ELECTRONIC WORKSHOPS IN COMPUTING

Series edited by Professor C.J. van Rijsbergen

**D Duke, University of York, UK and A S Evans, University of Bradford, UK
(Eds)**

BCS-FACS Northern Formal Methods Workshop

Proceedings of the BCS-FACS Northern Formal Methods Workshop,
Ilkley, UK. 23-24 September 1996

Towards Formal Methods for Agent-Based Systems

M. Fisher and M. Wooldridge

Springer

Published in collaboration with the
British Computer Society

Towards Formal Methods for Agent-Based Systems

Michael Fisher

Department of Computing, Manchester Metropolitan University
Manchester M1 5GD, United Kingdom

Michael Wooldridge

Mitsubishi Electric Digital Library Group
18th Floor, Centre Point, 103 New Oxford St.
London WC1A 1EB, United Kingdom

Abstract

This paper considers how formal methods might be developed for use in agent-based systems. In particular, it presents an approach based upon the description of agents using temporal specifications. Not only can these specifications be executed directly to provide the behaviour of an individual agent, but collections of such descriptions can be executed under a specific operational model in order to give the behaviour of multi-agent systems. In addition to animating their behaviour, verification of the agent descriptions with respect to required logical properties can be carried out; the paper outlines how temporal belief logics can be used to achieve this. Finally, the current limitations of this work are considered and its extension towards the ultimate goal of a full formal development framework for agent-based systems is discussed.

1 Introduction

The technology of intelligent agents and multi-agent systems provides the software developer with a powerful tool for approaching a complex and important class of software application problems for which mainstream software engineering techniques provide no solution. However, the successful application of agent technology will ultimately depend upon the development and use of principled methodologies for engineering agent-based systems. While many researchers and practitioners are *building* agent-based systems, almost none are directing research effort at *methodologies* for agent systems [20].

This paper describes one particular approach to agent-system specification and implementation. Our work lies within the mainstream software engineering tradition of formal methods: such methods have a long and influential history and we believe they have much to offer the agent-system builder. Specifically, our approach provides:

- a powerful abstract notation for specifying multi-agent systems, based on temporal multi-modal logic;
- a high-level agent-based programming language, closely related to our specification language, allowing early prototyping;
- a framework for verifying that our systems meet their specification.

(The approach is described in more detail in [11].) The high-level specification framework is based on the use of a *temporal belief logic*. The temporal component of the logic allows us to represent the dynamic nature of multi-agent systems, and in particular, to specify liveness and safety requirements. The belief component of the logic allows us to specify the *information state* of agents — the knowledge they have about the environment they occupy, which may contain other agents. This aspect of a system is essential if we are to reason about agents' decision making processes.

The multi-agent programming language is known as Concurrent METATEM [15]: in this language, agents are programmed by giving them a temporal logic specification of the behaviour it is intended they should exhibit. An

agent's behaviour is generated at run-time by directly executing its specification under an appropriate operational model [2]. Finally, the verification aspect of our framework is based on an axiomatization of Concurrent METATEM within the temporal belief logic. This axiomatization allows us to develop, for any arbitrary Concurrent METATEM system, a temporal belief logic theory representing the properties of that system. Verification proceeds by showing that the specification of the system follows from the theory of the system. To enable verification to take place, we have developed a range of proof methods for the temporal belief logic, based on both semantic tableaux and resolution proof methods [12, 21].

The remainder of the paper is structured as follows. In §2, we outline how the behaviour of agents is provided and introduce the temporal notation used, while, in §3, we describe Concurrent METATEM in more detail. In §4, we develop a Temporal Belief Logic (TBL), discuss various approaches to defining the semantics of this logic and use it to axiomatize the properties of Concurrent METATEM systems. Finally, comments and conclusions are presented in §5.

2 Describing Agents using Temporal Logic

We use *temporal logic* to specify the behaviour of an individual agent as it provides the core elements for representing dynamic behaviour, in particular:

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

2.1 A Simple Temporal Logic

We next give a definition of the propositional temporal logic used for describing agent behaviour (called PML); for a fuller account of the properties and applications of temporal logic see, for example, [5].

PML is essentially classical propositional logic augmented by a set of modal connectives for referring to the *temporal ordering* of actions. PML is based on a model of time that is *linear* (i.e., each moment in time has a unique successor), *bounded in the past* (i.e., there was a moment that was the 'beginning of time'), and *infinite in the future* (i.e., there are an infinite number of moments in the future).

2.1.1 Syntax

Definition 1 *The language of PML contains the following symbols:*

1. *a countable set, Prop, of proposition symbols;*
2. *the symbol **true**;*
3. *the unary propositional connective '¬' and binary propositional connective '∨';*
4. *the unary temporal connectives '●' and '○';*
5. *the binary temporal connectives 'U' and 'S';*
6. *the punctuation symbols ')' and '('.*

All the remaining propositional and temporal connectives are introduced as abbreviations (see §2.1.2). The set of well-formed formulae of PML is defined as follows.

Definition 2 *Well-formed formulae of PML, WFF_P , are defined by the following rules.*

1. *All proposition symbols are in WFF_P ;*

$\langle M, u \rangle \models \mathbf{true}$	
$\langle M, u \rangle \models p$	iff $\pi_p(u, p) = T$
$\langle M, u \rangle \models \neg\phi$	iff $\langle M, u \rangle \not\models \phi$
$\langle M, u \rangle \models \phi \vee \psi$	iff $\langle M, u \rangle \models \phi$ or $\langle M, u \rangle \models \psi$
$\langle M, u \rangle \models \bigcirc\phi$	iff $\langle M, u+1 \rangle \models \phi$
$\langle M, u \rangle \models \bullet\phi$	iff $u > 0$ and $\langle M, u-1 \rangle \models \phi$
$\langle M, u \rangle \models \phi \mathcal{U} \psi$	iff $\exists v \in \mathbb{N} \cdot (u \leq v)$ and $\langle M, v \rangle \models \psi$ and $\forall w \in \{u, \dots, v-1\} \cdot \langle M, w \rangle \models \phi$
$\langle M, u \rangle \models \phi \mathcal{S} \psi$	iff $\exists v \in \mathbb{N} \cdot (v < u)$ and $\langle M, v \rangle \models \psi$ and $\forall w \in \{v+1, \dots, u-1\} \cdot \langle M, w \rangle \models \phi$

Figure 1: Semantics of PML

2. If ϕ is in WFF_p then so are $\neg\phi$, $\bigcirc\phi$ and $\bullet\phi$;
3. If ϕ and ψ are in WFF_p then so are $\phi \vee \psi$, $\phi \mathcal{U} \psi$, and $\phi \mathcal{S} \psi$;
4. If ϕ is in WFF_p , then so is (ϕ) .

2.1.2 Semantics

The meaning of the temporal connectives is quite straightforward, with formulae being interpreted at a particular moment in time. If ϕ and ψ are well-formed formulae of PML, then $\bigcirc\phi$ is satisfied at the current moment in time if ϕ is satisfied at the next moment in time, while $\phi \mathcal{U} \psi$ is satisfied now if ψ is satisfied at some future moment, and ϕ is satisfied until then. The past-time connectives are similar: $\bullet\phi$ is satisfied at the current moment in time if there was a last moment in time and ϕ was satisfied then; $\phi \mathcal{S} \psi$ is satisfied at the current moment in time if ψ was satisfied at some previous moment in time, and ϕ has been satisfied since then.

More formally, we define a model, M , for PML as a structure $\langle \sigma, \pi_p \rangle$ where

- σ is the ordered set of states $s_0, s_1, s_2, \dots$ representing ‘moments’ in time, and
- $\pi_p : \mathbb{N} \times \text{Prop} \rightarrow \{T, F\}$ is a function assigning T or F to each atomic proposition at each moment in time.

As usual, we use the relation ‘ $\models$ ’ to give the truth value of a formula in a model M , at a particular moment in time u . This relation is defined for well-formed formulae of PML in Fig. 1. Note that these rules only define the semantics of the basic propositional and temporal connectives; the remainder are introduced as abbreviations (we omit the propositional connectives, as these are standard):

$\bullet\phi$	$\stackrel{\text{def}}{=} \neg \bigcirc \neg\phi$
start	$\stackrel{\text{def}}{=} \bullet\mathbf{false}$
$\diamond\phi$	$\stackrel{\text{def}}{=} \mathbf{true} \mathcal{U} \phi$
$\square\phi$	$\stackrel{\text{def}}{=} \neg \diamond \neg\phi$
$\phi \mathcal{W} \psi$	$\stackrel{\text{def}}{=} \square\phi \vee \phi \mathcal{U} \psi$
$\blacklozenge\phi$	$\stackrel{\text{def}}{=} \mathbf{true} \mathcal{S} \phi$
$\blacksquare\phi$	$\stackrel{\text{def}}{=} \neg \blacklozenge \neg\phi$
$\phi \mathcal{Z} \psi$	$\stackrel{\text{def}}{=} \blacksquare\phi \vee \phi \mathcal{S} \psi$

Note that the ‘**start**’ operator is particularly useful in that it can only ever be satisfied at the first moment in time. We will see in the next section that this plays an important role in the definition of a normal form for PML.

2.2 A Normal Form for Temporal Formulae

While we use arbitrary PML formulae to specify agent behaviour, it is often useful to translate such formulae into a normal form. The normal form we use is called Separated Normal Form (SNF) as it separates past-time from present and future-time formulae. It has been used to provide the basis for clausal proof methods [13] and execution methods [2], and can be defined as follows [14]. Formulae in SNF are

$$\Box \bigwedge_{i=1}^n (P_i \Rightarrow F_i)$$

where each ‘ $P_i \Rightarrow F_i$ ’ (called a *rule*) is constructed as follows.

$$P_i \in \{\mathbf{start}, \bigwedge_{j=1}^q m_j, \bigodot \bigwedge_{j=1}^q m_j\} \quad F_i \in \{\Diamond l, \bigvee_{k=1}^r l_k\}$$

where l_k , m_j , and l are literals.

In addition to being relatively easy to generate an SNF translation of an arbitrary PML formula, this normal form provides us with an intuitive representation for temporal formulae. To illustrate this, we below provide a few simple examples of properties that might be represented directly as SNF rules.

- Specifying initial conditions:

$$\mathbf{start} \Rightarrow sad$$

- Defining transitions between states:

$$\bigodot (sad \wedge \neg rich) \Rightarrow sad$$

- Introducing new eventualities (goals):

$$\bigodot (\neg resigned \wedge sad) \Rightarrow \Diamond famous$$

$$\bigodot sad \Rightarrow \Diamond happy$$

- Introducing permanent properties:

$$\bigodot lottery-win \Rightarrow \Box rich$$

which, in SNF, becomes

$$\bigodot lottery-win \Rightarrow rich$$

$$\bigodot lottery-win \Rightarrow x$$

$$\bigodot x \Rightarrow rich$$

$$\bigodot x \Rightarrow x$$

where x is a new proposition symbol.

3 Executing Agent Descriptions using Concurrent METATEM

In this section we introduce Concurrent METATEM as a mechanism for executing agent descriptions. We first describe the execution of a single temporal specification, and then consider the concurrent operational framework in which sets of agents execute. Finally, we outline a simple example, which illustrates how the language is used. Note that, although first-order temporal logic is used to define an agent’s behaviour in general [10], we will, for the most part, restrict our discussion to the propositional case (the exception being the example system in §3.3).

3.1 Animating Individual Agents

A Concurrent METATEM system contains a number of concurrently executing *agents*, each directly executing a temporal logic specification, provided by a set of SNF rules. Execution of such a specification, φ , is taken to mean constructing a model, $\mathcal{M}$, for φ , i.e. $\mathcal{M} \models_{PML} \varphi$.

Given a set of SNF rules, there are many ways to go about constructing an appropriate model and thus animating the specification [8]. The mechanism we use is based upon the METATEM paradigm of executable temporal logics [1]. The key idea that underpins this approach was introduced by Gabbay [17], involves executing each

antecedent about past $\Rightarrow$ consequent about future

rule by, on the basis of the past antecedent, imperatively making the future consequent true. This approach is called the *Imperative Future* paradigm [17].

Thus, the execution of an agent's specification is carried out by, at each moment in time, checking the past-time consequents of its rules and collecting together present and future-time constraints. A choice among these constraints is then made and execution continues to the next state. Thus, this approach utilises a form of forward chaining. This basic execution is further constrained by attempting to satisfy goals, such as $\Diamond g$. If only a subset of the goals can be satisfied at a given state, the oldest outstanding goals are attempted first. Any remaining unsatisfied are passed on as constraints to the next state.

In METATEM, execution is allowed to backtrack if a contradiction is found. However, in Concurrent METATEM backtracking is not allowed past the output of a message to the agent's environment (see below). Thus, the agent has effectively *committed* its execution after a message is output.

3.2 Execution in a Multi-Agent Environment

While we described the execution of a single specification in the previous section, Concurrent METATEM executes an evolving set of agent descriptions concurrently, allowing asynchronous communication between them. In Concurrent METATEM [15], each agent has two main components:

- an *interface*, which defines how the agent may interact with its environment (i.e., other agents);
- a *computational engine*, which defines how the agent may act.

In Concurrent METATEM the computational engine is simply provided by direct execution of the agent's temporal rules using the METATEM paradigm. An agent's interface consists of three components:

- a unique *agent identifier* (or just agent id), which names the agent;
- a set of symbols defining what messages will be accepted by the agent — these are called *environment propositions*;
- a set of symbols defining messages that the agent may send — these are called *component propositions*.

For example, the interface definition of a 'car' agent might be:

$$car(go, stop, turn)[fuel, overheat]$$

Here, *car* is the agent id that names the agent, $\{go, stop, turn\}$ are the environment propositions, and $\{fuel, overheat\}$ are the component propositions. Within the individual agent's execution, if a component predicate is satisfied, this has the side-effect of *broadcasting* the value of that proposition to all other agents. If a particular message is received, a corresponding *environment proposition* is made true in the agent's execution.

Although the use of only broadcast message-passing may seem restrictive, not only can standard point-to-point message-passing be easily simulated by adding an extra 'destination' argument to each message, but also the use of broadcast message-passing as the communication mechanism gives the language the ability to define more adaptable and flexible systems [7, 4, 3, 19].

Finally, not only are agents autonomous, having control over which (and how many) messages to send and receive, but they can form *groups* within the agent space. Groups are dynamic, open and first-class, and this natural structuring mechanism has a variety of diverse applications [7, 9, 16].

<i>funder(apply)[grant]:</i>	
$\bullet \text{ apply}(X)$	$\Rightarrow \Diamond \text{ grant}(X);$
$\text{grant}(X) \wedge \text{grant}(Y)$	$\Rightarrow X=Y.$
<hr/>	
<i>prof1(grant)[apply]:</i>	
start	$\Rightarrow \text{apply}(\text{prof1});$
$\bullet \text{ apply}(\text{prof1})$	$\Rightarrow \text{apply}(\text{prof1}).$
<hr/>	
<i>prof2(apply,grant)[apply]:</i>	
$\bullet \text{ apply}(\text{prof1})$	$\Rightarrow \text{apply}(\text{prof2}).$
<hr/>	
<i>prof3(grant)[apply]:</i>	
$\bullet \text{ grant}(\text{prof1})$	$\Rightarrow \text{apply}(\text{prof3}).$

Figure 2: Competing Academics Example

3.3 Example: Competing Academics

In this section, we provide a simple example of a multi-agent system, represented in PML, that can be executed using Concurrent METATEM. This example is a variation of the Snow White system so over-used in papers on Concurrent METATEM and considers competition between three agents (representing academics) to secure funding from a *funder* agent. In order to secure funding, the agents must send *apply* messages. These are processed by the *funder* and appropriate *grant* messages are then sent out.

The behaviour of the agents involved is presented in Fig. 2, and is described below. Note that we, for clarity, we have used the first order version of the language.

- *prof1* wants to apply at every cycle (perhaps a professor from a MMU?) in order to maximise the chances of achieving a grant. This is achieved because in the agent's first rule, **start** is satisfied at the beginning of time, and so an *apply* message is broadcast then, thus ensuring that $\bullet \text{ apply}(\text{prof1})$ is satisfied in the next moment, thus firing rule 2, and so on.
- *prof2* needs to be spurred into applying by seeing that *prof1* has applied (perhaps *prof2* is a professor from another 'new' university?). Thus, the agent's one rule is triggered when an appropriate *apply* is seen.
- *prof3* only thinks about applying when it sees that *prof1* has achieved some funding (perhaps *prof3* is a professor from an 'old' university?). Again its single rule is activated once a *grant(prof1)* message has been received.
- Finally, the *funder* will accept *apply* messages and promises to grant to each applicant at some point in the future (first rule). However, *funder* will only *grant* to at most one applicant at a time (second rule).

This relatively simple example demonstrates a number of features of Concurrent METATEM:

- safety and liveness rules (in *funder*);
- several agents sending the same message;
- different interfaces; and,
- broadcast enabling behaviour of one agent to be dependent on messages sent to/from others.

In the next section, we consider how properties of the system we described in §2 may be verified.

4 Verifying Properties of Temporal Descriptions

In order to reason about multiple agents, each containing a temporal specification, we introduce a temporal belief logic. Like PML, Temporal Belief Logic (TBL) is a linear discrete temporal logic, with bounded past and infinite future. In TBL, the language of PML is extended to include $[i]\varphi$, where φ is a temporal formula and i is an agent id, representing the ‘belief’ by agent i that φ is true. Thus, this extension allows us to describe the properties of formulae that are *local* to an agent. We now formally define the syntax of TBL, and examine two alternative semantics for this type of language.

Syntactically, TBL can be thought of as PML augmented by an indexed set of unary modal operators $[i]$, (applied to formulae of PML), and closed under the propositional and temporal connectives of PML. The ‘ i ’ in $[i]$ is an agent id. A TBL formula of the form $[i]\phi$ should be read “ ϕ is in i ’s current state.” Thus if ϕ is a history formula, this would say that ϕ was in i ’s history; if ϕ was a rule, this would say that ϕ was one of i ’s rules, and if ϕ were a commitment, it would say that i was committed to ϕ .

Definition 3 *The language of TBL contains all the symbols of PML, and in addition the square brackets ‘ $[$ ’ and ‘ $]$ ’ and a countable set, obj , of agent ids.*

Definition 4 *Well-formed formulae of TBL are defined by the following rules:*

1. TBL contains all formulae of PML;
2. If ϕ is a formula of PML and $i \in obj$ then $[i]\phi$ is a formula of TBL;
3. TBL is closed under the propositional and temporal connectives of PML.

There many possible ways of defining the semantics of TBL. Two that we have investigated are as follows:

- Wooldridge’s TBL [24, 22].

This logic is similar to L^B that Konolige developed for his deduction model of belief [18], and has been extensively studied as a formalism for multi-agent systems [24, 11].

It extends PML models with a function mapping each agent id and each moment in time to a set of PML formulae representing the agent’s state at that moment in time. Thus, a model, $\mathcal{M}$, for TBL is a structure $\langle \sigma, \pi_p, \mathcal{O} \rangle$ where

$$\mathcal{O} : obj \times \mathbb{N} \rightarrow \text{powerset}(\text{WFF}_p)$$

and σ and π_p are as in PML.

The formula $[i]\phi$ will be satisfied if ϕ is in i ’s state at the appropriate time. This gives the following additional semantic rule, for formulae of the form $[i]\phi$.

$$\langle \mathcal{M}, u \rangle \models [i]\phi \quad \text{iff} \quad \phi \in \mathcal{O}(i, u)$$

Other formulae are interpreted using the same rules as before. We will call this *grounded* TBL.

- The second approach is to treat the $[i]$ operator as a normal modal connective with Kripke semantics given by a serial, transitive, euclidean accessibility relation, giving a belief logic KD45 [6].

Models for this approach are slightly more complex, consisting of two relations, one for the temporal axis and one for the modal one. Points now exist in two dimensional space, rather than just in a sequence.

Thus, a model, $\mathcal{M}$, for TBL is a structure $\langle S, \pi_p, N, R_i \rangle$ where

$$R_i : (S \times S \times Ag) \rightarrow (S \times S)$$

defines the modal relation for each agent id, while

$$N : (S \times S) \rightarrow (S \times S)$$

defined the temporal (next-time) relation, and π_p is as in PML but maps pairs of states to valuations.

The semantics of a formula is thus defined with respect to a model and a pair of states. The main difference with PML is the semantics of the $[i]$ operator:

$$\langle M, (l, u) \rangle \models [i]\phi \Leftrightarrow \text{if } ((l, u), (m, v)) \in R_i, \text{ then } \langle M, (m, v) \rangle \models \phi$$

We call this *modal* TBL.

Tableaux proof methods for grounded TBL are described in [23]; resolution proof methods for modal TBL are described in [12, 21]

4.1 Axiomatizing Concurrent METATEM

In this section, we show how TBL can be used to *axiomatize* the properties of Concurrent METATEM systems. This involves extending the basic TBL proof system to account for the particular properties of the Concurrent METATEM systems. Examples of these are given below.

SENDING MESSAGES:	$\vdash [i]P \Rightarrow \Diamond P$	(P is one of i 's component propositions)
RECEIVING MESSAGES:	$\vdash P \Rightarrow \bigcirc \Diamond [i]P$	(P is one of i 's environment propositions)
HISTORICAL ACCURACY:	$\vdash [i]A \Rightarrow \bigcirc [i]\bullet A$	
SYNCHRONOUS EXECUTION:	$\vdash \bullet [i]A \Leftrightarrow [i]\bullet A$	
RULE AVAILABILITY:	$\vdash [i]\mathbf{start} \Rightarrow \Box [i]R$	(R is one of i 's rules)
SYNCHRONISED START:	$\vdash \mathbf{start} \Rightarrow [i]\mathbf{start}$	

4.2 Properties of Academics

Using the axiomatization of Concurrent METATEM given above, we can start to prove properties of, for example, the Concurrent METATEM system presented in §3.3. Some examples of the types of properties we can prove include:

1. *prof1* believes that it will apply infinitely often;
2. the *funder* is fair — if *prof1* applies infinitely often, it will be successful infinitely often;
3. *prof1* believes that it will be successful infinitely often (from (1) and (2));
4. *prof3* believes that it will apply at some moment in the future (from (3)).

For more detailed proofs relating to this form of system, see [11].

5 Conclusions and Future Work

In this paper we have outlined an approach to the use of formal method for the specification, verification, and rapid prototyping of multi-agent systems. While the approach shows promise, there are many pressing issues that must be addressed as the techniques move to the production of realistically sized systems. In particular:

- We have no complete semantics for Concurrent METATEM: issues such as backtracking and true concurrency make it hard to simply define a semantic model for the language;
- We have proof methods for the temporal belief logics described above, but we have no way of directly executing these logics — this means that agents in Concurrent METATEM are currently somewhat restricted in their ability to reason about each other;

- The proof methods we have for temporal belief logics such as TBL are currently restricted to the propositional case — proof methods for (restricted) first-order temporal belief logics will be required if we hope to verify properties of realistic systems.
- The theoretical complexity of proof methods for temporal belief logics — even in the propositional case — is likely to limit the applicability of such methods in general.

Acknowledgements

Michael Fisher was partially supported by the SERC under research grant GR/K57282. The views expressed in this paper are those of the authors, and are not necessarily those of Mitsubishi Electric.

References

- [1] H. Barringer, M. Fisher, D. Gabbay, G. Gough, and R. Owens. METATEM: An Introduction. *Formal Aspects of Computing*, 7(5):533–549, 1995.
- [2] 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.
- [3] K. P. Birman. The Process Group Approach to Reliable Distributed Computing. Technical Report TR91-1216, Department of Computer Science, Cornell University, July 1991.
- [4] A. Borg, J. Baumbach, and S. Glazer. A Message System Supporting Fault Tolerance. In *Proceedings of the Ninth ACM Symposium on Operating System Principles*, pages 90–99, New Hampshire, October 1983. ACM. (In ACM Operating Systems Review, vol. 17, no. 5).
- [5] E. A. Emerson. Temporal and Modal Logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, pages 996–1072. Elsevier, 1990.
- [6] R. Fagin, J. Halpern, Y. Moses, and M. Vardi. *Reasoning About Knowledge*. MIT Press, 1996.
- [7] M. Fisher. A Survey of Concurrent METATEM — The Language and its Applications. In *First International Conference on Temporal Logic (ICTL)*, Bonn, Germany, July 1994. (Published in *Lecture Notes in Computer Science*, volume 827, Springer-Verlag).
- [8] M. Fisher. An Introduction to Executable Temporal Logics. *Knowledge Engineering Review*, 11(1):43–56, March 1996.
- [9] M. Fisher and J. Keane. Realising a Concurrent Object-Based Programming Model on Parallel Virtual Shared Memory Architectures. In Giloi, Jahnichen, and Schriver, editors, *Programming Models for Massively Parallel Computers*. IEEE Computer Society Press, 1995.
- [10] M. Fisher and R. Owens. From the Past to the Future: Executing Temporal Logic Programs. In *Proceedings of Logic Programming and Automated Reasoning (LPAR)*, St. Petersburg, Russia, July 1992. (Published in *Lecture Notes in Computer Science*, volume 624, Springer-Verlag).
- [11] M. Fisher and M. Wooldridge. On the Formal Specification and Verification of Multi-Agent Systems. *International Journal of Cooperative Information Systems*, 6(1), January 1997.
- [12] 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)*, Bonn, Germany, June 1996.
- [13] M. Fisher. A Resolution Method for Temporal Logic. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI)*, Sydney, Australia, August 1991. Morgan Kaufman.

- [14] M. Fisher. A Normal Form for First-Order Temporal Formulae. In *Proceedings of Eleventh International Conference on Automated Deduction (CADE)*, Saratoga Springs, New York, June 1992. (Published in *Lecture Notes in Computer Science*, volume 607, Springer-Verlag).
- [15] M. Fisher. Concurrent METATEM — A Language for Modeling Reactive Systems. In *Parallel Architectures and Languages, Europe (PARLE)*, Munich, Germany, June 1993. (Published in *Lecture Notes in Computer Science*, volume 694, Springer-Verlag).
- [16] M. Fisher. An Open Approach to Concurrent Theorem-Proving. In *Parallel Processing for Artificial Intelligence*. North-Holland, 1997. (In press.).
- [17] D. Gabbay. Declarative Past and Imperative Future: Executable Temporal Logic for Interactive Systems. In B. Banieqbal, H. Barringer, and A. Pnueli, editors, *Proceedings of Colloquium on Temporal Logic in Specification*, pages 402–450, Altrincham, U.K., 1987. (Published in *Lecture Notes in Computer Science*, volume 398, Springer-Verlag).
- [18] K. Konolige. *A Deduction Model of Belief*. Pitman/Morgan Kaufmann, 1986.
- [19] T. Maruichi, M. Ichikawa, and M. Tokoro. Modelling Autonomous Agents and their Groups. In Y. Demazeau and J. P. Muller, editors, *Decentralized AI 2 – Proceedings of the 2nd European Workshop on Modelling Autonomous Agents and Multi-Agent Worlds (MAAMAW '90)*. Elsevier/North Holland, 1991.
- [20] M. Wooldridge. Agent-based software engineering. *IEE Proceedings on Software Engineering*, February 1997. To appear.
- [21] M. Wooldridge, C. Dixon, and M. Fisher. A Tableau-Based Proof Method for Temporal Logics of Knowledge and Belief. To be submitted, 1996.
- [22] M. Wooldridge and M. Fisher. A First-Order Branching Time Logic of Multi-Agent Systems. In *10th European Conference on Artificial Intelligence (ECAI)*, Vienna, Austria, August 1992. Wiley and Sons.
- [23] M. Wooldridge and M. Fisher. A decision procedure for a temporal belief logic. In *First International Conference on Temporal Logic (ICTL)*, Bonn, Germany, July 1994. (Published in *Lecture Notes in Computer Science*, volume 827, Springer-Verlag).
- [24] M. Wooldridge. *The Logical Modelling of Computational Multi-Agent Systems*. PhD thesis, Department of Computation, UMIST, Manchester, UK, 1992.