow \neg plan .$$

However, motivation for the work described in this paper is provided by the observation that such a representation of the PRS can be improved, simply by extending Concurrent METATEM with elements from the BDI model.

4.4 Analysis

There are five notable points to make regarding this hybrid approach of extending executable temporal logic with aspects of the BDI model.

Firstly, since the key elements added are the KD45 belief modality and the user defined priority functions, the Concurrent METATEM execution mechanism remains relatively unchanged. The efficiency of such an approach depends primarily on the complexity of the priority functions provided. For example, if a simple age-ordering function is used, then the speed is comparable with that of Concurrent METATEM.

A variety of deliberation mechanisms can be represented by the use of user definable priority functions within the execution mechanism. The constraints upon this are the complexity of evaluating such functions and the expressive power of the logical notation. Regarding the latter, the logical rule form is as expressive as full temporal logic and so a wide variety of functions can potentially be provided.

For this hybrid approach a formal semantics can be developed, though it would be parameterised by the semantics of the priority functions. If only sequences of actions undertaken are considered, then executions represent appropriate models for the BDI logics [Rao, 1996b].

This approach provides increased expressive power. Not only can a variety of priority functions be defined, but additional elements of Concurrent METATEM, such as multi-threading and disjunctive rules, can potentially be incorporated.

Finally, while there are no explicit axioms linking, say, beliefs and intentions (as are often found in theoretical works concerning BDI), rules in the program can be used to partially provide these, for example $\mathbf{B}\varphi \Rightarrow \Diamond\xi$ links beliefs and desires (eventualities).

5 Conclusions and Future Work

In this paper, we have described an extension to Fisher's Concurrent METATEM language that presents the possibility of implementing BDI-like systems using a form of executable multi-modal logic. While we do not claim that all BDI systems can be implemented using this approach, we believe a significant range of applications can be, providing the priority functions defined are not excessively complex.

The key observations from this work are that, in BDI systems, beliefs are primitive and distinct, while desires and intentions can essentially be represented as the same kind of entities. This, together with the observation that desires/intentions correspond, in many respects, to eventualities in temporal logic, allows the definition of basic BDI entities within Concurrent METATEM. Thus, this work separates the basic elements of the model (beliefs and desires) from the mechanisms for manipulating these and deciding how to act (deliberation).

While this approach will not be as efficient as directly implemented BDI systems, such as the PRS, it provides the opportunity at least for prototyping of BDI systems. The execution of temporal logic through Concurrent METATEM is well understood and, although the extended language requires the execution of a multi-modal logic, it remains a relatively inexpensive extension. As execution at an attempt to construct a model, rather than a complete theorem-proving process, it represents a relatively cheap mechanism for animating a specification. At the other extreme, it is difficult to see how the multi-modal BDI logic from [Rao, 1996b] could be directly executed.

Our future work concerns three areas: extended languages; refined priority functions; and expanded formal semantics. As mentioned above, there are elements of Concurrent METATEM that are not required in the BDI model. It is thus interesting to consider how these elements could be used in the framework described here. Also, once such a system is produced, it is relatively easy to extend it with the other elements of Concurrent METATEM, such as cloning and grouping. The definition of a range of detailed deliberation strategies, via appropriate priority functions, is another obvious direction. Finally, the temporal semantics of Concurrent METATEM must be adapted to fully incorporate the above extensions.

References

[Barringer et al.,., 1991] H. Barringer, M. Fisher, D. Gabbay, and A. Hunter. Meta-Reasoning in Executable Temporal Logic. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning (KR&R).* Morgan Kaufmann, 1991.

[Barringer *et al,* 1996] H. Barringer, M. Fisher, D. Gabbay, R. Owens, and M. Reynolds, editors. *The Imperative Future: Principles of Executable Temporal Logics.* Research Studies Press, Chichester, United Kingdom, 1996.

[Birman, 1991] K. P. Birman. The Process Group Approach to Reliable Distributed Computing. Techanical Report TR91-1216, Department of Computer Science, Cornell University, July 1991.

iBratman, 1987] M. E. Bratman. *Intentions, Plans, and Practical Reason.* Harvard University Press, 1987.

[Emerson, 1990] E. A. Emerson. Temporal and Modal Logic. In *Handbook of Theoretical Computer Science.* Elsevier, 1990.

[Fischer *et al.,* 1996] K. Fischer, J. Miiller and M. Pischel. A Pragmatic BDI Architecture. In *Intelligent Agents II(LNAI 1037).* Springer-Verlag, 1996.

[Fisher and Owens, 1992] M. Fisher and R. Owens. From the Past to the Future: Executing Temporal Logic Programs. In *LPAR (LNCS 624).* Springer-Verlag, 1992.

[Fisher, 1993] M.Fisher. Concurrent METATEM — A Language for Modeling Reactive Systems. In *Parallel Architectures and Languages, Europe (LNCS 694).* Springer-Verlag, June 1993.

[Fisher, 1994] M. Fisher. A Survey of Concurrent METATEM — The Language and its Applications. In *First International Conference on Temporal Logic (LNCS 827).* Springer-Verlag, July 1994.

[Fisher, 1995] M. Fisher. Representing and Executing Agent-Based Systems. In *Intelligent Agents.* Springer-Verlag, 1995.

[Fisher, 1996] M. Fisher. An Introduction to Executable Temporal Logics. *Knowledge Engineering Review,* 11(1), March 1996.

[Fisher, 1997al M. Fisher. A Normal Form for Temporal Logic and its Application in Theorem-Proving and Execution. *Journal of Logic and' Computation,*7 (4), July 1997.

[Fisher, 1997b] M. Fisher. An Open Approach to Concurrent Theorem-Proving. In *Parallel Processing for Artificial Intelligence III.* North-Holland, 1997.

[Fisher *et al.,* 1996] M. Fisher, M. Wooldridge and C. Dixon. A Resolution-Based Proof Method for Temporal Logics of Knowledge and Belief. In *Proceedings of the International Conference on Formal and Applied Practical Reasoning (FAPR).* Springer-Verlag, June 1996.

[Georgeff and Ingrand, 1989] M. P. Georgeff and F. F. Ingrand. Decision-Making in an Embedded Reasoning System. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI).* Morgan Kaufmann, 1989.

[Hewitt, 1991] C.Hewitt. Open information systems for distributed artificial intelligence. *Artificial Intelligence,* 47, 1991.

[Jennings and Wooldridge, 1995] N. R. Jennings and M. Wooldridge. Applying Agent Technology. *Applied Artificial Intelligence,* 9(4), 1995.

[Kinny and Georgeff, 1997] D. N. Kinny and M. P. Georgeff. Modelling and design of multi-agent systems. In *Intelligent Agents III (ATAL-96).* Springer-Verlag, 1997.

[Manna and Pnueli, 1992] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems: Specification.* Springer-Verlag, New York, 1992.

[Maruichi *et ai,* 1991] T. Maruichi, M. Ichikawa, and M. Tokoro. Modelling Autonomous Agents and their Groups. In *Proceedings of the 2nd European Workshop on Modelling Autonomous Agents and Multi-Agent Worlds (MAAMAW '90).* Elsevier/North Holland, 1991.

[Mulder *et al,* 1996] M. Mulder, M. Fisher and J. Teur. A Comparison of Concurrent METATEM and DESIRE. Internal Report. Department of Mathematics and Computer Science, Vrije Universiteit Amsterdam, 1996.

[Rao and Georgeff, 1991] A. S. Rao and M. P. Georgeff. Modeling Agents within a BDI-Architecture. In *International Conference on Principles of Knowledge Represent tation and Reasoning (KR),* Cambridge, Massachusetts, April 1991. Morgan Kaufmann.

[Rao and Georgeff, 1995] A. S. Rao and M. Georgeff. BDI Agents: from theory to practice. In *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95),* San Francisco, CA, June 1995.

[Rao, 1996a] A. Rao. Agentspeak(l): BDI Agents Speak out in a Logical Computable Language. In *MAAMAW-96 (LNAI1038).* Springer-Verlag, 1996.

[Rao, 1996b] A. S. Rao. Decision procedures for prepositional linear-time Belief-Desire-Intention logics. In *Intelligent Agents II (LNAI 1037).* Springer-Verlag, 1996.

[Shoham, 1993] Y. Shoham. Agent-oriented programming. *Artificial Intelligence,* 60(1), 1993.

[Weerasooriya et al., 1995] D. Weerasooriya, A. Rao, and K. Ramamohanarao. Design of a concurrent agent-oriented language. In *Intelligent Agents.* Springer-Verlag, 1995.

[Wooldridge and Jennings, 1995] M. Wooldridge and N. R. Jennings. Agent theories, architectures, and languages: A survey. In *Intelligent Agents.* Springer-Verlag, 1995.