This is MYWORLD: The Logic of an Agent-Oriented DAI Testbed

Michael Wooldridge

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

M.Wooldridge@doc.mmu.ac.uk

Abstract. The ultimate goal of the work presented in this article is to develop practical frameworks for formally reasoning about multi-agent systems. Such frameworks are particularly important, as multi-agent approaches are inherently complex, and are already being applied in safety-critical domains such as air traffic control. The article is in three parts. The first contains an informal overview of MYWORLD, a testbed for experimentation in Distributed Artificial Intelligence (DAI). A MYWORLD system contains a number of concurrently executing agents, each of which is programmed along the lines proposed by Shoham in his Agent-Oriented Programming (AOP) proposal [22]. The second part of the article contains a detailed formal model of MYWORLD, which rigorously defines the possible states and state transitions of MYWORLD agents. The third part develops a logic that can be used to represent the properties of MYWORLD systems; this logic is closely related to the formal model of MYWORLD, in that the histories traced out in the execution of a system are used as the semantic basis for the logic. We comment on the applications of the logic, and conclude by indicating areas for future work.

1 Introduction

One area of much current interest in DAI is the use of mathematical logic for specifying the properties of agents and multi-agent systems. Probably the best-known example of this work is the Cohen-Levesque theory of intention [3]. To express this theory, the authors developed a quantified multi-modal logic, with modalities for representing beliefs and goals, and an apparatus for representing actions that was loosely based on dynamic logic [13]; beliefs and goals were characterised using possible worlds semantics [12]. Various concepts were then defined in a layered fashion, each layer building on the concepts introduced at the previous layer, until intention was introduced. The result was a theory that satisfies many of the properties that one would expect of intention, and which is also *formal*, with all the attendant advantages that rigour and formality bring. Building largely on this work, attempts have been made to use similar logics to capture various other properties of agents [18] and multi-agent systems [24].

Although this work is undoubtedly significant, it is important to realise what its limitations are. Agent theories expressed in a modal logic with possible worlds semantics can only serve as specifications in a very abstract sense; they cannot, in general, be refined into implementations in the conventional way[1]. There are several reasons for

[1] A notable exception is the *situated automata* paradigm [20].

this. First, there is no clear relationship between the logic used to express the specification, and the structure of (D)AI systems as they are typically built. In particular, it is not generally clear what possible worlds might correspond to in an implemented system. Secondly, such logics make unreasonable assumptions about the reasoning ability of agents: this is the logical omniscience problem [15]. For these reasons, we have proposed developing formalisms for reasoning about multi-agent systems where a precise relationship is maintained between the specification language and the systems we ultimately hope to build [23]. The technique depends on borrowing some ideas from mainstream computer science [16]. Specifically, we construct a formal model of the type of system we wish to reason about, and then use the execution histories traced out by such a system as the semantic foundation for a temporal logic, which can then be used to reason about the systems we are modelling. Since there is a precise relationship between the logic and the systems the logic is modelling, we can realistically claim that statements in the logic actually express properties of the systems we hope to model.

In this article, we demonstrate this approach by applying it to MYWORLD, an implemented testbed for DAI. A MYWORLD system contains a number of concurrently executing agents, each of which is programmed along the lines proposed by Shoham for his *agent-oriented programming* proposal [22]. Agents are given intentions and beliefs, and rules that define how they should generate and modify them. Additionally, they are given rules that represent information about how to achieve intentions; to achieve an intention, an agent must typically perform some actions. In the next section, we present a more detailed review of MYWORLD; this is followed, in section 3, by a formal model of MYWORLD agents and systems. In section 4, we present a logic called $\mathcal{L}_M$ (for MYWORLD logic), which can be used to represent and reason about the properties of MYWORLD systems. The logic extends classical first-order logic with the introduction of modalities for representing the beliefs, intentions, and actions of agents, as well as temporal modal connectives for representing the time-varying properties of MYWORLD systems. The article concludes with some comments on limitations of the work presented, and issues for future work.

Related work: Surprisingly few attempts have been made by researchers within (D)AI to develop formal models of real agents and multi-agent systems (although, as we observed above, many *abstract* models have been proposed [3, 18, 24]). Genesereth and Nilsson develop simple models of various classes of agents in [9, Chapter 13], though these models are also very abstract. Seel developed formal models of simple reactive agents, and investigated the use of temporal logics for reasoning about them [21]. The Z language has been used in a number of agent specifications: Goodwin used it to develop formal models various types of agents, and used these models to characterise various attributes of agency (such as capability, reactivity, rationality, and so on) [11]; a 240-page formal specification of the SOAR cognitive architecture has also been developed in Z [17]; and Craig has used Z to specify his CASSANDRA blackboard model [4]. Finally, in some work closely related to this article, Rao and Georgeff have considered the extent to which agents can be said to satisfy theories of agency [19]. However, they use a theory of agency expressed in a modal logic of beliefs, desires, and intentions, with a possible worlds semantics: such a logic is ultimately unsatisfactory for specifying

agents, for the reasons we outlined above.

Notational conventions: Most of the formalism in the article is presented using the VDM specification language: the first eight chapters of [14] cover the relevant material. Additionally, if $\mathcal{L}$ is a logical language, then we write $Form(\mathcal{L})$ for the set of (well-formed) formulae of $\mathcal{L}$.

2 A Glimpse of MyWorld

This section contains an informal overview of MyWorld, adapted from [26]. At run-time, a MyWorld system has four components:

- an *umpire*, or *world manager*, which has top-level control of the system;
- an *agent-oriented* language, for programming agents;
- a *world shell*, which defines the characteristics of an experimental 'world'; and
- a *scenario*, which represents a particular experiment.

The umpire is the part of MyWorld that has overall, top-level control of the system. Amongst other things, it is responsible for scheduling agent execution, monitoring the user interface, and responding to requests from the latter. The umpire is the 'generic' part of MyWorld, and was designed to allow as much flexibility and generality as possible. However, it was found necessary to build certain basic principles into it, the most important of which being the notions of animate and inanimate objects (animate objects being agents), time, space, events, and actions.

The world shell is the part of MyWorld that defines the environmental characteristics of a particular 'world' in which experiments may be performed. For example, the world shell defines what actions may be performed in an experiment, when such actions are legal, and what the results of an action performed in some particular situation are. Additionally, it defines what entities may appear in the world, and what their properties are. At the time of writing, we have implemented just one world shell; in future, we hope to construct a library of such shells, to allow experimentation in a variety of domains.

A scenario in MyWorld represents a specific experiment carried out. It describes the initial locations and properties of all the entities that appear in the experiment.

For the purposes of this article, the most important component of MyWorld is the agent-oriented part, for programming agents. This language characterises an agent in terms of five attributes: (i) a set of *beliefs*; (ii) a set of *intentions*; (iii) a set of *belief rules*; (iv) a set of *intention adoption rules*; and finally (v) a *belief revision function*. These components, and the way that they interact to generate the behaviour of an agent, will now be described in more detail.

Beliefs: An agent's beliefs represent that agent's information about the world it occupies: in more traditional AI terms, an agent's beliefs are its 'knowledge'. In the current implementation of MyWorld, an agent's beliefs are a set of ground atoms of first-order predicate logic (cf. PROLOG facts).

Belief sets are not fixed: they may change over time, by new beliefs being added and old beliefs being removed. New beliefs arise from three sources:

- from inferences made via *belief rules*;
- from perceiving the world; and
- from performing non-logical 'cognitive', or 'private' actions.

Belief rules define how new beliefs are generated from old ones. A belief rule corresponds closely to a rule in the standard AI sense: it has antecedent and consequent parts, and will 'fire' if the antecedent unifies with the agent's belief set. In the current implementation, belief rules are applied exhaustively, in a forward chaining fashion, to generate new beliefs.

Perception is modelled by a *belief revision function* (cf. [8]). This function looks at the state of the system and the agent's current beliefs, and generates a new set of beliefs as a result. Old beliefs may also be removed by a belief revision function.

Finally, cognitive, or private actions, correspond to an agent exploiting its internal computational resources. For example, imagine an agent consulting an internal database; this would be an action that was not visible to other agents — hence the term 'cognitive' action. The result of such an action would be some information, which appears in the form of new beliefs.

Intentions: Intentions represent desires that an agent will attempt to bring about. An intention contains a *goal* part, a *motivation* part, and a *rating* part. The goal represents the 'purpose' of the intention — if it is ever believed to be satisfied, then the intention is fulfilled. The motivation represents what must be believed in order for the intention to be maintained — if the motivation ever becomes false, then there is no point in maintaining the intention, and it is dropped. So an intention will be maintained until either its goal is believed to be satisfied or its motivation is no longer present[2]. The *rating* of an intention represents its priority; the higher the rating, the more important the intention. The highest rated intention is called the *current* intention, and will guide the actions of the agent, in a way that we describe below.

New intentions are generated via intention adoption rules (IARs). An IAR is a pair, containing:

- an *adoption condition*; and
- an *intention skeleton* (an intention containing variables).

The idea is that on every cycle, the agent tries to match the adoption condition of each IAR with its beliefs; if it succeeds, then the variables in the corresponding intention skeleton are instantiated, and the resulting intention is added to its intention set.

Strategy rules: Intentions are linked to actions by *strategy rules*. Strategy rules represent information about how to achieve intentions. A strategy rule is a pair, consisting of:

[2] This is very close to what Cohen and Levesque call a *persistent relativised goal* [3].

– a *condition*, corresponding to the goal of an intention;
– a *strategy function*, which takes as its sole argument the state of the agent, and returns an action, that the agent has chosen to perform.

Strategy rules loosely resemble knowledge areas in the PRS [10]. Strategy functions are the closest that agents in MYWORLD come to doing any planning; they may be thought of as crude procedural plans.

Agent execution: Let us now summarise the behaviour of an agent on a single scheduler cycle:

1. update beliefs through belief revision function; apply belief rules exhaustively;
2. update intentions by:
   – removing those that are no longer applicable; and
   – finding and adding those that have now become applicable;
3. find the highest rated intention and use strategy rules in order to find a strategy function;
4. evaluate the strategy function in order to find some action;
5. execute the action that results.

3 A Formal Model of MYWORLD

In this section, we construct a formal model of MYWORLD, focussing particularly on the agent-language component. In section 4, we use this formal model as the foundation upon which to construct a logic for representing and reasoning about the properties of MYWORLD systems.

Beliefs and belief rules: In the current implementation of MYWORLD, beliefs are simply *ground atoms* of first-order logic: beliefs are thus similar to PROLOG facts. To represent beliefs, we require a set of *terms*, made up of a set of *constants* and a set of *variables*. We also require a set of *predicate symbols*.

$$Const = \{a, b, c, \ldots\} \qquad Var = \{x, y, z, \ldots\}$$

$$Term = Const \cup Var \qquad Pred = \{P, Q, \ldots\}$$

Notice that the only functional terms allowed are constants. An *atom* is an application of a predicate symbol, (called the *head* of the atom), to a list of terms.

$$\begin{array}{lrcl} Atom :: & head & : & Pred \\ & term\text{-}list & : & Term^* \end{array}$$

A function *at-vars* is defined, which takes an atom and returns the set of variables it contains.

$$at\text{-}vars : Atom \rightarrow Var\text{-set}$$

$$at\text{-}vars(mk\text{-}Atom(P, tl)) \triangleq Var \cap \mathsf{elems}\, tl$$

A *ground* atom is one containing no variables.

$$GAtom = Atom \quad \text{where} \quad inv\text{-}GAtom(at) \triangleq at\text{-}vars(at) = \{\ \}$$

The set of possible beliefs is defined to be the set of ground atoms.

$$Bel = GAtom$$

Agents are able to reason by applying *belief rules* to their belief set. Conceptually, a belief rule corresponds to a first-order formula of the form

$$\forall \overline{x} \cdot \varphi(\overline{x}, \overline{a}) \Rightarrow \psi(\overline{y}, \overline{b})$$

where $\overline{x} = x_1, \ldots, x_m$ and $\overline{y} = y_1, \ldots, y_n$ are tuples of variables, such that $\{y_1, \ldots, y_n\} \subseteq \{x_1, \ldots, x_m\}$, $\overline{a} = a_1, \ldots, a_o$ and $\overline{b} = b_1, \ldots, b_p$ are tuples of constants, and $\varphi(\overline{x}, \overline{a})$ and $\psi(\overline{y}, \overline{b})$ are conditions. Note that variables in the consequent must appear in the antecedent. This form of rule is, of course, very similar to that which appears throughout AI; the way in which such a rule may be applied is obvious. For simplicity, we shall assume that conditions may contain only conjunctions; we do not consider disjunctions or negations. This relieves us of the need to deal with issues such as negation as failure, which would otherwise obscure more important points. We define a type for conditions.

$$Cond = Atom^*$$

A condition is thus a sequence of predicates, representing their conjunction. The function *at-vars* is extended to conditions, so that *vars* takes a condition and returns the set of variables it contains.

$$vars : Cond \rightarrow Var\text{-set}$$

$$vars(c) \triangleq \bigcup \{at\text{-}vars(at) \mid at \in \text{elems}\, c\}$$

Antecedents and consequents are then simply conditions; a belief rule is a pair containing an antecedent and a consequent, with the restriction on variables as above.

$$\begin{array}{rrcl} BelRule :: & ante & : & Cond \\ & conse & : & Cond \end{array}$$

$$\text{inv}\ (mk\text{-}BelRule(an, cn)) \triangleq vars(cn) \subseteq vars(an)$$

The applicability of a belief rule with respect to a set of beliefs is determined by unification of the antecedent in the belief set; as we have no functional terms other than constants, unification is a straightforward process. First, a *binding* is defined to be a map from variables to constants.

$$Binding = Var \xrightarrow{m} Const$$

Next, a function *at-apply* is defined, which applies a binding to an atom: the function returns the atom that results from replacing all variables with the constant they are bound to.

$$\mathit{at\text{-}apply} : \mathit{Binding} \times \mathit{Atom} \to \mathit{Atom}$$

$$\begin{array}{l}\mathit{at\text{-}apply}(\beta, at) \triangleq \\ \quad \mathsf{let}\ tl = \mathit{term\text{-}list}(at)\ \mathsf{in} \\ \quad \mathsf{let}\ tl' = \{n \mapsto a \mid (n \in \mathsf{inds}\, tl) \wedge (tl(n) = x) \wedge (x \in \mathit{Var}) \wedge (\beta(x) = a)\}\ \mathsf{in} \\ \quad \mathsf{let}\ tl'' = tl \dagger tl'\ \mathsf{in} \\ \quad \mu(at, \mathit{term\text{-}list} \mapsto tl'')\end{array}$$

The function *at-unifiers* takes a belief set and an atom, and returns that set of bindings which, when applied to the atom, yield members of the belief set.

$$\mathit{at\text{-}unifiers} : \mathit{Atom} \times \mathit{Bel}\text{-set} \to \mathit{Binding}\text{-set}$$

$$\mathit{at\text{-}unifiers}(at, bs) \triangleq \{\beta \mid (\beta \in \mathit{Binding}) \wedge (\mathit{at\text{-}apply}(\beta, at) \in bs)\}$$

The function *at-apply* is extended to the function *apply*, which applies a binding to a condition.

$$\mathit{apply} : \mathit{Binding} \times \mathit{Cond} \to \mathit{Cond}$$

$$\mathit{apply}(\beta, c) \triangleq \{n \mapsto \mathit{at\text{-}apply}(\beta, c(n)) \mid n \in \mathsf{inds}\, c\}$$

The function *unifiers* is a similar extension of *at-unifiers*.

$$\mathit{unifiers} : \mathit{Cond} \times \mathit{Bel}\text{-set} \to \mathit{Binding}\text{-set}$$

$$\mathit{unifiers}(c, bs) \triangleq \bigcap\{\mathit{at\text{-}unifiers}(at, bs) \mid at \in \mathsf{elems}\, c\}$$

It is convenient to define a boolean-valued function *fires*, which takes a condition and a belief set, and returns true iff the condition is satisfied by the belief set.

$$\mathit{fires} : \mathit{Cond} \times \mathit{Bel}\text{-set} \to \mathbb{B}$$

$$\mathit{fires}(c, bs) \triangleq \mathit{unifiers}(c, bs) \neq \{\,\}$$

The function *fire-belrule* takes a belief rule and a set of beliefs, and returns the set of beliefs that results from firing the rule. If the rule cannot fire, then the function returns the empty set.

$$\mathit{fire\text{-}belrule} : \mathit{BelRule} \times \mathit{Bel}\text{-set} \to \mathit{Bel}\text{-set}$$

$$\begin{array}{l}\mathit{fire\text{-}belrule}(br, bs) \triangleq \\ \quad \bigcup\{\mathsf{elems}\, \mathit{apply}(\beta, \mathit{conse}(br)) \mid \beta \in \mathit{unifiers}(\mathit{ante}(br))\}\end{array}$$

Finally, a function *close* is defined, which returns the closure of a belief set under some belief rules; that is, it returns the belief set that results from exhaustively applying the rules to the belief set. Note that the definition of *close* makes use of an auxilliary function *close-aux*.

$$\mathit{close\text{-}aux} : \mathbb{N} \times \mathit{BelRule}\text{-set} \times \mathit{Bel}\text{-set} \to \mathit{Bel}\text{-set}$$

$$\begin{array}{l}\mathit{close\text{-}aux}(u, brs, bs) \triangleq \\ \quad \mathsf{if}\ u = 0 \\ \quad \mathsf{then}\ \{at \mid \exists br \in brs \cdot at \in \mathit{fire\text{-}belrule}(br, bs)\} \\ \quad \mathsf{else}\ \{at \mid \exists br \in brs \cdot at \in \mathit{fire\text{-}belrule}(br, \mathit{close\text{-}aux}(u - 1, brs, bs))\}\end{array}$$

$$close : BelRule\text{-set} \times Bel\text{-set} \rightarrow Bel\text{-set}$$

$$close(brs, bs) \triangleq \bigcup\{close\text{-}aux(u, brs, bs) \mid u \in \mathbb{N}\}$$

Finally, we look at belief revision. In MYWORLD, a belief revision function maps an environment state and a belief set into a new belief set; the type *Env*, for environment state, is defined later.

$$BRF = Env \times Bel\text{-set} \rightarrow Bel\text{-set}$$

Intentions and intention adoption rules: An intention is a triple, containing: (i) a *goal* part; (ii) a *motivation* part; and (iii) a *rating*. The goal represents what the agent would believe if the intention was satisfied; the motivation represents what must be believed by the agent in order for the intention to be maintained; and the rating represents how important the intention is considered to be. We begin by defining *intention skeletons*. An intention skeleton is essentially an intention that can have a variable for a rating.

$$\begin{array}{lrcl} IntSk :: & goal & : & Cond \\ & motivation & : & Cond \\ & rating & : & \mathbb{N} \cup Var \end{array}$$

An intention is then an intention skeleton that does not have a variable for a rating.

$$Int = IntSk \quad \text{where} \quad inv\text{-}Int(mk\text{-}Int(g, m, r)) \triangleq r \notin Var$$

Note that variables may appear in the goal or motivation parts of an intention, in which case they are considered to be existentially quantified. An *intention adoption rule* is a pair, consisting of an intention skeleton and an adoption condition.

$$\begin{array}{lrcl} IARule :: & adcond & : & Cond \\ & intsk & : & IntSk \end{array}$$

$$\text{inv } (mk\text{-}IARule(c, mk\text{-}IntSk(g, m, r))) \triangleq r \in Var \Rightarrow r \in vars(c)$$

The invariant on *IARule* ensures that if the rating part of the skeleton is a variable, then it is one of the variables that occurs in the adoption condition. Thus, when the rule fires, this variable will be instantiated. The function *iar-apply* takes an intention adoption rule and a binding, and returns the intention that results from applying the binding to all variables that occur in the intention skeleton.

$$iar\text{-}apply : Binding \times IARule \rightarrow Int$$

$$\begin{array}{l} iar\text{-}apply(\beta, iar) \triangleq \\ \quad \text{let } g = goal(intsk(iar)) \text{ in} \\ \quad \text{let } m = motivation(intsk(iar)) \text{ in} \\ \quad \text{let } r = rating(intsk(iar)) \text{ in} \\ \quad \text{let } r' = \text{if } in \in Var \text{ then } \beta(r) \text{ else } r \\ \quad mk\text{-}Int(apply(\beta, g), apply(\beta, m), r') \end{array}$$

The function *fire-iarule* takes an intention adoption rule and a belief set, and returns the set of intentions made current by the rule. Once again, the applicability of the rule is determined by unification in the belief set. Note that if the rule cannot fire, then this function returns the empty set.

$$\mathit{fire\text{-}iarule} : \mathit{IARule} \times \mathit{Bel}\text{-set} \rightarrow \mathit{Int}\text{-set}$$

$$\mathit{fire\text{-}iarule}(\mathit{iar}, \mathit{bs}) \triangleq \{\mathit{iar\text{-}apply}(\beta, \mathit{iar}) \mid \beta \in \mathit{unifiers}(\mathit{adcond}(\mathit{iar}), \mathit{bs})\}$$

The boolean-valued function *satisfied-int* takes an intention and a belief set, and returns true iff the intention is satisfied with respect to the belief set.

$$\mathit{satisfied\text{-}int} : \mathit{Int} \times \mathit{Bel}\text{-set} \rightarrow \mathbb{B}$$

$$\mathit{satisfied\text{-}int}(\mathit{int}, \mathit{bs}) \triangleq \mathit{fires}(\mathit{goal}(\mathit{int}), \mathit{bs})$$

The boolean-valued function *applicable-int* takes an intention and a belief set, and returns true iff the intention is still applicable with respect to the belief set, i.e., if the motivation is still present.

$$\mathit{applicable\text{-}int} : \mathit{Int} \times \mathit{Bel}\text{-set} \rightarrow \mathbb{B}$$

$$\mathit{applicable\text{-}int}(\mathit{int}, \mathit{bs}) \triangleq \mathit{fires}(\mathit{motivation}(\mathit{int}), \mathit{bs})$$

The function *update-intentions* takes a set of intentions, (representing those currently held by an agent), a set of beliefs, (also representing those currently held by the agent), and a set of intention adoption rules, and returns the set that results by removing those that are no longer applicable, or that are satisfied, and adding those that have become applicable.

$$\mathit{update\text{-}intentions} : \mathit{Int}\text{-set} \times \mathit{Bel}\text{-set} \times \mathit{IARule}\text{-set} \rightarrow \mathit{Int}\text{-set}$$

$$\begin{aligned}
&\mathit{update\text{-}intentions}(\mathit{ints}, \mathit{bs}, \mathit{iars}) \triangleq \\
&\quad \text{let } \mathit{sat} = \{\mathit{int} \mid (\mathit{int} \in \mathit{ints}) \wedge \mathit{satisfied\text{-}int}(\mathit{int}, \mathit{bs})\} \text{ in} \\
&\quad \text{let } \mathit{inap} = \{\mathit{int} \mid (\mathit{int} \in \mathit{ints}) \wedge \neg\, \mathit{applicable\text{-}int}(\mathit{int}, \mathit{bs})\} \text{ in} \\
&\quad \text{let } \mathit{new} = \bigcup\{\mathit{fire\text{-}iarule}(\mathit{iar}, \mathit{bs}) \mid \mathit{iar} \in \mathit{iars}\} \text{ in} \\
&\quad \text{let } \mathit{ints}' = (\mathit{ints} - (\mathit{sat} \cup \mathit{inap})) \cup \mathit{new} \text{ in} \\
&\quad \text{let } \mathit{sat}' = \{\mathit{int} \mid (\mathit{int} \in \mathit{ints}') \wedge \mathit{satisfied\text{-}int}(\mathit{int}, \mathit{bs})\} \text{ in} \\
&\quad \text{let } \mathit{inap}' = \{\mathit{int} \mid (\mathit{int} \in \mathit{ints}') \wedge \neg\, \mathit{applicable\text{-}int}(\mathit{int}, \mathit{bs})\} \text{ in} \\
&\quad \mathit{ints}' - (\mathit{sat}' \cup \mathit{inap}')
\end{aligned}$$

This function ensures that newly adopted intentions will be consistent with an agent's beliefs: intentions will not be adopted if they are believed to be satisfied, or if there is no motivation for them. Finally, the function *highest-rated* takes a set of intentions, representing those currently held by an agent, and returns the highest rated of these.

$$\begin{aligned}
&\mathit{highest\text{-}rated}\ (\mathit{ints}\text{: }\mathit{Int}\text{-set})\ \mathit{int}\text{: }\mathit{Int} \\
&\text{pre } \mathit{ints} \neq \{\,\} \\
&\text{post } \mathit{int} \in \mathit{ints} \wedge \neg\, (\exists \mathit{int}' \in \mathit{ints} \cdot \mathit{rating}(\mathit{int}') > \mathit{rating}(\mathit{int}))
\end{aligned}$$

Strategy rules: A strategy rule represents information about how to achieve intentions. In the current implementation, a strategy rule has a *condition* part and a *strategy function* part. A strategy function is best thought of as a kind of procedural plan, that operates on an agent's internal state to generate an *action*, representing that which the agent has chosen to perform. The condition part of a strategy rule determines the circumstances under which the associated strategy is applicable.

$$Ac = \ldots$$

$$Strat = Bel\text{-set} \times Int\text{-set} \rightarrow Ac$$

$$\begin{array}{lll} StratRule & :: & cond : Cond \\ & & strat : Strat \end{array}$$

(The content of the set *Ac*, of actions, is not significant.) We shall demand that an agent's set of strategy rules is *strongly complete*, in the sense that, for any given set of beliefs, they are guaranteed to pick out precisely one strategy function[3]. This ensures that an agent is never uncertain about which strategy function to apply. This notion of strong completeness is formalised in the following boolean-valued function.

$$strongly\text{-}complete : StratRule\text{-set} \rightarrow \mathbb{B}$$

$$strongly\text{-}complete(srs) \triangleq \forall bs \in Bel\text{-set} \cdot \exists!\, sr \in srs \cdot fires(cond(sr), bs)$$

The function *chosen-strategy* takes a strategy rule set and a set of beliefs, and returns the strategy picked out by the rule set.

$$\begin{array}{l} chosen\text{-}strategy\ (srs{:}\ StratRule\text{-set}, bs{:}\ Bel\text{-set})\ st{:}\ Strat \\ \mathsf{pre}\ strongly\text{-}complete(srs) \\ \mathsf{post}\ \exists str \in srs \cdot fires(cond(str), bs) \wedge (strat(str) = st) \end{array}$$

Agents and agent operation: We now have all the definitions required to introduce a type for agents.

$$\begin{array}{lrl} Agent & :: & bel : Bel\text{-set} \\ & & int : Int\text{-set} \\ & & br : BelRule\text{-set} \\ & & iar : IARule\text{-set} \\ & & sr : StratRule\text{-set} \\ & & brf : BRF \end{array}$$

$$\begin{array}{l} \mathsf{inv}\ (mk\text{-}Agent(bs, ints, brs, iars, srs, brf)) \triangleq \\ \quad strongly\text{-}complete(srs) \wedge \\ \quad \forall int \in ints \cdot applicable\text{-}int(int, bs) \wedge \neg\, satisfied\text{-}int(int, bs) \end{array}$$

The second invariant ensures that the agent's intentions are consistent with its beliefs.

Next, we define a next-state function for agents. This function captures the idea of an agent observing its environment, and updating its beliefs on the basis of its observations, applying its belief rules where possible, and then updating its intentions in light of its new beliefs. (The type *Env*, for environment state, is defined below.)

[3] *Weak* completeness would mean that the rules picked out *at least one* action [23].

$$\mathit{agent\text{-}next\text{-}state} : \mathit{Agent} \times \mathit{Env} \to \mathit{Agent}$$

$$\begin{array}{l}\mathit{agent\text{-}next\text{-}state}(\mathit{mk\text{-}Agent}(\mathit{bs}, \mathit{ints}, \mathit{brs}, \mathit{iars}, \mathit{srs}, \mathit{brf}), \mathit{env}) \triangleq \\ \quad \mathsf{let}\ \mathit{bs}' = \mathit{brf}(\mathit{env}, \mathit{bs})\ \mathsf{in} \\ \quad \mathsf{let}\ \mathit{bs}'' = \mathit{close}(\mathit{bs}', \mathit{brs})\ \mathsf{in} \\ \quad \mathsf{let}\ \mathit{ints}' = \mathit{update\text{-}intentions}(\mathit{ints}, \mathit{bs}'', \mathit{iars})\ \mathsf{in} \\ \quad \mathit{mk\text{-}Agent}(\mathit{bs}'', \mathit{ints}', \mathit{brs}, \mathit{iars}, \mathit{srs}, \mathit{brf})\end{array}$$

Finally, we define a function *chosen-action*, which takes an agent and returns the action that the agent has chosen to perform.

$$\mathit{chosen\text{-}action} : \mathit{Agent} \to \mathit{Ac}$$

$$\begin{array}{l}\mathit{chosen\text{-}action}(\mathit{mk\text{-}Agent}(\mathit{bs}, \mathit{ints}, \mathit{brs}, \mathit{iars}, \mathit{srs}, \mathit{brf})) \triangleq \\ \quad \mathit{chosen\text{-}strategy}(\mathit{srs}, \mathit{bs})(\mathit{bs}, \mathit{ints})\end{array}$$

Systems: A MyWorld system may be regarded as containing a set of named entities, which fall into two categories: inanimate entities, or *objects*, which have *attributes*, but do not have any internal structure and are not able to change the state of the system, and agents. The idea is that objects correspond to chairs, books, pints of beer, pieces of string, and so on: things which do not originate actions, but rather have actions performed on them. Agents are things that originate actions. We require types for attributes and names.

$$\mathit{Attribute} = \ldots \qquad \mathit{Name} = \ldots$$

(The content of these two sets is not significant.) The state of a MyWorld system at some point during execution may then be characterised by two maps, which associate names with attributes and agents respectively. A value of this type represents a 'snapshot' of a MyWorld system during execution. We introduce a separate type for environment state, which represents the state of every object in the system.

$$\mathit{Env} = \mathit{Name} \xrightarrow{m} \mathit{Attribute}\text{-set}$$

$$\begin{array}{lll}\mathit{Sys} :: & \mathit{objects} & : \mathit{Env} \\ & \mathit{agents} & : \mathit{Name} \xrightarrow{m} \mathit{Agent}\end{array}$$

Finally, we model a world shell as a function that takes an action and an environment state, and returns the returns the environment state that results from the attempted performance of the action. Such a function represents the 'natural laws' of an experimental world, and the environmental constraints that hold in it.

$$\mathit{World} = \mathit{Ac} \times \mathit{Env} \to \mathit{Env}$$

Note that this definition implicitly assumes that agents are not directly affected by the performance of an action.

4 The Logic of MyWorld

In this section, we develop the logic $\mathcal{L}_M$, which can be used to represent the properties of MyWorld systems. $\mathcal{L}_M$ is closely related to the formal model of MyWorld that

we constructed in the preceding section: we use the *histories* traced out in the execution of a system as its semantic basis. Although this technique has long been used in mainstream computer science [16], it has only recently been applied in DAI [23]. This earlier work made two limiting assumptions. First, a very simple model of agents was used. Secondly, it was assumed that agents act in *synchrony*, rather than *concurrently*. In developing $\mathcal{L}_M$, we make no such assumptions: $\mathcal{L}_M$ is based on a more realistic model of both agents and their execution.

Semantic concepts: We begin by setting the scene with a short discussion on agent execution. Consider the behaviour of an agent during a single scheduler cycle. It begins by perceiving its environment, updating its beliefs through its belief revision function, then updating its intentions, and so on, until finally, it executes an action. A direct attempt to model this behaviour would lead us to a number of difficulties. For example, what are an agent's beliefs while it is applying its belief rules? What action is it performing during this time? To avoid such problems, we assume that agents update their internal state *instantaneously* at the beginning of a scheduler cycle, and spend the rest of that cycle with fixed beliefs and intentions, performing their chosen action. The results of an action are assumed to come into effect at the end of the cycle. Finally, we shall also assume that once an agent's scheduler cycle is complete, it immediately begins another, without any pause.

Before developing the structures used to represent execution histories, we must fix on a model of time. We choose to let time be *linear*, (i.e., there is only one 'timeline'), *bounded in the past* (i.e., there was a time at which system execution began), and *infinite in the future*, (i.e., the system is non-terminating). Unusually, we also choose to let time be *dense*, meaning that for any two time points, it is possible to find a third between them. A convenient temporal model is thus $(\mathbb{R}^+, <)$, i.e., the positive reals ordered by the less-than relation. The reason for fixing on such a model is that it allows us to represent 'real' concurrency with comparative ease [2]. The use of the temporal logic of reals for modelling the behaviour of a group of agents was first proposed by Fisher, who used the technique to give a semantics to his Concurrent METATEM language [5].

We now introduce the technical apparatus for dealing with time. An *interval* over $\mathbb{R}^+$ between $x, y \in \mathbb{R}^+$, where $x < y$, is the subset of $\mathbb{R}^+$ that falls between x and y.

$$interval : \mathbb{R}^+ \times \mathbb{R}^+ \rightarrow \mathbb{R}^+\text{-set}$$

$$interval(x,y) \quad \triangleq \quad \{z \mid (z \in \mathbb{R}^+) \wedge (x \leq z < y)\}$$

The set of all intervals is *Interval*.

$$Interval = \{interval(x,y) \mid (x,y \in \mathbb{R}^+) \wedge (x < y)\}$$

We assume two functions *start* and *end*, which give the start and end points of an interval, respectively — their formal definitions are trivial, and are therefore omitted. To represent execution histories, we essentially use functions that map times to the state of the entity they are modelling. The first such function is *os*, which gives the state of every object in the system.

$$os : \mathbb{R}^+ \rightarrow Env$$

To model the time-varying state of agents, we use two functions. The first, *cycle*, takes an agent name and a scheduler cycle number, and gives that interval of that cycle. Note that scheduler cycles are indexed by the natural numbers, and we thus assume that each agent has a countably infinite number of such cycles; this is a kind of (rather extreme) *fairness* assumption [7].

$$cycle\colon Name \times \mathbb{N} \to Interval$$

The second, *as*, takes a time and returns a map that gives the state of every agent at that time.

$$as\colon \mathbb{R}^{+} \to (Name \xrightarrow{m} Agent)$$

Unfortunately, the situation is complicated by the fact that various relationships exist between the entities in a system. For example, once an agent performs an action, we expect the effects of that action to be reflected in the subsequent state of the system. To capture these relationships, we place a number of constraints on the functions representing execution histories.

Constraint 1: An agent's scheduler cycles meet each other [1]. For example, if an agent's first cycle extends from t to t', and its second extends from t'' to t''', then $t' = t''$. Formally, this constraint is expressed as follows.

$$\forall i \in Name \cdot \forall u \in \mathbb{N} \cdot end(cycle(i, u)) = start(cycle(i, u + 1))$$

Constraint 2: Every agent's state is fixed within its scheduler cycles.

$$\forall i \in Name \cdot \forall u \in \mathbb{N} \cdot cycle(i, u) = \iota \Rightarrow \forall x, y \in \iota \cdot as(x)(i) = as(y)(i)$$

Constraint 3: The end point of every scheduler cycle is unique.

$$\forall i, j \in Name \cdot (i \neq j) \Rightarrow \forall u, v \in \mathbb{N} \cdot end(cycle(i, u)) \neq end(cycle(j, v))$$

Constraint 4: An agent's internal state during a cycle is a result of perceiving the world at the beginning of that cycle.

$$\begin{array}{l}
\forall i \in Name \cdot \\
\quad \forall u \in \mathbb{N}_1 \cdot \\
\qquad \text{let } \iota = cycle(i, u - 1) \text{ in} \\
\qquad \text{let } \iota' = cycle(i, u) \text{ in} \\
\qquad \text{let } \textit{prev-ag-st} = as(start(\iota)) \text{ in} \\
\qquad \text{let } \textit{cur-ag-st} = as(start(\iota')) \text{ in} \\
\qquad \text{let } \textit{cur-obj-st} = os(start(\iota')) \text{ in} \\
\qquad \textit{cur-ag-st}(i) = \textit{agent-next-state}(\textit{prev-ag-st}(i), \textit{cur-obj-st})
\end{array}$$

Constraint 5: Actions have effects: they change the state of the system in which they are executed. Recall from the preceding section that the effect an action has on a system is determined by a world function. Also, we assumed that an action only achieves its effects at the end of a cycle, at which time the system instantaneously changes state. This leads to the following constraint, which can only be expressed with respect to some world function $\omega \in World$.

$$
\begin{array}{lll}
\langle \mathit{fmla} \rangle ::= \langle \mathcal{L}_0\text{-}\mathit{fmla} \rangle & & \mid \forall \langle \mathit{var} \rangle \cdot \langle \mathit{fmla} \rangle \\
\quad \mid \neg \langle \mathit{fmla} \rangle & & \mid \langle \mathit{fmla} \rangle \vee \langle \mathit{fmla} \rangle \\
\quad \mid \langle \mathit{fmla} \rangle \, \mathcal{U} \, \langle \mathit{fmla} \rangle & & \mid \langle \mathit{fmla} \rangle \, \mathcal{S} \, \langle \mathit{fmla} \rangle \\
\quad \mid (\mathsf{Bel}\ \langle \mathit{name} \rangle\ \langle \mathcal{L}_0\text{-}\mathit{fmla} \rangle) & & \mid (\mathsf{Intend}\ \langle \mathit{name} \rangle\ \langle \mathcal{L}_0\text{-}\mathit{fmla} \rangle\ \langle \mathcal{L}_0\text{-}\mathit{fmla} \rangle\ \langle \mathit{nn} \rangle) \\
\quad \mid (\mathsf{Do}\ \langle \mathit{name} \rangle\ \langle \mathit{ac} \rangle) & &
\end{array}
$$

Fig. 1. Syntax of $\mathcal{L}_M$

$$
\begin{array}{l}
\forall i \in \mathit{Name} \cdot \\
\quad \forall u \in \mathbb{N}_1 \cdot \\
\qquad \mathsf{let}\ \iota = \mathit{cycle}(i, u-1)\ \mathsf{in} \\
\qquad \mathsf{let}\ \iota' = \mathit{cycle}(i, u)\ \mathsf{in} \\
\qquad \mathsf{let}\ \mathit{prev\text{-}obj\text{-}st} = \mathit{os}(\max \iota)\ \mathsf{in} \\
\qquad \mathsf{let}\ \mathit{cur\text{-}obj\text{-}st} = \mathit{os}(\mathit{start}(\iota'))\ \mathsf{in} \\
\qquad \mathsf{let}\ \mathit{prev\text{-}ag\text{-}st} = \mathit{as}(\mathit{start}(\iota))(i)\ \mathsf{in} \\
\qquad \mathsf{let}\ \alpha = \mathit{chosen\text{-}action}(\mathit{prev\text{-}ag\text{-}st})\ \mathsf{in} \\
\qquad \mathit{cur\text{-}obj\text{-}st} = \omega(\alpha, \mathit{prev\text{-}obj\text{-}st})
\end{array}
$$

There are other constraints that we might wish to place on these functions. For example, we might specify that the system remains unchanged except for the performance of actions within it. However, we shall leave such refinements for future work.

Syntax: $\mathcal{L}_M$ is a *quantified multi-modal* logic. It extends classical first-order logic by the introduction of a set of temporal modal connectives for representing the time-varying properties of MyWorld systems, as well as three further connectives, for representing the *beliefs*, *intentions*, and *actions* of agents within a system. For convenience, we shall assume an underlying classical first-order logic $\mathcal{L}_0$, defined over the sets *Pred* of predicate symbols, *Var* of variable symbols, and *Const* of constant symbols (see section 3). The syntax of the logic is defined by the grammar in Figure 1. The terminal symbols, other than literals, that appear in this grammar are interpreted as follows: $\langle \mathcal{L}_0\text{-}\mathit{fmla} \rangle \in \mathit{Form}(\mathcal{L}_0)$, $\langle \mathit{var} \rangle \in \mathit{Var}$, $\langle \mathit{name} \rangle \in \mathit{Name}$, $\langle \mathit{nn} \rangle \in \mathbb{N}$, and $\langle \mathit{ac} \rangle \in \mathit{Ac}$.

The temporal connectives $\mathcal{U}$ and $\mathcal{S}$ are called *until* and *since*, respectively. The other modal connectives, Bel, Intend, and Do, are for representing the beliefs, intentions and actions of agents, respectively. The classical connectives $\vee$ and $\neg$ have their standard interpretation, as does the universal quantifier $\forall$; the existential quantifier and the remaining connectives of classical logic are introduced as abbreviations, in the standard way.

Semantics: The semantics of $\mathcal{L}_M$-formulae are defined by a set of semantic rules, each of the form $\mathcal{I} \models \varphi$, where $\mathcal{I}$ is an *interpretation structure* and φ is a formula. Such an expression means that the structure $\mathcal{I}$ *satisfies* φ; the symbol '$\models$' is called the *satisfaction relation*. For $\mathcal{L}_M$, interpretation structures are triples of the form $\langle M, \beta, t \rangle$, where M is a

$\langle M, \beta, t\rangle \models P(\tau_1, \ldots, \tau_n)$	iff $\mathit{at\text{-}apply}(\beta, \mathit{mk\text{-}Atom}(P, [\tau_1, \ldots, \tau_n])) \in$ $\pi(\mathit{mk\text{-}Sys}(\mathit{os}(t), \mathit{as}(t)))$
$\langle M, \beta, t\rangle \models \neg\, \varphi$	iff $\langle M, \beta, t\rangle \not\models \varphi$
$\langle M, \beta, t\rangle \models \varphi \vee \psi$	iff $\langle M, \beta, t\rangle \models \varphi$ or $\langle M, \beta, t\rangle \models \psi$
$\langle M, \beta, t\rangle \models \forall x \cdot \varphi$	iff $\langle M, \beta \dagger \{x \mapsto a\}, t\rangle \models \varphi$ for all $a \in \mathit{Const}$
$\langle M, \beta, t\rangle \models \varphi\, \mathcal{U}\, \psi$	iff $\exists t' \in \mathbb{R}^+$ s.t. $(t < t')$ and $\langle M, \beta, t'\rangle \models \psi$ and $\forall t'' \in \mathbb{R}^+$ if $(t < t'' < t')$, then $\langle M, \beta, t''\rangle \models \varphi$
$\langle M, \beta, t\rangle \models \varphi\, \mathcal{S}\, \psi$	iff $\exists t' \in \mathbb{R}^+$ s.t. $(t' < t)$ and $\langle M, \beta, t'\rangle \models \psi$ and $\forall t'' \in \mathbb{R}^+$ if $(t' < t'' < t)$, then $\langle M, \beta, t''\rangle \models \varphi$
$\langle M, \beta, t\rangle \models$ (Bel $i\ \varphi$)	iff $\mathit{apply}(\beta, \varphi) \in \mathit{bel}(\mathit{as}(t)(i)) \cup \mathit{br}(\mathit{as}(t)(i))$
$\langle M, \beta, t\rangle \models$ (Intend $i\ \varphi\ \psi\ n$)	iff $\mathit{mk\text{-}Int}(\mathit{apply}(\beta, \varphi), \mathit{apply}(\beta, \psi), n) \in \mathit{int}(\mathit{as}(t)(i))$
$\langle M, \beta, t\rangle \models$ (Do $i\ \alpha$)	iff $\alpha = \mathit{chosen\text{-}action}(\mathit{as}(t)(i))$

Fig. 2. Semantics of $\mathcal{L}_M$

logical model for $\mathcal{L}_M$, $\beta \in \mathit{Binding}$ is a binding, and $t \in \mathbb{R}^+$ is a *reference time*. Logical models for $\mathcal{L}_M$ are themselves structures, which have the form:

$$M = \langle \mathit{os}, \mathit{as}, \mathit{cycle}, \pi \rangle$$

where *os*, *as*, and *cycle* are functions with constraints (1)–(4) holding between them as defined earlier, and

$$\pi : \mathit{Sys} \rightarrow \mathit{GAtom}\text{-set}$$

is an interpretation function, which takes a system and returns the set of ground atoms that represent the properties of, and relationships between, the various objects in that system[4]. The semantics of $\mathcal{L}_M$ are given in Figure 2. Note that the rules make use of various auxiliary functions (such as *apply*) that we defined in section 3. These functions are syntactic abbreviations: they serve only to simplify the statement of the formal semantics, and play no other part in the interpretation process.

Note that world functions do not appear in logical models. If we have a model M and world $\omega \in \mathit{World}$ such that the components of M satisfy Constraint 5 with respect to ω, then we say that M satisfies the laws of the world ω, and write M_ω to indicate this.

We now discuss the non-standard connectives of $\mathcal{L}_M$. First, the temporal modal connectives. A formula $\varphi\, \mathcal{U}\, \psi$ means that at some time in the future, ψ is satisfied, and *until* then, φ is satisfied. A formula $\varphi\, \mathcal{S}\, \psi$ means that ψ was satisfied at some time in the past, and *since* that time, φ has been satisfied. Using just these two connectives, we may define the remaining connectives of linear temporal logic (notice, however, that as time is *dense*, there is no *next time* connective in $\mathcal{L}_M$). First, $\Diamond \varphi$ means that either now or at some time in the future, φ is satisfied; $\Box \varphi$ means that φ is satisfied now and at all future times. We can define these connectives as follows:

[4] If we were defining the semantics of, for example, first-order logic, then we would make π a function that gave the extension of each predicate symbol.

$$\Diamond \varphi \triangleq \mathsf{true}\, \mathcal{U}\, \varphi \qquad \Box \varphi \triangleq \neg \Diamond \neg \varphi.$$

(In the first of these definitions, true is any classical tautology.) Just as the $\mathcal{S}$ connective mirrors the behaviour of $\mathcal{U}$ in the past, so we can define two unary connectives, $\blacklozenge$ and $\blacksquare$, that mirror the behaviour of $\Diamond$ and $\Box$ in the past (we omit the formal definitions, as these are very similar to those for $\Diamond$ and $\Box$).

We also have a *weak* version of the $\mathcal{U}$ connective: $\mathcal{W}$ allows for the possibility that its second argument is never satisfied.

$$\varphi\, \mathcal{W}\, \psi \triangleq \varphi\, \mathcal{U}\, \psi \vee \Box \varphi$$

A past time version of the $\mathcal{W}$ connective is $\mathcal{Z}$ ('zince').

Turning to the connectives for representing the properties of agents, a formula (Bel i φ) means that the agent i believes φ. In the current implementation, this means that either φ is one of the facts present in i's belief set, or φ is one of i's belief rules. A formula (Intend i φ ψ n) means that i intends to achieve φ with respect to motivation ψ and rating $n \in \mathbb{N}$. A formula (Do i α) means that i is doing action α.

Finally, note that we have simplified the formal semantics in a number of ways. First, we indicate that (Bel i φ) is satisfied if the $\mathcal{L}_0$ formula φ is present in i's belief set or belief rule-set. In the formal model of MyWorld that we developed earlier, the element's of an agent's belief set are defined to be VDM structures representing formulae, rather than formulae themselves. However, the meaning should be clear. Similar comments apply to the Intend connective. We have also assumed that the function *apply*, (which applies a binding to a condition), has been extended to arbitrary $\mathcal{L}_0$-formulae.

Properties: The temporal fragment of $\mathcal{L}_M$ inherits the expected properties of its underlying Temporal Logic of Reals — an axiomatization is given in [2]. In addition to these, there are a number of axioms relating to the agent part of $\mathcal{L}_M$[5]. By knowing what the beliefs, intentions, and various rule-sets of an agent are, these axioms allow us to derive a set of formulae that capture many of the properties of that agent. These formulae are called the *theory* of the agent: the systematic derivation of such a theory is the first step on the road to formally verifying the properties of the agent, and the system to which it belongs. We comment briefly on specification and verification below.

First, we have an axiom which tells us that agents apply belief rules exhaustively. Suppose φ, ψ, φ', and ψ' are $\mathcal{L}_0$-formulae, and $\exists \beta \in \mathit{Binding}$ such that $\mathit{apply}(\beta, \varphi) = \varphi'$ and $\mathit{apply}(\beta, \psi) = \psi'$. Then the following axiom is sound:

$$\vdash ((\mathsf{Bel}\ i\ \varphi \Rightarrow \psi) \wedge (\mathsf{Bel}\ i\ \varphi')) \Rightarrow (\mathsf{Bel}\ i\ \psi'). \qquad (A1)$$

Axiom ($A2$) tells us about the adoption of intentions via IARs. Suppose φ, ψ, χ, φ', ψ', and χ' are $\mathcal{L}_0$-formulae, and $\exists \beta \in \mathit{Binding}$ such that $\mathit{apply}(\beta, \varphi) = \varphi'$, …, and $\mathit{apply}(\beta, \chi) = \chi'$. Then if agent i has an IAR with adoption condition φ, goal ψ, motivation χ, and rating n, then ($A2$) will correctly describe i:

$$\vdash (\mathsf{Bel}\ i\ \varphi') \Rightarrow (\mathsf{Intend}\ i\ \psi'\ \chi'\ n). \qquad (A2)$$

[5] We shall not present formal proofs of these axioms, as they are all immediate from the model given in the preceding section.

Note that it is much simpler to actually use axioms ($A1$) and ($A2$) than it is to state them formally!

There are a number of other axioms that capture properties of intentions. Axiom ($A3$) tells us that once an agent has adopted an intention, it keeps it until either it is believed to be satisfied, or its motivation is no longer believed to be present.

$$\vdash (\mathsf{Intend}\ i\ \varphi\ \psi\ n) \Rightarrow \left[\begin{array}{c} (\mathsf{Intend}\ i\ \varphi\ \psi\ n) \\ \mathcal{U} \\ (\mathsf{Bel}\ i\ \varphi) \vee \neg (\mathsf{Bel}\ i\ \psi) \end{array}\right] \tag{$A3$}$$

Axioms ($A4$) and ($A5$) tell us that an agent's intentions are consistent with it's beliefs.

$$\vdash (\mathsf{Intend}\ i\ \varphi\ \psi\ n) \Rightarrow \neg (\mathsf{Bel}\ i\ \varphi) \tag{$A4$}$$

$$\vdash (\mathsf{Intend}\ i\ \varphi\ \psi\ n) \Rightarrow (\mathsf{Bel}\ i\ \psi) \tag{$A5$}$$

It is interesting to compare ($A3$)–($A5$) with those axioms considered by researchers developing abstract theories of intention [3, 18]. For example, ($A3$) captures the important properties of *persistent relativised goals*, as defined by Cohen-Levesque [3, pp254–255]; the only significant property they lack is that according to the Cohen-Levesque theory, an agent will drop a persistent relativised goal if it believes that it can never be achieved. MyWorld agents are not (yet!) capable of such reasoning.

Reasoning about MyWorld **systems:** The issues surrounding the use of logics like $\mathcal{L}_M$ to reason about DAI systems are considered at length in [23], and more briefly in [6, 25]; they will not, therefore, be discussed again here. Instead, we simply note that $\mathcal{L}_M$ can be used in both the specification and verification of MyWorld systems. A specification S is given as a set of $\mathcal{L}_M$-formulae; any system whose execution histories all satisfy the formulae is considered to satisfy the specification. For verification, one attempts to derive the theory T of a particular system, using axioms like ($A1$)–($A5$), above, and show via formal proof that the specification follows from the theory, i.e., that $T \vdash S$. Although specification is a straightforward process, verification is generally considered much more difficult; it is particularly awkward when one uses complex logics like $\mathcal{L}_M$, for which automated theorem proving tools are not likely to become available in the near future.

5 Concluding Remarks

In this article, we hope to have demonstrated two points: (i) that it is both possible and desirable to develop rigorous formal models of implemented multi-agent systems; and (ii) that it is possible to use such formal models in the development of more abstract formalisms for reasoning about implemented multi-agent systems. In future, we hope to extend the work presented in this article in a number of ways. First, MyWorld agents, as described in this article, have a very simple structure — we are currently reimplementing the system to provide a more powerful agent language. The formal model will need to be redeveloped when this work is complete. Secondly, the axiomatization given in section 4 is not *complete*, in that there are properties of MyWorld systems that we cannot prove using it. In particular, the relationship between intention and action needs further study.

References

1. J. F. Allen. Towards a general theory of action and time. *Artificial Intelligence*, 23(2):123–154, 1984.
2. H. Barringer, R. Kuiper, and A. Pnueli. A really abstract concurrent model and its temporal logic. In *Proceedings of the Thirteenth ACM Symposium on the Principles of Programming Languages*, pages 173–183, 1986.
3. P. R. Cohen and H. J. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42:213–261, 1990.
4. I. Craig. *Formal Specification of Advanced AI Architectures*. Ellis Horwood: Chichester, England, 1991.
5. M. Fisher. Towards a semantics for Concurrent METATEM. In M. Fisher and R. Owens, editors, *Executable Modal and Temporal Logics*. Springer-Verlag: Heidelberg, Germany, 1995.
6. M. Fisher and M. Wooldridge. Specifying and verifying distributed intelligent systems. In M. Filgueiras and L. Damas, editors, *Progress in Artificial Intelligence — Sixth Portuguese Conference on Artificial Intelligence (LNAI Volume 727)*, pages 13–28. Springer-Verlag: Heidelberg, Germany, October 1993.
7. N. Francez. *Fairness*. Springer-Verlag: Heidelberg, Germany, 1986.
8. P. Gärdenfors. *Knowledge in Flux*. The MIT Press: Cambridge, MA, 1988.
9. M. R. Genesereth and N. Nilsson. *Logical Foundations of Artificial Intelligence*. Morgan Kaufmann Publishers: San Mateo, CA, 1987.
10. M. P. Georgeff and A. L. Lansky. Reactive reasoning and planning. In *Proceedings of the Sixth National Conference on Artificial Intelligence (AAAI-87)*, pages 677–682, Seattle, WA, 1987.
11. R. Goodwin. Formalizing properties of agents. Technical Report CMU–CS–93–159, School of Computer Science, Carnegie-Mellon University, Pittsburgh, PA, May 1993.
12. J. Y. Halpern and Y. Moses. A guide to completeness and complexity for modal logics of knowledge and belief. *Artificial Intelligence*, 54:319–379, 1992.
13. D. Harel. Dynamic logic. In D. Gabbay and F. Guenther, editors, *Handbook of Philosophical Logic Volume II — Extensions of Classical Logic*, pages 497–604. D. Reidel Publishing Company: Dordrecht, The Netherlands, 1984. (Synthese library Volume 164).
14. C. B. Jones. *Systematic Software Development using VDM (second edition)*. Prentice Hall, 1990.
15. K. Konolige. *A Deduction Model of Belief*. Pitman Publishing: London and Morgan Kaufmann: San Mateo, CA, 1986.
16. Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems*. Springer-Verlag: Heidelberg, Germany, 1992.
17. B. G. Milnes. A specification of the Soar cognitive architecture in Z. Technical Report CMU–CS–92–169, School of Computer Science, Carnegie-Mellon University, Pittsburgh, PA, August 1992.
18. A. S. Rao and M. P. Georgeff. Modeling rational agents within a BDI-architecture. In R. Fikes and E. Sandewall, editors, *Proceedings of Knowledge Representation and Reasoning (KR&R-91)*, pages 473–484. Morgan Kaufmann Publishers: San Mateo, CA, April 1991.
19. A. S. Rao and M. P. Georgeff. An abstract architecture for rational agents. In C. Rich, W. Swartout, and B. Nebel, editors, *Proceedings of Knowledge Representation and Reasoning (KR&R-92)*, pages 439–449, 1992.

20. S. Rosenschein and L. P. Kaelbling. The synthesis of digital machines with provable epistemic properties. In J. Y. Halpern, editor, *Proceedings of the 1986 Conference on Theoretical Aspects of Reasoning About Knowledge*, pages 83–98. Morgan Kaufmann Publishers: San Mateo, CA, 1986.
21. N. Seel. *Agent Theories and Architectures*. PhD thesis, Surrey University, Guildford, UK, 1989.
22. Y. Shoham. Agent-oriented programming. *Artificial Intelligence*, 60(1):51–92, 1993.
23. M. Wooldridge. *The Logical Modelling of Computational Multi-Agent Systems*. PhD thesis, Department of Computation, UMIST, Manchester, UK, October 1992. (Also available as Technical Report MMU–DOC–94–01, Department of Computing, Manchester Metropolitan University, Chester St., Manchester, UK).
24. M. Wooldridge. Coherent social action. In *Proceedings of the Eleventh European Conference on Artificial Intelligence (ECAI-94)*, pages 279–283, Amsterdam, The Netherlands, 1994.
25. M. Wooldridge. Temporal belief logics for modelling distributed artificial intelligence systems. In G. M. P. O'Hare and N. R. Jennings, editors, *Foundations of Distributed Artificial Intelligence*. John Wiley & Sons: Chichester, England, 1995. (To appear).
26. M. Wooldridge and D. Vandekerckhove. MYWORLD: An agent-oriented testbed for distributed artificial intelligence. In S. M. Deen, editor, *Proceedings of the 1993 Workshop on Cooperating Knowledge Based Systems (CKBS-93)*, pages 263–274. DAKE Centre, University of Keele, UK, 1994.