Model Checking Multi-Agent Systems

Mustapha Bourahla
Computer Science Department, University of Biskra, Algeria
mbourahla@hotmail.com

Mohamed Benmohamed
Computer Science Department, University of Constantine, Algeria
ibnm@yahoo.fr

Keywords: Multi-Agent Systems, Multi-Modal Branching-Time Logic, Model Checking

Received: September 21, 2004

Multi-agent systems are increasingly complex, and the problem of their verification and validation is acquiring increasing importance. In this paper we show how a well known and effective verification technique, model checking, can be generalized to deal with multi-agent systems. This paper explores a particular type of multi-agent system, in which each agent is viewed as having the three mental attitudes of belief (B), desire (D), and intention (I). We present a new approach to the verification of multi-agent systems, based on the use of possible-worlds framework to describe the system, a multi-modal branching-time logic BDI_{CTL}*, with a semantics that is grounded in traditional decision theory, to specify the properties, and a decision procedure based on model checking technique. An imperative multi-agent programming language and a formal semantics for this language in terms of the* BDI_{CTL} *logic are used to specify multi-agent systems. The multi-agent program is used to systemically construct the agents state spaces. Then an automatic synthesis of these state spaces using the agents mental attitudes will generate the possible worlds structures. These possible worlds will be used by the adopted decision procedure to solve the problems of verification. A preliminary implementation of the approach shows promising results.*

Povzetek: Predstavljen je nov algoritem za preverjanje pravilnosti multi-agentnih sistemov.

1 Introduction

The design of (in particular safety-critical control) systems that are required to perform high-level management and control tasks in complex dynamic environments is becoming of increasing commercial importance. Such systems include the management and control of air traffic systems, telecommunications networks, business processes, space vehicles, and medical services. Experience in applying conventional software techniques to develop such systems has shown that they are very difficult and very expensive to build, verify, and maintain. Agent-oriented systems, based on a radically different view of computational entities, offer prospects for a qualitative change in this position.

A number of different approaches have emerged as candidates for the study of agent-oriented systems [3, 6, 16, 18, 19]. One such architecture [16] views the system as a rational agent having certain mental attitudes of Belief (beliefs can be viewed as the informative component of system state), Desire (desires can be thought of as representing the motivational state of the system), and Intention (the intentions of the system capture the deliberative component of the system). Thus BDI (Belief, Desire and Intention) represents the information, motivational, and deliberative states of the agent. These mental attitudes determine the system's behavior and are critical for achieving adequate or optimal performance when deliberation is subject to resource.

To describe the belief, desire, and intention components of the system state a propositional form is used, based on possible worlds. Thus, the possible worlds model [16] consists of a set of possible worlds where each possible world is a tree structure. A particular index within a possible world is called a situation. With each situation we associate a set of belief-accessible worlds, desire-accessible worlds, and intention-accessible worlds; intuitively, those worlds that the agent believes to be possible, desires and intends to bring about, respectively.

In this paper, we address the problem of verification for such formalisms which is increasingly important. The formalism of multi-agent temporal logic [16] is introduced towards lifting one of the most successful verification techniques, model checking [4], for the validation of multi-agent systems. Multi-agent temporal logic BDI_{CTL} combines, within a single framework, the aspects of temporal logic, used to reason about the temporal evolution of finite-state automata, with agent-related aspects such as belief, desire and intention.

The problem of extending the standard temporal logic model checking techniques, and then using the related tools, to deal with the multi-agent aspects of the logic, is the specification of the possible worlds and the relation between them. The essential of our contribution is to present an approach by which we help reducing the specification time. This approach is based on the automatic synthesis of the mental attitudes of agents. Each mental state will be an index to a new created world using the specifications of the different agents. For illustrating our approach, we designed a sub-language for specifying multi-agent systems. The specification will be agent-oriented. A tool is developed for constructing the state space of each agent in the multi-agent system. Then an algorithm is developed for synthesizing the agent models of the specified multi-agent system. The synthesis result is a possible worlds model. At the end, we have adopted the standard model checking for the analysis of these models of multi-agent systems. A symbolic model checking tool for verifying multi-agent systems has been implemented. The preliminary results are extremely promising.

This paper is structured as follows. In Section 2 we describe the multi-agent temporal logic (BDI_{CTL}). In Section 3, we present the specification sub-language and its underlying intuitions, and define the language and the semantics as a temporal logic. In Section 4, we present the algorithm for synthesizing the corresponding multi-agent structures. In Section 5, we present the extended general algorithm for model checking. Finally, in Section 6 we outline the results, discuss future work, and draw some conclusions.

2 Multi-Agent Temporal Logic BDI_{CTL}

The temporal logic BDI_{CTL} [16] we consider is extension of Computation Tree Logic CTL [7] that has been used extensively for reasoning about concurrent programs. The branching-time logic CTL is extended to represent the mental state or belief-desire-intention state of an agent. This logic can then be used to reason about agents and the way in which their beliefs, desires, and actions can bring about the satisfaction of their desires. The syntax of BDI_{CTL} is as follows.

$$\varphi ::= true \mid p \mid \neg\varphi \mid \varphi \vee \varphi \mid \exists X\varphi \mid \exists G\varphi \mid \exists \varphi U \varphi \mid B_i\varphi \mid D_i\varphi \mid I_i\varphi.$$

The primitives of this language include a nonempty set AP of atomic propositions, propositional connectives $\vee$ and $\neg$, modal operators B (agent believes), D (agent desires), and I (agent intends), and temporal operators of CTL. The CTL temporal operators are $\exists X\varphi$ (φ might hold at next time instant), $\exists\varphi U\psi$ (it might be the case that ψ holds at a certain time future and until then φ holds), and $\exists G\varphi$ (φ might hold for all future time instants). Temporal operators are compactly characterized by $\exists\varphi U\psi \Leftrightarrow (\psi \vee (\varphi \wedge \exists X \exists(\varphi U \psi)))$ and by $\exists G\varphi \Leftrightarrow (\varphi \wedge \exists X \exists G\varphi)$. We have operators $B_i\varphi$, $D_i\varphi$, and $I_i\varphi$ which mean that agent i has a belief, desire, and intention of φ, respectively. This grammar is not given in its most succinct form and there exist equivalence rules to express the same formula with different operators; for example, $\forall F\varphi$ (φ is inevitable) is equivalent to $\neg\exists G\neg\varphi$. In practice, by using these equivalence rules, a formula can be written such that the negation appears only at the level of atomic propositions. Such a form of a formula is known as Negative Normal Form (henceforth NNF form).

The traditional possible-worlds semantics of beliefs considers each world to be a collection of propositions and models belief by a belief-accessibility relation $\mathcal{B}$ linking these worlds. A formula is said to be believed in a world if and only if it is true in all its belief-accessible worlds [10]. The accessibility relation $\mathcal{B}$ is a relation between the world at an index and at a time point to a set of worlds. Intuitively, an agent believes a formula in a world at a particular index if and only if in all its belief-accessible worlds the formula is true. We consider each possible world to be a tree structure with a single past and a branching future [5]. Evaluation of formulas is with respect to a world and a state. Hence, a state acts as an index into a particular tree structure or world of the agent. The belief-accessibility relation maps a possible world at a state to other possible worlds. The desire-, and intention-accessibility relations behave in a similar fashion. More formally, we have the following definition of a Kripke structure.

Definition 1 *A Kripke structure is defined to be a tuple* $K = \langle W, S, \{S_w : w \in W\}, \{R_w : w \in W\}, \{I_w : w \in W\}, L, \mathcal{B}, \mathcal{D}, \mathcal{I}\rangle$, *where* W *is a set of possible worlds,* S *is the set of states,* S_w *is the set of states in each world* $w \in W$ ($S = \cup_{w\in W} S_w$), R_w *is a total tree relation, i.e.,* $R_w \subseteq S_w \times S_w$, I_w *a set of initial states* ($I_w \subseteq S_w$), $L : W \times S \to 2^{AP}$ *is a function that labels for each world* $w \in W$, *each state* $s \in S_w$ *with the set of atomic propositions true in that state, and* $\mathcal{B}$, $\mathcal{D}$, *and* $\mathcal{I}$ *are relations on the worlds* W *and states* S *(i.e.* $\mathcal{O} \subseteq W \times S \times W$*), where* $\mathcal{O}$ *is one of* $\mathcal{B}$, $\mathcal{D}$, *or* $\mathcal{I}$.

We also define a world to be a sub-world of another if one of them contains fewer paths, but they are otherwise identical to each other. More formally, we have the following definition.

Definition 2 *A world* w' *is a sub-world of the world* w, *denoted by* $w' \sqsubseteq w$, *if and only if*

1. $S_{w'} \subseteq S_w$, $I_{w'} \subseteq I_w$, $R_{w'} \subseteq R_w$,
2. $\forall s \in S_{w'}$, $L(w', s) = L(w, s)$,
3. $\forall s \in S_{w'}$, $(w', s, v) \in \mathcal{B}$ *iff* $(w, s, v) \in \mathcal{B}$; *and similarly for* $\mathcal{D}$ *and* $\mathcal{I}$.

The semantics of BDI_{CTL} involves two dimensions: an epistemic and a temporal dimension. The truth of a formula depends on both the epistemic world w and the temporal state s. A pair (w, s) (denoted also s^w) is called a situation in which BDI_{CTL} formulas are evaluated. The relation between situations is traditionally called an accessibility relation (for beliefs) or a successor relation (for time).

A BDI_{CTL}-model $\mathcal{M}$ is represented as a Kripke structure. We note a model $\mathcal{M}$ in world w as $\mathcal{M}_w$. A trace (path) in a world $w \in W$ starting from s^w is an infinite sequence of states $\rho_w = s_0^w s_1^w s_2^w \cdots$ such that $s_0^w = s^w$, and for every $i \geq 0$, $\langle s_i^w, s_{i+1}^w \rangle \in R_w$. The $(i+1)$-th state of trace ρ_w is denoted $\rho_w[i]$. The set of paths starting in state s^w of the model $\mathcal{M}_w$ is defined by $\Pi_{\mathcal{M}_w}(s^w) = \{\rho_w \mid \rho_w[0] = s^w\}$.

For any BDI_{CTL}-model $\mathcal{M}_w$ and state $s^w \in S_w$, there is an infinite computation tree with root labeled s^w such that $\langle s_i^w, s_j^w \rangle$ is an arc in the tree if and only if $\langle s_i^w, s_j^w \rangle \in R_w$. Satisfaction of formulas, denoted by $\models_{\mathcal{M}_w}$, is given with respect to a model $\mathcal{M}$, a world w, and state s. The expression $s \models_{\mathcal{M}_w} \varphi$ is read as "model $\mathcal{M}$ in world w and state s satisfies φ".

- $s \models_{\mathcal{M}_w} p$ iff $p \in L(w, s)$
- $s \models_{\mathcal{M}_w} \neg p$ iff $s \not\models_{\mathcal{M}_w} p$
- $s \models_{\mathcal{M}_w} \varphi \vee \psi$ iff $s \models_{\mathcal{M}_w} \varphi$ or $s \models_{\mathcal{M}_w} \psi$
- $s \models_{\mathcal{M}_w} \exists X \varphi$ iff $\exists \rho_w \in \Pi_{\mathcal{M}_w}(s) : \rho_w[1] \models_{\mathcal{M}_w} \varphi$
- $s \models_{\mathcal{M}_w} \exists G \varphi$ iff $\exists \rho_w \in \Pi_{\mathcal{M}_w}(s) : \forall j \geq 0 : \rho_w[j] \models_{\mathcal{M}_w} \varphi$
- $s \models_{\mathcal{M}_w} \exists \varphi U \psi$ iff $\exists \rho_w \in \Pi_{\mathcal{M}_w}(s) : (\exists j \geq 0 : \rho_w[j] \models_{\mathcal{M}_w} \psi) \wedge (\forall k, 0 \leq k < j : \rho_w[k] \models_{\mathcal{M}_w} \varphi)$
- $s \models_{\mathcal{M}_w} B_i(\varphi)$ iff $\forall v, (w, s, v) \in \mathcal{B} : \forall s' \in \mathcal{B}(B_i(\varphi)) : s' \models_{\mathcal{M}_v} \varphi$
- $s \models_{\mathcal{M}_w} D_i(\varphi)$ iff $\forall v, (w, s, v) \in \mathcal{D} : \forall s' \in \mathcal{D}(D_i(\varphi)) : s' \models_{\mathcal{M}_v} \varphi$
- $s \models_{\mathcal{M}_w} I_i(\varphi)$ iff $\forall v, (w, s, v) \in \mathcal{I} : \forall s' \in \mathcal{I}(I_i(\varphi)) : s' \models_{\mathcal{M}_v} \varphi$

We denote the set of states in the world v that are accessible to the state s in the world w, where $(w, s, v) \in \mathcal{O}$ by $\mathcal{O}(O_i(\varphi))$ (more details are in Section 4). A formula φ is said to be valid in $\mathcal{M}_v$, written as $\models_{\mathcal{M}_v} \varphi$, if $s \models_{\mathcal{M}_v} \varphi$ for every state $s \in S_v$. A formula is valid if it is true in every state, in every world, in every structure (model).

Recall that an agent i has a belief φ, denoted $B_i(\varphi)$, in state s if and only if φ is true in all the belief-accessible worlds of the agent at state s. As the belief-accessibility relation is dependent on the state, the mapping of B at some other state may be different. Thus the agent can change its beliefs about the options available to it. Similar to belief-accessible worlds, for each state we also associate a set of desire-accessible worlds to represent the desires of the agent. Thus, in the same way that we treat belief, we say that the agent has a desire φ in state s if and only if φ is true in all the desire-accessible worlds of the agent in state s.

In the philosophical literature, desires can be inconsistent and the agent need not know the means of achieving these desires. Desires have the tendency to 'tug' the agent in different directions. They are inputs to the agent's deliberation process, which results in the agent choosing a subset of desires that are both consistent and achievable. In the AI literature such consistent achievable desires are usually called goals. The desires as presented here are logically consistent, but due to the branching-time structure, conflicting desires can 'tug' the agent along different execution paths. That is, while the desires may be logically consistent, they may not all be realizable, as the agent can only follow one execution path in the branching tree of possible executions. The deliberation process must eventually resolve these conflicts and choose a set of realizable desires before the agent can act intentionally.

Intentions are similarly represented by sets of intention-accessible worlds. These worlds are ones that the agent has chosen to attempt to realize. The intention-accessibility relation is used to map the agent's current world and state to all its intention-accessible worlds. We say that the agent intends a formula in a certain state if and only if it is true in all the agent's intention-accessible worlds at that state.

3 Specification of Multi-Agent Systems

A multi-agent system contains a finite number of agents. The basic form of an agent is "agent A is init P", where A is the name of the agent and P is the program body. Each agent in a multi-agent system is assumed to have a unique name, drawn from a set of $agent\ identifiers$. The main part of an agent, which determines its behavior, is the program body P. The basis of program bodies is a simple imperative language, containing iteration ($loop$ loops), sequence (the ; constructor), selection (a form of the if, $then$, $else$ statement), choice (the | constructor), and assignment statements.

An agent A is allowed to execute by a do instruction any of a set $Actions = \{\alpha, \cdots\}$ of $external\ actions$. The simplest way to think of external actions is as $native\ methods$ in a programming language like Java. They provide a way for agents to execute actions that do not simply affect the agent's internal state, but its external environment. The basic form of the do instruction is $do\ \alpha$, where $\alpha \in Actions$ is the external action to be performed. When we incorporate communication, we do so by modeling message sending as an external action to be performed.

In a conventional programming language, conditions in if statement are only allowed to be dependent on program variables. Unusually, we allow conditions in if statement to be arbitrary formulas of the BDI_{CTL} logic (any acceptable formula is allowed as a condition). To make this more concrete, consider the following:

$$if\ B_j p\ then\ r := p\ else\ r := false$$

The idea is that if the agent executing this instruction believes that agent j believes that p, then the agent executing the instruction assigns the value of p to r. If the agent executing the instruction believes it is not the case that agent j believes p, then it assigns the value $false$ to r. Notice the form of words used here: the agent executing this if instruction must believe that j believes p; the condition does not depend on what j actually believes, but on what the agent executing the statement believes that j believes. As this example illustrates, conditions can thus refer to the mental state of other agents. The general form of a $loop$ construct, as in conventional programming languages, is $loop\ P\ endloop$, where P is a program.

Given a collection $\{A_1, \cdots, A_n\}$ of agents, they are composed into a multi-agent system by the parallel composition operator "$\|$": $A_1 \parallel \cdots \parallel A_n$. Formally, the abstract syntax of multi-agent systems is defined by the grammar below.

MAS ::= $Agent \parallel \cdots \parallel Agent$
Agent ::= agent A is Init P
Init ::= init $p := true\ or\ false$, where $p \in AP$
P ::= $do\ \alpha \mid p\ :=\ true\ or\ false$
$\quad \mid if\ \varphi\ then\ P \mid if\ \varphi\ then\ P\ else\ P$
$\quad \mid loop\ P\ endloop \mid P\ ';'\ P \mid P\ '|'\ P$

Example 1 *To clarify this syntax, let us consider the following scenario involving two agents: a receiver rcv and a sender snd. snd continuously reads news on a certain subject (p) from its sensors (e.g., the standard input). Once read the news, snd informs rcv only if it believes that rcv does not have the correct knowledge about that subject (this in order to minimize the traffic over the network). Once received the news, rcv acknowledges this fact back to snd. After the reception of acknowledgement from the agent rcv, the agent snd will believe that the agent rcv believes $p \vee q$ (q is a propositional atom) or it believes that the agent rcv believes p (or q) in the case that the agent rcv at the beginning, does not have the correct knowledge about p (or $\neg p$).*

We have therefore three agents: snd, rcv, and a network (communication protocol) $protocol$ which allows them to interact. The descriptions of snd, rcv and the communication protocol $protocol$, are given below respectively.

agent snd is
init $\forall p \in AP : p\ :=\ false$
loop
do read(p);
if $p \wedge \neg B_{rcv} p$ *then*
do putmsg(inform(snd, rcv, p*));*
do getmsg(inform(rcv, snd, $B_{rcv} p$*));*
$(B_{rcv} p := true) \mid (B_{rcv}(p \vee q) := true)$;
else if $\neg p \wedge \neg B_{rcv} \neg p$ *then*
do putmsg(inform(snd, rcv, $\neg p$*));*
do getmsg(inform(rcv, snd, $B_{rcv} \neg p$*));*
$(B_{rcv} q := true) \mid (B_{rcv}(p \vee q) := true)$;
endloop

agent rcv is
init $\forall p \in AP : p\ :=\ false$
loop
{
do getmsg(inform(snd, rcv, p*));*
$p := true$;
do putmsg(inform(rcv, snd, $B_{rcv} p$*));*
} | {
do getmsg(inform(snd, rcv, $\neg p$*));*
$q := true$;
do putmsg(inform(rcv, snd, $B_{rcv} \neg p$*));*
}
endloop

agent protocol is
init $\forall p \in AP : p\ :=\ false$
loop
$\forall p \in AP : p\ :=\ false$;
{
$B_{snd} \forall F$ *do putmsg(inform(snd, rcv, p)) :=* $true$;
$B_{rcv} \forall F$ *do getmsg(inform(snd, rcv, p)) :=* $true$
} | {
$B_{snd} \forall F$ *do putmsg(inform(snd, rcv,* $\neg p$*)) :=* $true$;
$B_{rcv} \forall F$ *do getmsg(inform(snd, rcv,* $\neg p$*)) :=* $true$
};
$\forall p \in AP : p\ :=\ false$;
{
$B_{rcv} \forall F$ *do putmsg(inform(rcv, snd,* $B_{rcv} p$*)) :=* $true$;
$B_{snd} \forall F$ *do getmsg(inform(rcv, snd,* $B_{rcv} p$*)) :=* $true$
} | {
$B_{rcv} \forall F$ *do putmsg(inform(rcv, snd,* $B_{rcv} \neg p$*)) :=* $true$;
$B_{snd} \forall F$ *do getmsg(inform(rcv, snd,* $B_{rcv} \neg p$*)) :=* $true$
}
endloop

$mas = protocol \parallel snd \parallel rcv$

In these descriptions, the news subject of the information exchange is the truth value of the propositional atom p. $inform(snd, rcv, p)$ returns a message with sender snd, receiver rcv, and content p ($inform$ is a FIPA (Foundation for Intelligent Physical Agents) primitive). $putmsg$ and $getmsg$ are the primitives for putting and getting (from the communication channel) a message. $read$ allows for reading from the standard input. B_{rcv} is the operator used to represent the beliefs of rcv as perceived by the other agents, and dually for B_{snd}. Notice that the communication protocol has beliefs about rcv and snd and therefore must have a representation of how they behave. We suppose that this representation coincides with what rcv and snd actually are, as described above. This allows us to model the fact that the communication protocol behaves correctly following what snd and rcv do. snd also has beliefs about rcv. We suppose that snd (which in principle does not know anything about how rcv works) only knows that rcv can be in one of two states, with p being either $true$ or $false$. In the example, $B_{snd} \forall F\ do < statement >$ (or $B_{rcv} \forall F\ do < statement >$) intuitively means that snd(rcv) will necessarily reach a state in which it will have just performed the action corresponding to $< statement >$. The agent program $protocol$ codifies the fact that the protocol implements the information flow between snd and rcv, and the fact that it always delivers the messages it is asked to deliver. Some properties that we may want to prove are:

1. *An agent liveness property, e.g., that snd will eventually believe that rcv believes p or believes $\neg p$. Its expression is $\models_{\mathcal{M}_{w_{snd}}} \forall F(B_{rcv} p \vee B_{rcv} \neg p)$. Where w_{snd} is the world seen by the agent snd.*

2. *An overall system liveness property, e.g., that if it believes p, then in the future snd will believe that rcv*

will believe p. Its expression is $\models_{\mathcal{M}} B_{snd}(p) \supset \forall F B_{snd} \forall F B_{rcv} p$.

3.1 Formal Semantics

The semantics of a multi-agent program will be defined as a formula of BDI_{CTL}, which characterizes the acceptable computations of the system, and the "mental state" of the agents in the system.

The agent program semantic function is defined in terms of the function $[\![\cdots]\!]_{Bexp} : Bexp \rightarrow B$, which gives the semantics of Boolean expressions. The four remaining semantic functions are defined in Figure 1. The idea is that the semantics are defined inductively by a set of definitions, one for each construct in the language.

A declaration "$agent\ A\ is\ init\ P$" binds a name A with the semantics of the $init$ statements and the program body P. We capture the semantics of this by systematically substituting name A for the place-holder name $self$ in $[\![init]\!]_{Init} \wedge [\![P]\!]_P$. The semantics of a system $A_1 \parallel \cdots \parallel A_n$ is simply the conjunction of the semantics of the component agents A_i, together with some background assumptions ψ_{MAS}. The idea of the background assumptions is that these capture general properties of a multi-agent system that are not captured by the semantics of the language.

4 Construction of Possible Worlds Model

We will develop an algorithm to construct a multi-agent structure as defined in Definition 1. First we need to build a structure for each agent specification then we will synthesize these structures. At the beginning, a multi-agent system will have a Kripke structure of the form $K = \langle W = \{w_1, \cdots, w_n\}, S = \{S_{w_1}, \cdots, S_{w_n}\}, R = \{R_{w_1}, \cdots, R_{w_n}\}, I = \{I_{w_1}, \cdots, I_{w_n}\}, L, \mathcal{B} = \emptyset, \mathcal{D} = \emptyset, \mathcal{I} = \emptyset\rangle$, where n is the number of agents. Then we will compute the sets $\mathcal{B}$, $\mathcal{D}$, and $\mathcal{I}$ using the worlds $w \in W$ and the labeling function L. At the end, a Kripke structure K will be constructed representing the multi-agent system using the algorithm below. The initial Kripke structure K is generated directly from the agents specifications. In each world, there is a finite set of the BDI operators of the form $O_i\varphi$ (where O stands for B, D, or I). This set is considered as a part of the atomic propositions AP.

Let us call $TrueBDI_{(w,v)}(s)$ the set of BDI atoms of world w (of the current agent), of the form $O_i\varphi$, which are true at s ($TrueBDI_{(w,v)}(s) = BDI_{(w,v)} \cap L(w,s)$). v is the world of the agent i. An accessibility relation $\mathcal{O}_{(w,v)} \subseteq BDI_{(w,v)} \times S_v$ (or $\mathcal{O}_{(w,v)}(O_i\varphi) \subseteq S_v$), constraints the truth of BDI ($O_i\varphi$) atoms of a world w to the truth values (of φ) in the world v. The states of world v accessible to s are those states belonging to the intersection, over the BDI atoms true at s, of the sets of states accessible to $TrueBDI_{(w,v)}(s)$. We extend the accessiblity relation to a relation over a set of BDI atoms $\mathcal{A} \subseteq BDI_{(w,v)}$ as follows.

$$\mathcal{O}_{(w,v)}(\mathcal{A}) = \bigcap_{O_i\varphi \in \mathcal{A}} \mathcal{O}_{(w,v)}(O_i\varphi)$$

Therefore, the set of states of v accessible to a state s of w will be simply denoted by $\mathcal{O}_{(w,v)}(TrueBDI_{(w,v)}(s))$.

Depending on the kind of BDI operator being considered, the accessibility relation may have different properties. What makes $\mathcal{M}$ a model of a multi-agent possible world is the particular structure of the accessibility relations among adjacent sub-worlds.

Definition 3 *A BDI_{CTL} model $\mathcal{M}$ is a possible world structure if for every word w, every BDI atom $O_i\varphi$ of w and every $s \in S_w$ the following conditions hold.*

1. *If $O_i\varphi \in L(w,s)$, then $s' \in \mathcal{O}_{(w,v)}(TrueBDI_{(w,v)}(s))$ implies that s' is reachable in v and $s' \models_{\mathcal{M}_v} \varphi$.*
2. *If $O_i\varphi \notin L(w,s)$, then for some reachable state $s' \in \mathcal{O}_{(w,v)}(TrueBDI_{(w,v)}(s))$, $s' \models_{\mathcal{M}_v} \neg\varphi$.*

Condition 1 tells us what are the states in world v which are accessible to a given state s (satisfying $TrueBDI_{(w,v)}(s)$), according to the semantics of $BDIs$, namely that the argument of a BDI true at a state must be true in all the states reachable from it via accessibility relation. Condition 2, on the other hand, tells us what are the states of world w which actually comply to the semantics of $BDIs$, i.e. the states which assign truth values to BDI atoms in accordance with the semantics of the BDI operator.

Let φ and ψ be two BDI_{CTL} formulas, assume that $\varphi \supset \psi$, it would be unreasonable to allow for a state satisfying the BDI atom $O_i\varphi$, yet not satisfying the BDI atom $O_i\psi$ at the same time. This is the kind of situation that this condition prevents. Indeed, let us suppose there is a state s of a world w satisfying $O_i\varphi$. By Condition 1 of Definition 3, any reachable state s' of world v accessible to s must satisfy φ. By Condition 2 of Definition 3, for s not to satisfy the BDI atom $O_i\psi$, there must be a reachable state s'' in world v accessible to s and which does not satisfy ψ. But, according to Condition 1, all the states accessible to s must satisfy φ and, consequently, ψ as well, which is impossible.

On the other hand, the definition of multi-agent structure allows for a state s of a world w to satisfy both BDI atoms $O_i\varphi$ and $O_i\neg\varphi$, where φ is BDI_{CTL} formula. This happens when there is no state in world v is accessible to s (i.e. when $\mathcal{O}_{(w,v)}(TrueBDI_{(w,v)}(s))$ is empty). This corresponds to the situation where world w, when in state s, ascribes inconsistent $BDIs$ to world v. Notice however that this kind of inconsistency is of a different nature from the one ruled out by Definition 3. Indeed, allowing a state s not to satisfy $O_i\psi$ while satisfying $O_i\varphi$ (where $\varphi \supset \psi$) would make the specification of w itself inconsistent, while allowing both $O_i\varphi$ and $O_i\neg\varphi$ would not. It

$$
\begin{aligned}
[\![init\ p]\!]_{Init} &= B_{self}p, p \in AP \\
[\![do\ \alpha]\!]_P &= I_{self}\alpha, \alpha \in Actions \\
[\![p := e]\!]_P &= \forall X B_{self}[\![e]\!]_{Bexp} \\
[\![if\ \varphi\ then\ P]\!]_P &= B_{self}\varphi \Rightarrow [\![P]\!]_P \\
[\![if\ \varphi\ then\ P_1\ else\ P_2]\!]_P &= B_{self}\varphi \Rightarrow [\![P_1]\!]_P \wedge \neg B_{self}\varphi \Rightarrow [\![P_2]\!]_P \\
[\![loop\ P\ endloop]\!]_P &= [\![P\ ;\ loop\ P\ endloop]\!]_P \\
[\![P_1; P_2]\!]_P &= [\![P_1]\!]_P \Rightarrow [\![P_2]\!]_P \\
[\![P_1 \mid P_2]\!]_P &= [\![P_1]\!]_P \vee [\![P_2]\!]_P \\
[\![agent\ A\ is\ init\ P]\!]_{Agent} &= ([\![init]\!]_{Init} \wedge [\![P]\!]_P)[A \mapsto self] \\
[\![A_1 \parallel \cdots \parallel A_n]\!]_{MAS} &= [\![A_1]\!]_{Agent} \wedge \cdots \wedge [\![A_n]\!]_{Agent} \wedge \psi_{MAS}
\end{aligned}
$$

Figure 1: Semantics of multi-agent program

is clearly possible, though, to rule out also the latter situation, by adding the additional constraint that every state s must have a non-empty set of accessible states of the world below (i.e. $\mathcal{O}_{(w,v)}(TrueBDI_{(w,v)}(s)) \neq \emptyset$).

4.1 Synthesizing Multi-Agent Structure

In this section we present a synthesis algorithm that automatically constructs the suitable multi-agent Kripke structure $\mathcal{M}$ from a set of independently generated structures for each agent specification and a selected set of BDI atoms, thus leading to significant savings in the modeling phase. The synthesis algorithm is reported below. It takes as input a set of agents represented as world structures, and a set of BDI atoms. Intuitively, the algorithm at each world computes as a first step the accessibility relations associated to each BDI operator of the world. This is done according to Condition 1 of Definition 3. The second step is to implement Condition 2 of the same definition. The idea is to check whether there are states of the current world where the negation of some BDI atoms conflicts with other BDI atoms true at that state. Condition 2 tells us no such state is admissible in a multi-agent structure as they correspond to impossible combination of BDI atoms. Therefore, we need to get rid of all those states in the structure of the world. Once those two steps are performed at each world, the resulting structure is indeed a multi-agent structure.

Algorithm 1 *BUILD-MODEL(w, $\mathcal{M}$)*

```
{
 for each i ∈ agent identifiers do
     Let v be the world structure of the agent i
     if BDI_(w,v) ≠ ∅ then
        Let wv be the world of the agent i as viewed by
        the agent of the world w
        M ← BUILD-MODEL(wv, M)
        M ← CreateAR(w, v, M)
     end if
 end for
 return(M)
}
```

The initial call is BUILD-MODEL(top, $\mathcal{M}$), where top is the root of the Kripke structure (in our example, is the $protocol$ agent). At the end of the algorithm, $\mathcal{M}$ will contain the accessibility relations of the structure rooted at w. The algorithm BUILD-MODEL recursively descends depth-first the tree of worlds rooted at w, and builds the accessibility relations (algorithm below) with all the worlds one level below the current world w. The creation of the accessibility relations is using the algorithm MAS-Sat(w, φ) (descried in the next section) which computes the set of states satisfying the formula φ in the world w.

Algorithm 2 *CreateAR(w, v, $\mathcal{M}$)*

```
{
 /* Condition 1 of Definition 3 */
 for each O_i φ ∈ BDI_(w,v) do
     [[φ]]_v ← MAS-Sat(v, φ)
     O_(w,v)(O_i φ) ← [[φ]]_v
 end for
 /* Condition 2 of Definition 3 */
 BadStates ← ∅
 for each O_i φ ∈ BDI_(w,v) do
     [[¬φ]]_v ← MAS-Sat(v, ¬φ)
     BadBDI ← {A ⊆ BDI_(w,v) \ {O_i φ} |
                  O_(w,v)(A) ∩ [[¬φ]]_v = ∅}
     BadStates ← BadStates ∪ {s ∈ S_w |
                  TrueBDI_(w,v)(s) ⊆ BadBDI}
 end for
 S'_w ← S_w \ BadStates
 if R'_w (which is R_w restricted to S'_w)
   is total tree relation then
   substitute w with ⟨S'_w, R'_w, I_w ∩ S'_w⟩ in M
 else remove w from M
 return(M)
}
```

Example 2 *In Figure 2, the Kripke structure generated from the agents specifications, contains three worlds for the agents $protocol$, snd and rcv. The initial state is marked by 0 and the list of atomic propositions true at a state are written beside the circle representing that state. The values of symbols m_1, m_2, m_3 and m_4 are $inform(snd, rcv, p)$,*

$inform(snd, rcv, \neg p)$, $inform(rcv, snd, B_{rcv}p)$ and $inform(rcv, snd, B_{rcv}\neg p)$, respectively. The first step of the synthesis is the creation of the accessibility relations. The agent $protocol$ has beliefs on the agent snd and the agent rcv thus, there are accessibility relations from its world to new created worlds for the two agents as believed by the agent $protocol$. These accessibility relations are illustrated by dotted edges. We have also accessibility relations shown by dashed edges from the world(s) representing the agent snd to new created world representing the agent rcv because the agent snd has beliefs on the agent rcv. In the second step, we have removed the states (with their edges) that are making conflicts (there are two states as colored in the world of the agent snd). Then, the resulting Kripke structure is a possible world representing the multi-agent system which can be used to check specified properties for the multi-agent system.

5 BDI_{CTL} Model Checking

In this section, we present an extension of the standard CTL model checking algorithm [4]. Given a BDI_{CTL}-formula φ and a world of BDI_{CTL}-model $\mathcal{M}_w$ with a finite set of states (S_w), the model checking algorithm MAS-Sat(w, φ) (presented below) computes the set of states from the world w satisfying the BDI_{CTL} formula φ. This set is denoted $[\![\varphi]\!]_w$, and is computed in a recursive way, i.e. by computing for each sub-formula ψ of φ the set $[\![\psi]\!]_w$. In order to decide whether $s \models_{\mathcal{M}_w} \varphi$ we just have to check whether $s \in [\![\varphi]\!]_w$.

Algorithm 3 *MAS-Sat(w, φ)*

{
case φ of
$p \mid p \in AP : [\![\varphi]\!]_w \leftarrow \{s \mid p \in L(w,s)\}$
$O_j\psi \mid O_j\psi \in AP : [\![\varphi]\!]_w \leftarrow \{s \mid O_j\psi \in L(w,s)\}$
$O_j\psi \mid O_j\psi \notin AP$: *Let v be the world of the agent j*
and let wv be the world of the agent j
as viewed by the agent of the world w
$[\![\psi]\!]_{wv} \leftarrow$ *MAS-Sat(wv, ψ)*
$\mathcal{O}^{-1}_{(w,v)}([\![\psi]\!]_{wv}) \leftarrow \{\mathcal{A} \subseteq BDI_{(w,v)} \mid \mathcal{O}_{(w,v)}(\mathcal{A}) \subseteq [\![\psi]\!]_{wv}\}$
$[\![\varphi]\!]_w \leftarrow \{s \in S_w \mid TrueBDI_{(w,v)}(s) \subseteq \mathcal{O}^{-1}_{(w,v)}([\![\psi]\!]_{wv})\}$
$\neg\psi : [\![\varphi]\!]_w \leftarrow S_w \setminus$ *MAS-Sat(w, ψ)*
$\psi \vee \gamma : [\![\varphi]\!]_w \leftarrow$ *MAS-Sat(w, ψ) $\cup$ MAS-Sat(w, γ)*
$\exists X\psi : Q \leftarrow$ *MAS-Sat(w, ψ)*
$[\![\varphi]\!]_w \leftarrow \{s \in Q \mid \exists\langle s, s'\rangle \in R_w \wedge s' \in Q\}$
$\exists G\psi : [\![\varphi]\!]_w \leftarrow \nu Z.([\![\psi]\!]_w \cap \exists X\ Z)$
$\exists(\psi U\gamma) : [\![\varphi]\!]_w \leftarrow \mu Z.([\![\psi]\!]_w \cup ([\![\gamma]\!]_w \cap \exists X\ Z))$
end case
return($[\![\varphi]\!]_w$)
}

The standard model checking algorithm is adopted to accept formulas of the form $O_j\psi$ which are not BDI atoms. To compute the set of states satisfying these formulas, first we compute the satisfaction set $[\![\psi]\!]_{wv}$ of the sub-formula ψ, then we compute the set of BDI atoms ($\mathcal{O}^{-1}_{(w,v)}([\![\psi]\!]_{wv})$) whose accessible states are sub-sets of $[\![\psi]\!]_{wv}$. At the end, the satisfaction set of $O_j\psi$ is the states whose true BDI atoms are subsets of $\mathcal{O}^{-1}_{(w,v)}([\![\psi]\!]_{wv})$.

For the last two cases $\exists G\psi$ and $\exists(\psi U\gamma)$ we calculate a fix-point. The satisfaction set of $\exists G\psi$ is the greatest fix-point ($\nu Z.([\![\psi]\!]_w \cap \exists X\ Z)$), and the satisfaction set of $\exists(\psi U\gamma)$ is the least fix-point ($\mu Z.([\![\psi]\!]_w \cup ([\![\gamma]\!]_w \cap \exists X\ Z))$).

6 Conclusion and Related Work

We have presented a new approach to the verification of multi-agent systems, based on the use of possible worlds to describe the system, modal temporal logic to specify the properties, and a decision procedure based on model checking technique. One contribution is the presentation of an imperative multi-agent programming language, and a formal semantics for this language in terms of the BDI_{CTL} logic. The multi-agent program is used to systemically construct the agents state spaces. Our main contribution is the synthesis of these state spaces using the agents mental attitudes to generate the possible worlds structures. These possible worlds will be used by the adopted decision procedure to solve the problems of verification.

The notions of possible worlds is inspired by the works in [15, 17, 16] and the works in the field of multi-language systems [8, 9]. Other related work is in [1], where a finitely nested data structure is used to model the belief-desire-intention states. The authors of [11] present an automata theoretic approach to temporal modal logic restricted to the case of single nesting of beliefs, applied to the specification of knowledge-based systems.

In [22], the authors present the MABLE language for the specification of multi-agent systems. In this work, modalities are translated into nested data structures in the spirit of [1]. The author of [2] use a modified version of the AgentSpeak(L) language [14] to specify agents and to exploit existing model checkers. Both the works of [22] and [2] translate the specification into a SPIN specification to perform the verification. Effectively, the attitudes for the agents are reduced to predicates, and the verification involves only the temporal verification of those. In [13] a tool is provided to translate an interpreted system into SMV code, but the verification is limited to static epistemic properties, i.e. the temporal dimension is not present, and the approach is not fully symbolic. The work of [12] is concerned with verification of interpreted systems for model checking knowledge and time based on OBDD's.

Currently we are investigating the extension in many directions. One is the extension of the language to support the other types of expression. In particular the arithmetic expressions, by incorporating a tool for abstracting the program using the framework of predicate abstractions. Another problem which is taking our attention is the explosion problem, where techniques like the equivalence based reduction or space partition can be investigated. One of the

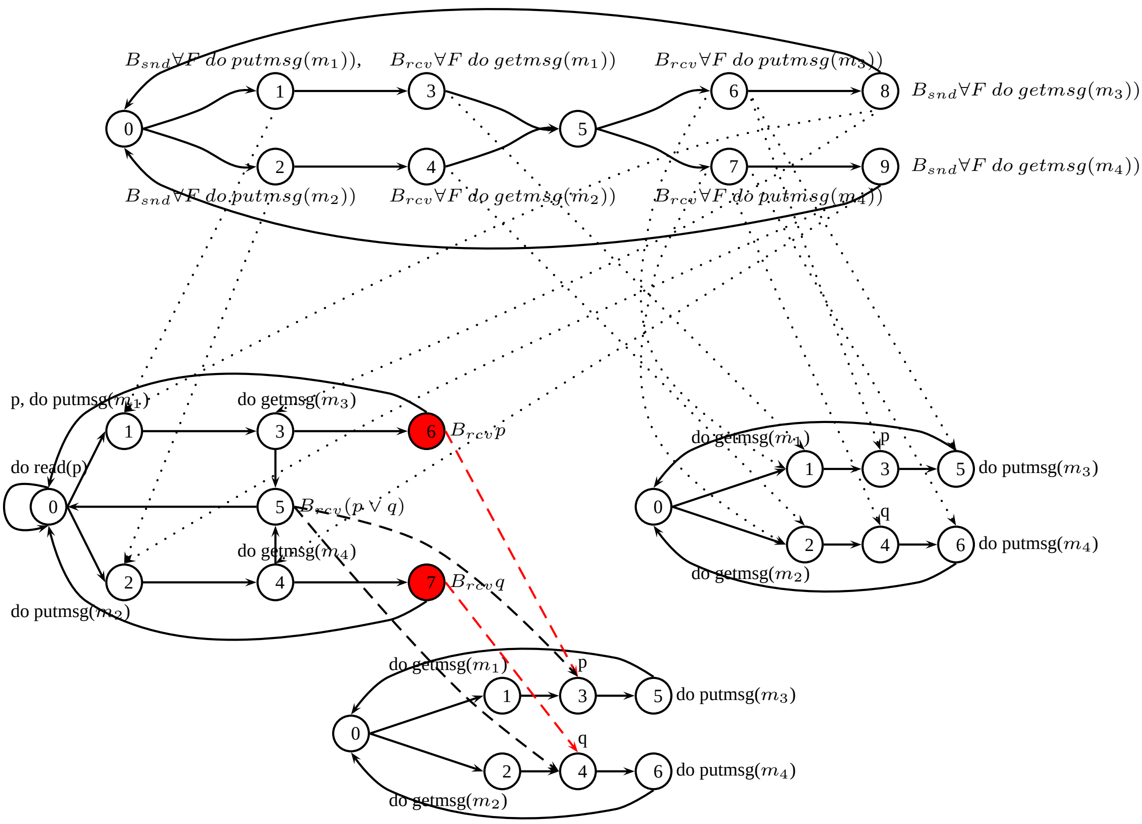

Figure 2: Construction of the possible worlds

most and interesting extension is to treat the case of functional dependencies between the mental attitudes, where a mental attitude is considered to be a function of one or more other mental attitudes.

References

[1] Benerecetti M., F. Giunchiglia, and L. Serafini (1998) Model checking multi-agent systems, *Journal of Logic and Computation* **8(3)**, pp. 401–423.

[2] Bordini R. H., M. Fisher, C. Pardavila, and M. Wooldridge (2003) Model checking AgentSpeak. *Proceedings of the Second International Joint Conference on Autonomous Agents and Multi-agent Systems (AAMAS'03)*.

[3] Bratman, M. E., D. Israel, and M. E. Pollack (1988) Plans and resource bounded practical reasoning, *Computational Intelligence* **4**, pp. 349–355.

[4] Clarke, E. M., O. Grumberg, and D. A. Peled (1999) *Model Checking*, MIT Press.

[5] Cohen P. R., and H. J. Levesque (1990) Intention is Choice with Commitment, *Artificial Intelligence* **42**, pp. 213–261.

[6] Doyle J. (1992) Rationality and its roles in reasoning, *Computational Intelligence* **8(2)**, pp. 376–409.

[7] Emerson E. A., and J. Srinivasan (1989) Branching time temporal logic, Linear Time, Branching Time and Partial Order, *Proceedings of Logics and Models for Concurrency*, Springer-Verlag, pp. 123–172.

[8] Ghidini C. and F. Giunchiglia (2001) Local Models Semantics, or Contextual Reasoning = Locality + Compatibility, *Artificial Intelligence* **127(2)**, pp. 221–259.

[9] Ghidini C. and L. Serafini (1994) Multi-language hierarchical logics (or: how we can do without modal logics), *Artificial Intelligence* **65**, pp. 29–70.

[10] Halpern J. Y., and Y. O. Moses (1990) A guide to completeness and complexity for modal logics of knowledge and belief, *Artificial Intelligence* **54**, pp. 319–379.

[11] van der Meyden R. and M. Y. Vardi (1998) Synthesis from Knowledge-Based Specifications, *Proceedings of the 9th International Conference on Concurrency Theory (CONCUR'98)*.

[12] Raimondi F. and A. Lomuscio (2004) Verification of multi-agent systems via ordered binary decision diagrams: an algorithm and its implementation, *Pro-*

ceedings of the First International Joint Conference on Autonomous Agents and Multi-agent Systems (AAMAS'04).

[13] Raimondi F. and A. Lomuscio (2003) A tool for specification and verification of epistemic and temporal properties of multi-agent system, *Electronic Notes in Theoretical Computer Science*.

[14] Rao A. S. (1996) AgentSpeak(L): BDI agents speak out in a logical computable language, *Lecture Notes in Computer Science*.

[15] Rao A. S., and M. Georgeff (1998) Decision procedures for BDI logics, *Journal of Logic and Computation* **8(3)**, pp. 293–344.

[16] Rao A. S., and M. P. Georgeff (1991) Modeling rational agents within a BDI architecture, *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning*, Morgan Kaufmann.

[17] Rao A. S., and M. P. Georgeff (1992) An abstract architecture for rational agents, *Knowledge Representation and Reasoning*, pp. 439–449.

[18] Rosenschein S. J., and L. P. Kaelbling (1986) The synthesis of digital machines with provable epistemic properties, *Proceedings of the First Conference on Theoretical Aspects of Reasoning about Knowledge*, Morgan Kaufmann, 1986.

[19] Shoham Y. (1991) Agent0: A simple agent language and its interpreter, *Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI91)*, pp. 704–709.

[20] Woodridge M. (2000) Computationally grounded theories of agency, *Fourth International Conference on Multi-Agent Systems (ICMAS-2000)*, pp. 13–20.

[21] Woodridge M., and M. Fisher (1994) A decision procedure for a temporal belief logic, *Proceedings of the First International Conference on Temporal Logic*.

[22] Wooldridge M, Fisher M., M.P. Huget, and S. Parsons (2002) Model checking multi-agent systems with MABLE. *Proceedings of the First International Joint Conference on Autonomous Agents and Multi-agent Systems (AAMAS'02)*.