Symbolic model checking of multi-agent systems using OBDDs

Franco Raimondi, Alessio Lomuscio
Department of Computer Science
King's College London
London, UK
email: {franco,alessio}@dcs.kcl.ac.uk

Abstract

We present an algorithm for symbolic model checking temporal-epistemic properties of multi-agent systems, expressed in the formalism of interpreted systems. We first introduce a technique for the translation of interpreted systems into boolean formulae, and then present a model-checking algorithm based on this translation. The algorithm is tailored for the use of OBDDs, as they offer a compact and efficient representation for boolean formulae.

1 Introduction

Theoretical investigations in the area of multi-agent systems (MAS) have traditionally focused on the use of MAS as a *specification* tool for complex systems. Various logics have been explored to give formal foundations to MAS, particularly for *mental attitudes* [9] of agents, such as knowledge, belief, desire, etc. To consider the temporal evolution of these attitudes, temporal logics such as CTL and LTL [8] have been included in MAS formalisms, thereby producing combinations of temporal logic with, for example, epistemic, doxastic, and deontic logics.

Although the problem of specifying systems' behaviour is worth investigating, the problem of *verification* of MAS must also be taken into account to ensure that systems behave as they are supposed to. *Model checking* is a well-established verification technique for distributed systems specified by means of temporal logics [5, 8]. The problem of model checking is to verify whether a logical formula φ expressing a certain required property is true in a model M representing the system, that is establishing whether or not $M \models \varphi$. This approach can also be applied to MAS, where in this case M is a semantical model representing the evolutions of the MAS, and φ is a formula expressing temporal-intentional properties of the agents. Recent work along these lines includes [15], in which M. Wooldridge et al. present the MABLE language for the specification of MAS. In this work, modalities are translated simply as nested data structures (in the spirit of [1]). Bordini et al. [2] use a modified version of the AgentSpeak(L) language [12] to specify agents and to exploit existing model checkers. Both the works of M. Wooldridge et al. and of Bordini et al. translate the specification into a SPIN specification to perform the verification. The works of van der Meyden and Shilov [13], and van der Meyden and Su [14], are concerned with verification of interpreted systems. They consider the verification of a particular class of interpreted systems, namely the class of synchronous distributed systems with perfect recall. An algorithm for model checking is introduced in the first paper using automata, while in [14] verification is performed for a specific class of temporal specifications and interpreted systems; also, [14] suggests the use of OBDDs for this approach.

The aim of this paper is to present an algorithm for model checking epistemic and temporal properties of interpreted systems [6]. This differs from previous work by treating all the modalities explicitly in the verification process. We do temporal-epistemic model checking because the verification of epistemic properties (and their evolution with time) is crucial in many scenarios, including communication protocols and security protocols.

Interpreted systems are a formalism for representing epistemic properties of MAS and their evolution with time. The algorithm that we present does not involve the translation into existing model checkers, it is fully *symbolic*, and it is based on boolean functions. Boolean functions can be represented and manipulated efficiently by means of OBDDs, as it has been shown for CTL model checking [10].

The rest of the paper is organised as follows: in Section 2 we briefly review OBBDs-based model checking and the formalism of interpreted systems. In Section 3.1 we present the translation of interpreted systems into boolean formulae, while in Section 3.2 we introduce an algorithm based on this translation. We conclude in Section 4.

2 Preliminaries

2.1 Model checking and OBDDs

Given a model M and a formula φ in some logic, the problem of *model checking* involves establishing whether or not $M \models \varphi$ holds. Tools have been built to perform this task automatically, where M is a model of some temporal logic [5, 8, 7]. SMV [10] and SPIN [7] are two well-known model checkers; in these tools the model is given indirectly by means of a program P. It is not efficient to build explicitly the model M represented by P, because M has a size which is exponential in the number of variables of P (this fact is known as the *state explosion problem*). Instead, various techniques have been developed to perform *symbolic model checking*, which is the problem of model checking where the model M is not described or computed in extension. Techniques for symbolic model checking include using automata [7] and OBDDs [3] for the representation of all the parameters needed by the algorithms. For the purpose of this paper, we will only consider symbolic model checking of the temporal logic CTL using OBDDs [4].

OBDDs (Ordered Binary Decision Diagrams) are an efficient representation for the manipulation of boolean functions. In [3] algorithms are provided for the manipulation and composition of OBDDs. The idea of CTL model checking using OBDDs is to represent states of the model and relations by means of boolean formulae. Moreover, CTL formulae are represented by sets of states, i.e. by boolean formulae. Model checking is then performed by composing OBDDs, or by computing fix-points of operators on OBDDs (see [8] for details). By means of this approach large systems have been checked, including hardware and software components.

2.2 Interpreted Systems

An interpreted system is a semantic structure representing the temporal evolution of a system of agents. Each agent i ($i = \{1, \ldots, n\}$) is characterised by a set of *local states* L_i and by a set of actions Act_i that may be performed. Actions are performed in compliance with a protocol $P_i : L_i \rightarrow 2^{Act_i}$; notice that this definition allows for non-determinism. A tuple $g = (l_1, \ldots, l_n) \in L_1 \times \ldots, L_n$, where $l_i \in L_i$ for each i, is called a *global state* and gives a snapshot of the system. Given a set I of *initial global states*, the evolution of the system is described by n evolution functions[1]: $t_i : L_1 \times \ldots \times L_n \times Act_1 \times \ldots \times Act_n \rightarrow L_i$ In this formalism the environment in which agents "live" is usually modelled by means of a special agent E; we refer to [6] for more details.

The set I, t_i and the protocols P_i generate a set of *computations* (also called *runs*). Formally, a computation π is a sequence of global states $\pi = (g_0, g_1, \ldots)$ such that $g_0 \in I$ and, for each pair $(g_j, g_{j+1}) \in \pi$, there exists a set of actions a enabled by the protocols such that $t(g_j, a) = g_{j+1}$. $G \subseteq (L_1 \times \ldots \times L_n)$ denotes the set of *reachable* global states.

Interpreted systems semantics can be used to interpret formulae of a temporal language enriched with epistemic operators [6]. Here we assume a temporal tree structure to interpret CTLK

[1] This definition is equivalent to the definition of a single evolution function t as in [6].

formulae [11]. The syntax of CTLK is defined in terms of a countable set of propositional variables $\mathcal{P} = \{p, q, \ldots\}$ and using the following modalities:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid EX\varphi \mid EG\varphi \mid E(\varphi U \varphi) \mid K_i\varphi$$

The modalities AX, EF, AF, AG, AU are derived in the standard way. Further, given a set of agents Γ, two group modalities can be introduced: $E_\Gamma\varphi$ and $C_\Gamma\varphi$ denote, respectively, that every agent in the group knows φ, and that φ is *common knowledge* in the group (see [6] for details).

Given a valuation function $\mathcal{V} : \mathcal{P} \to 2^G$, satisfaction in a global state g is defined as follows:

$g \models p$	iff	$g \in \mathcal{V}(p)$,
$g \models \neg\varphi$	iff	$g \not\models \varphi$,
$g \models \varphi_1 \vee \varphi_2$	iff	$g \models \varphi_1$ or $g \models \varphi_2$,
$g \models EX\varphi$	iff	there exists a computation π such that $\pi_0 = g$ and $\pi_1 \models \varphi$,
$g \models EG\varphi$	iff	there exists a computation π such that $\pi_0 = g$ and $\pi_i \models \varphi$ for all $i \geq 0$.
$g \models E(\varphi U \psi)$	iff	there exists a computation π such that $\pi_0 = g$ and a $k \geq 0$ such that $\pi_k \models \psi$ and $\pi_i \models \varphi$ for all $0 \leq i < k$,
$g \models K_i\varphi$	iff	$\forall g' \in G$, $g \sim_i g'$ implies $g' \models \varphi$
$g \models E_\Gamma\varphi$	iff	$\forall g' \in G$, $g \sim_\Gamma^E g'$ implies $g' \models \varphi$
$g \models C_\Gamma\varphi$	iff	$\forall g' \in G$, $g \sim_\Gamma^G g'$ implies $g' \models \varphi$

where π_j denotes the global state at place j in π. $\sim_i$ is an epistemic accessibility relation for agent i defined by: $g \sim_i g'$ iff $l_i(g) = l_i(g')$, i.e. if the local state of agent i is the same in g and in g' (notice that this is an equivalence relation). $g \sim_\Gamma^E g'$ iff $g \sim_i g'$ for some $i \in \Gamma$. $\sim_\Gamma^C$ is the reflexive transitive closure of $\sim_\Gamma^E$.

3 A model checking algorithm for CTLK

The main idea of this paper is to use algorithms based on OBDDs to verify temporal and epistemic properties of multi-agent systems, in the spirit of traditional model checking for temporal logics. To this end, it is necessary to encode all the parameters needed by the algorithms by means of boolean functions, and then to represent boolean functions by means of OBDDs. As this last step can be performed automatically using software libraries that are widely available, in this paper we introduce only the translation of interpreted systems into boolean formulae (Section 3.1). In Section 3.2 we present an algorithm based on this translation for the verification of CTLK formulae.

3.1 Translating an interpreted system into boolean formulae

The local states of an agent can be encoded by means of boolean variables (a boolean variable is a variable that can assume just one of the two values 0 or 1). The number of boolean variables needed for each agent is $nv(i) = \lceil log_2 |L_i| \rceil$. Thus, a global state can be identified by means of $N = \sum_i nv(i)$ boolean variables: $g = (v_1, \ldots, v_N)$. The evaluation function $\mathcal{V}$ associates a set of global states to each propositional atom, and so it can be seen as a boolean function. The protocols, too, can be expressed as boolean functions (actions being represented with boolean variables $(a_1, \ldots, a_M)$ similarly to global states).

The definition of t_i in Section 2.2 can be seen as specifying a list of *conditions* $c_{i,1}, \ldots, c_{i,k}$ under which agent i changes the value of its local state. Each $c_{i,j}$ has the form "if [conditions on global state and actions] then [value of "next" local state for i]". Hence, t_i can be expressed as a boolean formula as follows:

$$t_i = c_{i,1} \oplus \ldots \oplus c_{i,k}$$

where $\oplus$ denotes exclusive-or. We assume that the last condition $c_{i,k}$ of t_i prescribes that, if none of the conditions $c_{i,j} (j < k)$ is true, then the local state for i does not change. This assumption is key to keep compact the description of an interpreted system, as in this way only the conditions that are actually causing a change need to be listed.

The algorithm presented in Section 3.2 requires the definition of a boolean function $R_t(g, g')$ representing a temporal relation between g and g'. $R_t(g, g')$ can be obtained from the evolution function t_i as follows. First, we introduce a *global* evolution function t:

$$t = \bigwedge_{i \in \{1,\ldots,n\}} t_i = \bigwedge_{i \in \{1,\ldots,n\}} (c_{i,1} \oplus \ldots \oplus c_{i,k_i})$$

Notice that t is a boolean function involving two global states and a joint action $a = (a_1, \ldots, a_M)$. To abstract from the joint action and obtain a boolean function relating two global states only, we can define R_t as follows:

$R_t(g, g')$ iff $\exists a \in Act : t(g, a, g')$ is true and each local action $a_i \in a$ is enabled by the protocol of agent i in the local state $l_i(g)$.

The quantification over actions above can be translated into a propositional formula using a disjunction (see [10, 5] for a similar approach to boolean quantification):

$$R_t(g, g') = \bigvee_{a \in Act} [(t(g, a, g') \wedge P(g, a)]$$

where $P(g, a)$ is a boolean formula imposing that the joint action a must be consistent with the agents' protocols in global state g. R_t gives the desired boolean relation between global states.

3.2 The algorithm

In this section we present the algorithm SAT_{CTLK} to compute the set of global states in which a CTLK formula φ holds, denoted with $[[\varphi]]$. The following are the parameters needed by the algorithm:

- the boolean variable $(v_1, \ldots, v_N)$ and $(a_1, \ldots, a_M)$ to encode global states and joint actions;
- n boolean functions $P_i(v_1, \ldots, v_N, a_1, \ldots, a_M)$ to encode the protocols of the agents;
- the boolean function R_t to encode the temporal transition;
- the boolean function R_Γ^E to encode $\sim_\Gamma^E$, defined by $R_\Gamma^E = \bigwedge_{i \in \Gamma} R_i$.
- the function $\mathcal{V}(p)$ returning the set of global states in which the atomic proposition p holds. We assume that the global states are returned encoded as a boolean function of $(v_1, \ldots, v_N)$;
- a set of initial states I, encoded as a boolean function;
- the set of reachable states G. This can be computed as the fix-point of the operator $\tau = (I(g) \vee \exists g'(R_t(g', g) \wedge Q(g'))$ where $I(g)$ is true if g is an initial state and Q denotes a set of global states. The fix-point of τ can be computed by iterating $\tau(\emptyset)$ by standard procedure (see [10]);
- n boolean functions R_i to encode the accessibility relations $\sim_i$ (these functions are easily defined using equivalence on local states of G).

The algorithm is as follows:

```
SAT_CTLK(φ) {
  φ is an atomic formula: return V(φ);
  φ is ¬φ1: return G \ SAT_CTLK(φ1);
  φ is φ1 ∧ φ2: return SAT_CTLK(φ1)∩
    SAT_CTLK(φ2);
  φ is EXφ1: return EX_CTLK(φ1);
  φ is E(φ1Uφ2): return EU_CTLK(φ1, φ2);
  φ is EGφ1: return EG_CTLK(φ1);
  φ is K_iφ1: return K_CTLK(φ1, i);
  φ is E_Γφ1: return E_CTLK(φ1, Γ);
  φ is C_Γφ1: return C_CTLK(φ1, Γ);
}
```

In the algorithm above, EX_{CTLK}, EG_{CTLK}, EU_{CTLK} are the standard procedures for CTL model checking [8] in which the temporal relation is R_t and, instead of temporal states, global states are considered. The procedures $K_{CTLK}(\varphi, i)$ and $E_{CTLK}(\varphi, \Gamma)$ and $C_{CTLK}(\varphi, \Gamma)$ are presented below.

```
K_CTLK(φ, i) {
  X = SAT_CTLK(¬φ);
  Y = {g ∈ G | K_i(g, g') and g' ∈ X}
  return ¬Y;
}
```

```
E_CTLK(φ, Γ) {
  X = SAT_CTLK(¬φ);
  Y = {g ∈ G | R^E_Γ(g, g') and g' ∈ X}
  return ¬Y;
}
```

```
C_CTLK(φ, Γ) {
  X = SAT_CTLK(φ);
  Y = G;
  while ( X != Y ) {
    X = Y;
    Y = {g ∈ G | R^E_Γ(g, g') and g' ∈ Y and g' ∈ SAT_CTLK(φ)}
  return Y;
}
```

The procedure $C_{CTLK}(\varphi, \Gamma)$ is based on the equivalence [6]

$$C_\Gamma \varphi = E_\Gamma(\varphi \wedge C_\Gamma \varphi)$$

which implies that $[[C_\Gamma \varphi]]$ is the fix-point of the (monotonic) operator $\tau(Q) = [[E_\Gamma(\varphi \wedge (Q))]]$. Hence, $[[C_\Gamma \varphi]]$ can be obtained by iterating $\tau(G)$.

Notice that all the parameters can be encoded as OBDDs. Moreover, all the operations inside the algorithms can be performed on OBDDs as presented in [3].

To check that a formula holds in a model, it is enough to check whether or not the result of SAT_{CTLK} is equivalent to the set of reachable states.

4 Conclusion

Temporal logic model checking using OBDDs [10] is one of the most successful techniques for the verification of distributed systems. In the last decade, this methodology has been used for the verification of both software and hardware components.

In this paper we have presented an algorithm for the verification of temporal-epistemic properties based on the manipulation of boolean functions. The methodology presented here encodes directly a MAS (specified in the formalism of interpreted systems) by means of boolean formulae; then, the algorithm allows for the (fully symbolic) verification of temporal-epistemic properties. Moreover, the algorithm allows for the verification of two group modalities (E_Γ and C_Γ) and is not restricted to a particular class of interpreted systems, nor to a particular class of formulae. We are currently implementing the algorithm and in the future we aim at testing epistemic and temporal properties of various scenarios from the MAS literature. This will help in evaluating the efficiency of the algorithm.

References

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

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

[3] R. E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transaction on Computers*, pages 677–691, August 1986.

[4] J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and L. J. Hwang. Symbolic model checking: 10^{20} states and beyond. *Information and Computation*, 98(2):142–170, June 1992.

[5] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. The MIT Press, Cambridge, Massachusetts, 1999.

[6] Ronald Fagin, Joseph Y. Halpern, Yoram Moses, and Moshe Y. Vardi. *Reasoning about Knowledge*. The MIT Press, Cambridge, Massachusetts, 1995.

[7] G. J. Holzmann. The model checker spin. *IEEE transaction on software engineering*, 23(5), May 1997.

[8] M. R. A. Huth and M. D. Ryan. *Logic in Computer Science: Modelling and Reasoning about Systems*. Cambridge University Press, Cambridge, England, 2000.

[9] John McCarthy. Ascribing mental qualities to machines. In Martin Ringle, editor, *Philosophical Perspectives in Artificial Intelligence*, pages 161–195. Humanities Press, Atlantic Highlands, New Jersey, 1979.

[10] K. McMillan. *Symbolic model checking: An approach to the state explosion problem*. Kluwer Academic Publishers, 1993.

[11] W. Penczek and A. Lomuscio. Verifying epistemic properties of multi-agent systems via model checking. *Fundamenta Informaticae*, 55(2):167–185, 2003.

[12] A. S. Rao. AgentSpeak(L): BDI agents speak out in a logical computable language. *Lecture Notes in Computer Science*, 1038:42–??, 1996.

[13] R. van der Meyden and N. V. Shilov. Model checking knowledge and time in systems with perfect recall. *FSTTCS: Foundations of Software Technology and Theoretical Computer Science*, 19, 1999.

[14] R. van der Meyden and Kaile Su. Symbolic model checking the knowledge of the dining cryptographers. Submitted, 2002.

[15] Michael Wooldridge, Michael Fisher, Marc-Philippe Huget, and Simon Parsons. Model checking multi-agent systems with MABLE. In Maria Gini, Toru Ishida, Cristiano Castelfranchi, and W. Lewis Johnson, editors, *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'02)*, pages 952–959. ACM Press, July 2002.