The interpreted system model of knowledge, belief, desire and intention

Author
Su, K, Luo, X, Sattar, A, Orgun, MA

Published
2006

Conference Title
Proceedings of the International Conference on Autonomous Agents

DOI
10.1145/1160633.1160668

Rights statement
© ACM, 2006. This is the author’s version of the work. It is posted here by permission of ACM for your personal use. Not for redistribution. The definitive version was published in Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems, ISBN 1595933034, http://doi.acm.org/10.1145/1160633.1160668

Downloaded from
http://hdl.handle.net/10072/12087

Link to published version
http://portal.acm.org/toc.cfm?id=1160633&type=proceeding&coll=GUIDE&dl=GUIDE&CFID=23772535&CFTOKEN=76851635

Griffith Research Online
https://research-repository.griffith.edu.au
The Interpreted System Model
of Knowledge, Belief, Desire and Intention

Kaile Su¹, Xiangyu Luo¹, Abdul Sattar¹, Mehmet A Orgun²
¹Institute for Integrated and Intelligent Systems, Griffith University, Brisbane, Australia
²Department of Computing, Macquarie University, Sydney, Australia
{k.su, x.luo, a.sattar}@griffith.edu.au; mehmet@comp.mq.edu.au

ABSTRACT

We present a new model of knowledge, belief, desire and intention, called the interpreted KBDI-system model (or KBDI-model for short). The key point of the interpreted KBDI-system model is that we express an agent’s knowledge, belief, desire and intention as a set of runs (computing paths), which is exactly a system in the interpreted system model, a well-known agent model due to Halpern and his colleagues. Our KBDI-model is computationally grounded in that we are able to associate a KBDI-model with a computer program, and formulas, involving agents’ knowledge, belief, desire (goal) and intention, can be understood as properties of program computations. We present a sound and complete proof system with respect to our KBDI-model and explore how symbolic model checking techniques can be applied to model checking multi-agent systems with KBDI-models.

1. INTRODUCTION

The possible worlds semantics [1] is a fruitful approach to formalizing agent systems via modal logics. The well-known theory of intention [5] and the formalism of the belief-desire-intention paradigm [10], for example, are along this line. However, it is still not very clear how to obtain concrete agent models with the belief, desire and intention accessibility relations from specific agent programs.

The interpreted system model [13] offers a natural interpretation, in terms of the states of computer processes, to S5 epistemic logic. The salient point of the interpreted system model is that we are able to associate an interpreted system with a computer program, and formulas in epistemic logic that are valid with respect to the interpreted system can be understood as valid properties of program computations. In this sense, the interpreted system model is computationally grounded [14].

The aim of this paper is to present a computationally grounded model of knowledge, belief, desire and intention, called the interpreted KBDI-system model (or KBDI-model for short), by extending the interpreted system model. The key point of the KBDI-model is that an agent’s beliefs, desires and intentions as well as its knowledge is characterized as a set of runs (computing paths), which is exactly a computationally grounded system in the interpreted system model.

Intuitively, a KBDI-model includes a system $K$ as in the interpreted system model, as well as, for each agent $i$, its beliefs $B_i$, desires $D_i$, and intentions $I_i$, which are subsystems or subsets of $K$. Agent $i$’s knowledge is determined by the set of those runs that are consistent with its local observations (or local state). Similarly, agent $i$’s beliefs are defined by the set of runs in $B_i$ that are consistent with agent $i$’s observations. Agent $i$’s desires and intentions can also be defined in the same way. Intuitively, runs in $B_i$ are possible computing paths from the viewpoint of the agent and those in $D_i$ are the computing paths that the agent desires. Thus, it is reasonable to assume that $D_i \subseteq B_i$ because every desired computing path should be possible. Nevertheless, we need not assume that $I_i \subseteq D_i$ or even $I_i \subseteq B_i$ because an agent’s intention may fail to achieve its goal and the actual computing path may be beyond the agent’s belief even though the agent has chosen and completed an intentional series of actions.

2. KBDI LOGIC

This section introduces a multimodal logic of knowledge, belief, desire and intention, referred to as Observation-based KBDI logic (KBDI). As shown below, the semantics of KBDI logic is given in terms of the interpreted KBDI-system model. According to this semantics, the computation of agents’ knowledge, belief, desire, and intention is based on agents’ observations, that is, local states.

2.1 Interpreted systems

Consider a system composed of multiple (say $n$) agents in an environment. We represent the system’s state or the global state as a tuple $(s_0, s_1, \ldots, s_n)$, where $s_i$ is the environment’s local state and, for $1 \leq i \leq n$, is agent $i$’s local state.

Let $L_e$ be a set of possible local states of the environment and $L_i$ a set of possible local states for agent $i$, for $i = 1, \ldots, n$. We take $G \subseteq L_e \times L_1 \times \cdots \times L_n$ to be the set of reachable global states of the system. A run $r$ over $G$ is a function from the time domain— the natural numbers in our case— to $G$. Thus, a run over $G$ can be identified with a sequence of global states in $G$. A point $(r, m)$ consists of a run $r$ and time $m$. Given a point $(r, m)$, we denote the first component of the tuple $r(m) = (s_0, s_1, \ldots, s_n)$ by $r_e(m) = s_0$ and, for each $i$ ($1 \leq i \leq n$), the $i + 1$'th component of the tuple $r(n)$ by $r_i(n) = s_i$. Thus, $r_e(m)$ is the local state of agent $i$ in run $r$ at “time” $m$.

The idea of the interpreted system semantics is that a run repre-
sent one possible computation of a system and a system may have a number of possible runs, so we say a system is a set of runs. Assume that we have a set $Φ$ of primitive propositions, which we can think of as describing basic facts about the system. An interpreted system $I$ consists of a pair $(R, π)$, where $R$ is a set of runs over a set of global states and $π$ is a valuation function, which gives the set of primitive propositions true at each point in $R$.

For every agent $i$, let the notation $(r, u) ≈_{i} (r′, v)$ denote that $r_{i}(u) = r′_{i}(v)$. Intuitively, $(r, u) ≈_{i} (r′, v)$ means that $(r, u)$ and $(r′, v)$ are indistinguishable to agent $i$. We also use the notation $(r, u) ≈_{spr} (r′, v)$ to denote that $u = v$ and, for every $j ≤ u$, $r_{i}(j) = r′_{i}(j)$ (here $spr$ stands for synchronous systems with perfect recall).

Let $KL_{n}$ denote the language of propositional logic augmented by the future-time connectives $⊙$ (next) and $U$ (until) and a modal (knowledge) operator $K_{i}$ for each agent $i$. The language $KL_{n}$ can be interpreted by using an interpreted system. The related satisfaction relationship $|=_{KL_{n}}|$ is as follows: Given $I = (R, π)$ and a point $(r, u) ∈ R$, we define $(I, r, u) |=_{KL_{n}} ϕ$ by induction on the structure of $ϕ$. When $ϕ$ is of the form $K_{i}ϕ$ then $(I, r, u) |=_{KL_{n}} ϕ$ iff $(I, r′, v) |=_{KL_{n}} ϕ$ for all $(r′, v)$ such that $(r, u) ≈_{i} (r′, v)$.

### 2.2 Interpreted KBDI-Systems

Given a set $G$ of global states and system $K$ over $G$, an agent’s mental state over system $K$ is a tuple $⟨B, D, I⟩$, where $B$, $D$ and $I$ are sets (sets of runs over $G$) such that $I ⊆ K$ and $D ⊆ B ⊆ K$. A KBDI-system is a structure $⟨K, M_{1}, \ldots , M_{n}⟩$, where $K$ is a system and for every $i, M_{i}$ is agent $i$’s mental state over $K$.

Assume that we have a set $Φ$ of primitive propositions which describe basic facts about agents and their environment. An interpreted KBDI-system $I$ consists of a pair $(S, π)$, where $S$ is a KBDI-system and $π$ is a valuation function, which gives the set of primitive propositions true at each point in $G$.

### 2.3 Syntax

Given a set $Φ$ of propositional atoms, the language of KBDI logic is defined by the following BNF notations:

\[
\langle \text{wff} \rangle ::= \text{any element of } Φ | \neg \langle \text{wff} \rangle | \langle \text{wff} \rangle \land \langle \text{wff} \rangle | \langle \text{wff} \rangle \lor \langle \text{wff} \rangle | \langle K_{i} \langle \text{wff} \rangle \rangle | \langle B_{i} \langle \text{wff} \rangle \rangle | \langle D_{i} \langle \text{wff} \rangle \rangle | \langle I_{i} \langle \text{wff} \rangle \rangle
\]

Informally, $K_{i}ϕ$, $B_{i}ϕ$, and $D_{i}ϕ$ mean that agent $i$ knows, believes and desires $ϕ$, respectively, while $I_{i}ϕ$ denotes that $ϕ$ holds under the assumption that agent $i$ acts based on his intention. The formulas not containing modalities $K_{i}$, $B_{i}$, $D_{i}$ and $I_{i}$ $(i = 1, \ldots , n)$ are called linear-temporal logic (LTL) formulas.

### 2.4 Semantics

We now proceed to interpret KBDI logic formulas in terms of interpreted KBDI-systems. In the following, we inductively define the satisfaction relation |= over a formula $ϕ$ and a pair of interpreted KBDI-system and a point. Given an interpreted KBDI-system $I = (S, π)$, suppose that $S = (K, M_{1}, \ldots , M_{n})$ and for every $i, M_{i} = (B_{i}, D_{i}, I_{i})$. Let $r$ be a run in $K$ and $u$ a natural number, then we have that:

- $(I, r, u) |=_{O} K_{i}ϕ$ iff $(I, r′, v) |=_{O} ϕ$ for those $(r′, v)$ such that $r′ ∈ K$ and $(r, u) ≈_{i} (r′, v)$;
- $(I, r, u) |=_{O} B_{i}ϕ$ iff $(I, r′, v) |=_{O} ϕ$ for those $(r′, v)$ such that $r′ ∈ B_{i}$ and $(r, u) ≈_{i} (r′, v)$;
- $(I, r, u) |=_{O} D_{i}ϕ$ iff $(I, r′, v) |=_{O} ϕ$ for those $(r′, v)$ such that $r′ ∈ D_{i}$ and $(r, u) ≈_{i} (r′, v)$;
- $(I, r, u) |=_{O} I_{i}ϕ$ iff $(I, r′, v) |=_{O} ϕ$ for those $(r′, v)$ such that $r′ ∈ I_{i}$ and $(r, u) ≈_{i} (r′, v)$.

The semantics of atomic formulas $ϕ$ or formulas of the form $¬ϕ$, $ϕ \land ϕ′$, $ϕ \lor ϕ′$ or $ϕ \equiv ϕ′$ can be dealt with in the usual manner.

We use $|=_{O} ϕ$ to denote that $ϕ$ is valid in every interpreted KBDI-system.

According to our definition, $D_{i}ϕ$ is true iff $ϕ$ is true along those runs that are desirable to agent $i$ and consistent with agent $i$’s observations. Thus, $D_{i}ϕ$ intuitively means that agent $i$’s goal implies that formula $ϕ$ holds.

For those agents with perfect recall and a global clock, we may use $≈_{spr}$ instead of $≈_{i}$ to interpret those formulas with modalities $B_{i}, D_{i}$ and $I_{i}$ and get an alternative satisfaction relationship $|=_{spr}$.

**Proposition 1.** The following axioms are valid with respect to both $|=_{O}$ and $|=_{O}^{spr}$:

- $K_{i}ϕ \Rightarrow ϕ$
- $Δ_{i}(ϕ = ψ) ⇒ (Δ_{i}ϕ ⇒ Δ_{i}ψ)$
- $Δ_{i}ϕ ⇒ K_{i}Δ_{i}ϕ, ¬Δ_{i}ϕ ⇒ K_{i}¬Δ_{i}ϕ$

where $Δ$ stands for $K, B, D$ or $I$.

- **Relationship between knowledge, belief, desire and intention**
- $B_{i}ϕ ⇒ D_{i}ϕ, K_{i}ϕ ⇒ B_{i}ϕ, K_{i}ϕ ⇒ I_{i}ϕ$
- **Temporal operators**
  - $(ϕ ≈ ψ) ⇒ (Oϕ ≈ Oψ)$
  - $¬(¬ϕ) ⇒ ϕ$
  - $ϕUψ ≡ ϕ ∨ (ϕ ∧ O(ϕUψ))$

**Proposition 2.** The following axioms are valid with respect to $|=_{spr}$:

The formula $D_{i}ϕ ≈ ϕ ⇒ D_{i}ϕ$ says that if agent $i$’s current goal implies $ϕ$ holds at the next point in time, then at the next point in time her goal will imply $ϕ$, that is, agent $i$ persists on her goal.

### 2.5 Proof system

We now discuss a proof system, called the KBDI proof system, for those agents with perfect recall and a global clock. The proof system contains the axioms of propositional calculus plus those in Propositions 1 and 2. It is closed under the propositional inference rules plus for every agent $i$: $\frac{ϕ}{ϕ_{K_{i}}}$.

**Theorem 3.** The KBDI proof system for agents with perfect recall and a global clock is sound and complete with respect to interpreted KBDI-systems with satisfaction relation $|=_{spr}$.

### 3. MODEL CHECKING KBDI LOGIC

The model checking problem is to give an algorithm to test whether a model satisfies a concerned specification expressed by a formula. However, in order to make our model checking algorithm practically useful, we must consider from where our model, an interpreted KBDI-system, comes. It would be satisfactory if we can derive our model from a program implemented in, say C or Java. However, to simplify the matter, we may consider some abstract programs such as finite-state programs, which are expressive enough from the standpoint of theoretical computer science. Moreover, to make our model checking system practically efficient, we use symbolic model checking techniques. Thus, a finite-state program in our approach is represented in a symbolic way.

#### 3.1 Symbolic Representation of Interpreted KBDI-System Model

A system as a set of infinite runs is not well suited to model checking directly as it is generally applied to finite state systems. However, we can represent an interpreted system as a finite-state program $(G, G_{0}, R, V)$, where $G$ is a set of states, $G_{0}$ a set of initial states, and $R$ a total “next time” relation, and $V$ associates
each state with a truth assignment function. A set of infinite runs is then obtained by “unwinding” the relation \( R \) starting from initial states in \( G_0 \).

We can present a symbolic representation of a finite-state program \( (G, G_0, R, V) \) as \( (x, \theta(x), \tau(x, x')) \). Also, we represent a system as a (symbolic) finite-state program with \( n \) agents, which is defined as a tuple \( \mathcal{P} = (x, \theta(x), \tau(x, x'), O_1, \ldots, O_n) \), where

1. \( x \) is a set of system variables;
2. \( \theta \) is a boolean formula over \( x \), called the initial condition;
3. \( \tau \) is a boolean formula over \( x \cup x' \), called the transition relation; and
4. for each \( i \), \( O_i \subseteq x \), containing agent \( i \)'s local variables, or observable variables.

For convenience, we may use \( \mathcal{P}(\theta, \tau) \) to denote a finite-state program with \( n \) agents \( (x, \theta(x), \tau(x, x'), O_1, \ldots, O_n) \), if \( x \) and \( O_1, \ldots, O_n \) are clear from the context.

Given a finite-state program \( \mathcal{P}(\theta, \tau) \) with \( n \) agents, we define an agent’s internal program over \( \mathcal{P}(\theta, \tau) \) as a tuple

\[
(\mathcal{P}(\theta_1, \tau_1), \mathcal{P}(\theta_2, \tau_2), \mathcal{P}(\theta_3, \tau_3))
\]

where, for \( j = 1, 2, 3 \), \( \theta_j \Rightarrow \theta \lor \tau_j \Rightarrow \tau \) is valid, and \( \theta_2 \Rightarrow \theta_1 \land \tau_2 \Rightarrow \tau_1 \) is valid.

Clearly, an agent’s internal program is exactly related to an agent’s mental state. Thus, we define a symbolic BDI-program with \( n \) agents as a tuple \( P_n = (\mathcal{P}_K, P_1, \ldots, P_n) \), where \( \mathcal{P}_K \) is a finite-state program with \( n \) agents and for each agent \( i \), \( P_i \) is agent \( i \)'s internal program over \( \mathcal{P}_K \). We use \( I_{\mathcal{P}_n} \) to denote the corresponding interpreted KBDI-system.

### 3.2 Simplified BDI-Programs

To obtain a BDI-program with \( n \) agents, one should define \( 1 + 3n \) finite-state programs. Our experience with a previous version of MCKBDI indicated that it is inconvenient for a programmer to code a BDI-program with \( n \)-agents because there may be many finite-state programs involved.

Therefore, we should simplify the definition of a BDI-program with \( n \)-agents.

**Definition 4.** Let \( \mathcal{P} = (x, \theta(x), \tau(x, x'), O_1, \ldots, O_n) \) be a finite-state program with \( n \)-agents, where, for each agent \( i \), \( O_i \) contains at least three particular boolean variables \( \text{WB}_i, \text{WD}_i, \) and \( \text{WI}_i \). A simplified BDI-program with \( n \)-agents derived from \( \mathcal{P} \) is a tuple \( (\mathcal{P}_E, P_1, \ldots, P_n) \), where for each agent \( i \), \( P_i = (\mathcal{P}(\theta_1, \tau_1)) \), \( \mathcal{P}(\theta_2, \tau_2), \mathcal{P}(\theta_3, \tau_3) \) such that

1. \( \theta_1 = \theta \land \text{WB}_i \) and \( \tau_1 = \tau \land \text{WB}_i \);
2. \( \theta_2 = \theta \land \text{WB}_i \land \text{WD}_i \) and \( \tau_2 = \tau \land \text{WB}_i \land \text{WD}_i \);
3. \( \theta_3 = \theta \land \text{WI}_i \) and \( \tau_3 = \tau \land \text{WI}_i \).

It is more convenient for a programmer to code a simplified BDI-program with \( n \)-agents, and it seems that the notion of a simplified BDI-Program does not lose much generality of that of a BDI-Program.

### 4. CONCLUDING REMARKS

We have proposed a new KBDI-model by using interpreted systems, and developed a computationally grounded BDI logic, called KBDI logic. We interpret BDI logic via two different semantics. One of them is based on the assumption that agents have perfect recall and there is a global clock, with respect to which a sound and complete proof system has been presented. We have explored how symbolic model checking techniques can be applied to model checking BDI-agents and obtained encouraging preliminary experimental results.

This paper is inspired by Su [11] and Su et al [12], which also provide agent models by extending the interpreted system model. However, Su [11] does not consider agents’ belief, desire and intention. The main advantages of the present paper over Su et al [12] are: (1) We introduce, for each agent \( i \), not only belief, desire and intention modalities \( B_i, D_i, \) and \( I_i \), but also a knowledge modality \( K_i \). Moreover, the resulting axiomatic system is more succinct and proved to be sound and complete. (2) A model checker with KBDI-models, called MCKBDI, is developed [4] Empirical results are more comprehensive, supported by convincing and motivating examples, where agents’ observations and actions are explicitly defined.

Model checking multi-agent systems has become an active research topic in the community of multi-agent systems. Many efforts have been devoted to model checking knowledge in multi-agent systems [5, 13, 14, 7]. However, comparatively little work has been carried out on model checking BDI-agents. Some general approaches to model checking BDI-agents were proposed in Rao and Georgeff [9] and Benerecetti and Cimatti [1]. But no method was given for generating models from actual systems, and so the techniques given there could not easily be applied to verifying real multi-agent systems. In Bordini et al [2], model checking techniques for AgentSpeak(L) [8] have been reported; however, AgentSpeak(L) has very simple semantics. The salient point of our work is that we present a general form of a BDI agent program, from which BDI agent models are generated and specifications in full BDI logics can be verified by symbolic model checking techniques.

### 5. REFERENCES


1The source codes and the user manual of MCKBDI are available at http://www.cit.gu.edu.au/~s360126/arc-mml/