Bounded Model Checking for Linear Time Temporal-Epistemic Logic*

Artur Męski[1,2], Wojciech Penczek[1,3], and Maciej Szreter[1]

1 **Institute of Computer Science, Polish Academy of Sciences, Poland**
{meski,penczek,mszreter}@ipipan.waw.pl
2 **University of Łódź, Faculty of Mathematics and Computer Science, Poland**
3 **University of Natural Sciences and Humanities, Siedlce, Poland**

Abstract

We present a novel approach to the verification of multi-agent systems using bounded model checking for specifications in LTLK, a linear time temporal-epistemic logic. The method is based on binary decision diagrams rather than the standard conversion to Boolean satisfiability. We apply the approach to two classes of interpreted systems: the standard, synchronous semantics and the interleaved semantics. We provide a symbolic algorithm for the verification of LTLK over models of multi-agent systems and evaluate its implementation against MCK, a competing model checker for knowledge. Our evaluation indicates that the interleaved semantics can often be preferable in the verification of LTLK.

1998 ACM Subject Classification D.2.4 Software/Program Verification

Keywords and phrases model checking, multi-agent systems, temporal-epistemic logic, verification, interpreted systems

Digital Object Identifier 10.4230/OASIcs.ICCSW.2012.88

1 Introduction

It is often crucial to ensure that multi-agent systems (MAS) conform to their specifications and exhibit some desired behaviour. This can be checked in a fully automatic manner using model checking [4], which is one of the rapidly developing verification techniques. Model checking has been studied by various researchers in the context of MAS and different modal logics for specifying MAS properties [2, 6, 7, 10, 13, 14, 20, 21].

In the verification of multi-agent systems, the construction of the full, reachable state-space is often required. This exploration can lead to the state-space explosion, where the size of the model grows exponentially with the number of agents. Therefore, several approaches alleviating this problem have been proposed. One of them is *bounded model checking* (BMC) [1], in which only a portion of the original model, truncated up to some specific depth, is considered. This approach can be combined either with a translation of the verification problem to the propositional satisfiability problem (SAT) [10, 18] or with techniques based on *binary decision diagrams* (BDDs) [9].

In this paper we present a novel approach to verification of MAS by BDD-based bounded model checking for linear time temporal logic extended with the epistemic component (LTLK, also called CKL_n [7]). The systems are modelled by two variants of Interpreted Systems: standard (IS) [5] and interleaved ones (IIS) [12]. IIS restrict IS by enforcing asynchronous

* Partly supported by National Science Centre under the grant No. 2011/01/B/ST6/05317 and 2011/01/B/ST6/01477. A longer version of this paper appeared in the proceedings of LAM'12 [16].

2012 Imperial College Computing Student Workshop (ICCSW'12).
Editor: Andrew V. Jones; pp. 88–94

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

semantics. This modifies the popular modelling approach for MAS by bringing the semantics known from verification of concurrent systems like networks of automata or variants of Petri nets. Our paper shows that the modelling approach has a very strong impact on the efficiency of verification. The experimental results exhibit that the IIS-based approach can greatly improve the practical applicability of the bounded model checking method for LTLK.

There has been already some intensive research on BMC for MAS, but mostly for the properties expressible in CTLK, based either on SAT [8, 18] or on BDDs [9]. A SAT-based verification method for the LTLK properties of MAS, modelled by IIS, was put forward in [19]. Our technical report [15] presents a BDD-based approach to verification of LTLK for IIS, while the SAT- and BDD-based approaches for IIS are compared in [17].

The rest of the paper is organised as follows. Sec. 2 provides the basic definitions and notations for LTLK and IS. Our method is described in Sec. 3. The last section contains the discussion of an experimental evaluation of the approach and the final remarks.

2 Preliminaries

2.1 Interpreted Systems

The semantics of *interpreted systems* [5] provides a setting to reason about MAS by means of specifications based on knowledge and linear or branching time. We begin by assuming a MAS to be composed of n agents[1] $\mathcal{A}$. We associate a set of *possible local states* L_i and *actions* Act_i to each agent $i \in \mathcal{A}$. We assume that the special action ϵ_i, called "null", or "silent" action of agent i belongs to Act_i; as it will be clear below the local state of agent i remains the same if the null action is performed. Also note we do not assume that the sets of actions of the agents are disjoint. We call $Act = \prod_{i \in \mathrm{A}} Act_i$ the set of all possible *joint actions*, i.e. tuples of local actions executed by agents. We consider a *local protocol* modelling the program the agent is executing. Formally, for any agent i, the actions of the agents are selected according to a *local protocol* $P_i : L_i \to 2^{Act_i}$. For each agent i, we define a relation $t_i \subseteq L_i \times Act \times L_i$, where $(l, (a_1, \ldots, a_n), l) \in t_i$ for each $l \in L_i$ if $a_i = \epsilon_i$. A *global state* $g = (g_1, \ldots, g_n)$ is a tuple of local states for all the agents corresponding to an instantaneous snapshot of the system at a given time. Given a global state $g = (g_1, \ldots, g_n)$ we denote by $l_i(g)$ the local component g_i of agent $i \in \mathcal{A}$ in g.

For each agent $i \in \mathcal{A}$, $\sim_i \subseteq G \times G$ is an *epistemic indistinguishability* relation over global states defined by $g \sim_i h$ if $l_i(g) = l_i(h)$. Further, let $\Gamma \subseteq \mathcal{A}$. The union of Γ's accessibility relations is defined as $\sim_\Gamma^E = \bigcup_{i \in \Gamma} \sim_i$. By $\sim_\Gamma^C$ we denote the transitive closure of $\sim_\Gamma^E$, whereas $\sim_\Gamma^D = \bigcap_{i \in \Gamma} \sim_i$.

A *global evolution* $\mathcal{T} \subseteq G \times Act \times G$ is defined as follows: $(g, a, h) \in \mathcal{T}$ iff there exists an action $a = (a_1, \ldots, a_n) \in Act$ such that for all $i \in \mathcal{A}$ we have $a_i \in P_i(l_i(g))$ and $(l_i(g), a, l_i(h)) \in t_i$. For $g, h \in G$ and $a \in Act$ s.t. $(g, a, h) \in \mathcal{T}$ we write $g \xrightarrow{a} h$. We assume that the global evolution relation $\mathcal{T}$ is total, i.e., for each $g \in G$ there exists $a \in Act$ and $h \in G$ such that $g \xrightarrow{a} h$.

An *infinite* sequence of global states and actions $\rho = g_0 a_0 g_1 a_1 g_2 \ldots$ is called a *path* originating from g_0 if there is a sequence of transitions from g_0 onwards, i.e., $g_i \xrightarrow{a_i} g_{i+1}$ for every $i \geq 0$. Any *finite* prefix of a path is called a *run*. By $length(\rho)$ we mean the number of the states of ρ if ρ is a run, and ω if ρ is a path. In order to limit the indices range of ρ which

[1] Note in the present study we do not consider the environment component. This may be added with no technical difficulty at the price of heavier notation.

can be a path or run, we define the relation $\trianglelefteq_\rho$. Let $\trianglelefteq_\rho \stackrel{def}{=} <$ if ρ is a path, and $\trianglelefteq_\rho \stackrel{def}{=} \leq$ if ρ is a run. A state g is said to be *reachable* from g_0 if there is a path or a run $\rho = g_0 a_0 g_1 a_1 g_2 \dots$ such that $g = g_i$ for some $i \geq 0$. The set of all the paths and runs originating from g is denoted by $\Pi(g)$. The set of all the paths originating from g is denoted by $\Pi^\omega(g)$.

▶ **Definition 1.** Given a set of propositions $\mathcal{PV}$ such that $\{true, false\} \subseteq \mathcal{PV}$, an *interpreted system* (IS), also called a *model*, is a tuple $M = (G, \iota, \mathcal{T}, \{\sim_i\}_{i\in\mathcal{A}}, \mathcal{V})$, where G is a set of global states, $\iota \in G$ is an initial (global) state such that each state in G is reachable from ι, $\mathcal{T}$ is the global evolution relation defined as above, and $V : G \to 2^{\mathcal{PV}}$ is a valuation function.

We define $\Pi = \bigcup_{g\in G} \Pi(g)$ to be the set of all the interleaved paths and runs originating from all states in G. By Π^ω we denote the set of all the paths of Π.

2.2 Interleaved Interpreted Systems

We define a restriction of interpreted systems, called *interleaved interpreted systems* in which global evolution function is restricted, so that every agent either executes a shared action or the null action. We assume that $\epsilon_i \in P_i(l)$, for any $l \in L_i$, i.e., we insist on the null action to be enabled at every local state. For each action $a \in \bigcup_{i\in\mathcal{A}} Act_i$ by $Agent(a) \subseteq \mathcal{A}$ we mean all the agents i such that $a \in Act_i$, i.e., the set of the agents potentially able to perform a. Then, the global evolution relation $\mathcal{T}$ is defined as before, but it is restricted by the following condition: if $(g, a, h) \in \mathcal{T}$ then there exists a joint action $a = (a_1, \dots, a_n) \in Act$, and an action $\alpha \in \bigcup_{i\in\mathcal{A}} Act_i \setminus \{\epsilon_1, \dots, \epsilon_n\}$ such that: $a_i = \alpha$ for all $i \in Agent(\alpha)$, and $a_i = \epsilon_i$ for all $i \in \mathcal{A} \setminus Agent(\alpha)$.

2.3 Syntax and Semantics of LTLK

▶ **Definition 2** (Syntax)**.** Let $\mathcal{PV}$ be a set of atomic propositions to be interpreted over the global states of a system, $p \in \mathcal{PV}$, $q \in \mathcal{A}$, and $\Gamma \subseteq \mathcal{A}$. Then, the syntax of LTLK is defined by the following BNF grammar: $\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \mathrm{X}\varphi \mid \varphi \mathrm{U} \varphi \mid \varphi \mathrm{R} \varphi \mid \mathrm{K}_q\varphi \mid \overline{\mathrm{K}}_q\varphi \mid \mathrm{E}_\Gamma\varphi \mid \overline{\mathrm{E}}_\Gamma\varphi \mid \mathrm{D}_\Gamma\varphi \mid \overline{\mathrm{D}}_\Gamma\varphi \mid \mathrm{C}_\Gamma\varphi \mid \overline{\mathrm{C}}_\Gamma\varphi$.

The temporal operators U and R are named as usual, respectively, *until* and *release*; X is the next step operator. The epistemic operators K_q, $\mathrm{D}_\Gamma, \mathrm{E}_\Gamma$, and C_Γ [5] represent, respectively, knowledge of agent q, distributed knowledge in the group Γ, "everyone in Γ knows", and common knowledge among agents in Γ, whereas $\overline{\mathrm{K}}_q$, $\overline{\mathrm{D}}_\Gamma, \overline{\mathrm{E}}_\Gamma$, and $\overline{\mathrm{C}}_\Gamma$ are the corresponding dual.

The logic LTL is the sublogic of LTLK which consists only of the formulae built without the epistemic operators, whereas ELTL is a fragment of LTL where negation can be applied to propositions only. ELTLK is the existential fragment of LTLK, defined by the following grammar: $\varphi ::= p \mid \neg p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \mathrm{X}\varphi \mid \varphi \mathrm{U} \varphi \mid \varphi \mathrm{R} \varphi \mid \overline{\mathrm{K}}_q\varphi \mid \overline{\mathrm{E}}_\Gamma\varphi \mid \overline{\mathrm{D}}_\Gamma\varphi \mid \overline{\mathrm{C}}_\Gamma\varphi$.

▶ **Definition 3** (Semantics)**.** Given a model $M = (G, \iota, \mathcal{T}, \{\sim_q\}_{q\in\mathcal{A}}, \mathcal{V})$, where $\mathcal{V}(s)$ is the set of propositions that hold at s, let Π be a set of all the paths and runs of M, and $\rho(i)$ denote the i-th state of a path or run $\rho \in \Pi$, and $\rho[i]$ denote the path or run ρ with a designated formula evaluation position i, where $i \trianglelefteq_\rho length(\rho)$. Note that $\rho[0] = \rho$. The formal semantics of LTLK is defined recursively as follows:

$M, \rho[i] \models p$ iff $p \in \mathcal{V}(\rho(i))$,
$M, \rho[i] \models \neg\varphi$ iff $M, \rho[i] \not\models \varphi$,

$$
\begin{array}{lll}
M,\rho[i] \models \varphi_1 \wedge \varphi_2 & \text{iff} & M,\rho[i] \models \varphi_1 \text{ and } M,\rho[i] \models \varphi_2,\\
M,\rho[i] \models \varphi_1 \vee \varphi_2 & \text{iff} & M,\rho[i] \models \varphi_1 \text{ or } M,\rho[i] \models \varphi_2,\\
M,\rho[i] \models \mathrm{X}\varphi & \text{iff} & \mathit{length}(\rho) > i \text{ and } M,\rho[i+1] \models \varphi,\\
M,\rho[i] \models \varphi_1 \mathrm{U} \varphi_2 & \text{iff} & (\exists k \geq i)[M,\rho[k] \models \varphi_2 \text{ and } (\forall i \leq j < k)\ M,\rho[j] \models \varphi_1],\\
M,\rho[i] \models \varphi_1 \mathrm{R} \varphi_2 & \text{iff} & [\rho \in \Pi^{\omega}(\iota) \text{ and } (\forall k \geq i)\ M,\rho[k] \models \varphi_2]\\
 & & \text{or } (\exists k \geq i)[M,\rho[k] \models \varphi_1 \text{ and } (\forall i \leq j \leq k)\ M,\rho[j] \models \varphi_2],\\
M,\rho[i] \models \mathrm{K}_q\varphi & \text{iff} & (\forall \rho' \in \Pi^{\omega}(\iota))(\forall k \geq 0)[\rho'(k) \sim_q \rho(i) \text{ implies } M,\rho'[k] \models \varphi],\\
M,\rho[i] \models \overline{\mathrm{K}}_q\varphi & \text{iff} & (\exists \rho' \in \Pi(\iota))(\exists k \geq 0)[\rho'(k) \sim_q \rho(i) \text{ and } M,\rho'[k] \models \varphi],\\
M,\rho[i] \models \mathrm{Y}_\Gamma\varphi & \text{iff} & (\forall \rho' \in \Pi^{\omega}(\iota))(\forall k \geq 0)[\rho'(k) \sim_\Gamma^{\mathrm{Y}} \rho(i) \text{ implies } M,\rho'[k] \models \varphi],\\
M,\rho[i] \models \overline{\mathrm{Y}}_\Gamma\varphi & \text{iff} & (\exists \rho' \in \Pi(\iota))(\exists k \geq 0)[\rho'(k) \sim_\Gamma^{\mathrm{Y}} \rho(i) \text{ and } M,\rho'[k] \models \varphi],
\end{array}
$$

where $\mathrm{Y} \in \{\mathrm{D},\mathrm{E},\mathrm{C}\}$.

Moreover, an ELTLK formula φ holds in the model M, denoted $M \models_\exists \varphi$, iff $M,\rho \models \varphi$ for some path or run $\rho \in \Pi(\iota)$. The intuition behind this definition is that ELTLK is obtained by restricting the syntax of the epistemic operators while the temporal ones remain the same.

3 BDD-based Bounded Model Checking for ELTLK

To perform BMC of ELTLK using BDDs [4] we combine the standard approach for ELTL [3] with the method for the epistemic operators [20] in a similar manner to the solution for CTL* of [4] where the methods for CTL and LTL are combined into a method for CTL*.

Algorithm 1 Labelling algorithm

```
1:  M_c := M, φ_c := φ
2:  while γ(φ_c) ≠ 0 do
3:     pick ψ ∈ 𝒴(φ_c) such that γ(ψ) = 1
4:     for all g ∈ [[M_c, sub(ψ)]] do
5:        𝒱_{M_c}(g) := 𝒱_{M_c}(g) ∪ {p_{sub(ψ)}}
6:     end for
7:     ψ := ψ[sub(ψ) ← p_{sub(ψ)}]
8:     for all g ∈ [[M_c, ψ]] do
9:        𝒱_{M_c}(g) := 𝒱_{M_c}(g) ∪ {p_ψ}
10:    end for
11:    φ_c := φ_c[ψ ← p_ψ]
12: end while
13: return [[M_c, φ_c]]
```

Labelling algorithm. Given a model $M = (G,\iota,\mathcal{T},\{\sim_q\}_{q\in\mathcal{A}},\mathcal{V})$, a set $G_R \subseteq G$ of its reachable states, and an ELTLK formula φ, we compute the set $[\![M,\varphi]\!] = \{g \in G_R \mid M,g \models_\exists \varphi\}$ by reducing ELTLK to ELTL under the assumption that we have the algorithms for computing this set for each φ being an ELTL formula or in the form $\mathrm{Y}p$, where $p \in \mathcal{PV}$, and $\mathrm{Y} \in \{\overline{\mathrm{K}}_q, \overline{\mathrm{E}}_\Gamma, \overline{\mathrm{D}}_\Gamma, \overline{\mathrm{C}}_\Gamma\}$ (we use the algorithms from [3] and [20], respectively). In order to obtain this set, we construct a new model M_c together with an ELTL formula φ_c, and compute the set $[\![M_c,\varphi_c]\!]$, which is equal to $[\![M,\varphi]\!]$. Initially φ_c equals φ, which is an ELTLK formula, and we process the formula in stages to reduce it to an ELTL formula by replacing with atomic propositions all its subformulae containing epistemic operators. If $\varphi = \mathrm{Y}\psi$ is an ELTLK formula, by $\mathit{sub}(\varphi)$ we denote the formula ψ nested in the epistemic operator Y. We begin by choosing some epistemic subformula ψ of φ_c, which consists of exactly one epistemic operator (line 3), and process it in two stages. First, we modify the valuation function of

M_c (line 5) such that every state initialising some path or run along which $sub(\psi)$ holds is labelled with the new atomic proposition $p_{sub(\psi)}$, and we replace with the variable $p_{sub(\psi)}$ every occurrence of $sub(\psi)$ in ψ (line 7). In the second stage, we deal with the epistemic operators having in their scopes atomic propositions only. By modifying the valuation function of M_c (line 9) we label with a new variable p_ψ every state initialising some path or run along which the modified simple epistemic formula ψ holds. Similarly to the previous stage, we replace every occurrence of ψ in φ_c with p_ψ (line 11). In the subsequent iterations, we process every remaining epistemic subformulae of φ_c in the same way until there are no more nested epistemic operators in φ_c (line 2), i.e., we obtain an ELTL formula φ_c, and the model M_c with the appropriately modified valuation function. Finally, we compute the set of all reachable states of M_c that initialise at least one path or run along which φ_c holds (line 13).

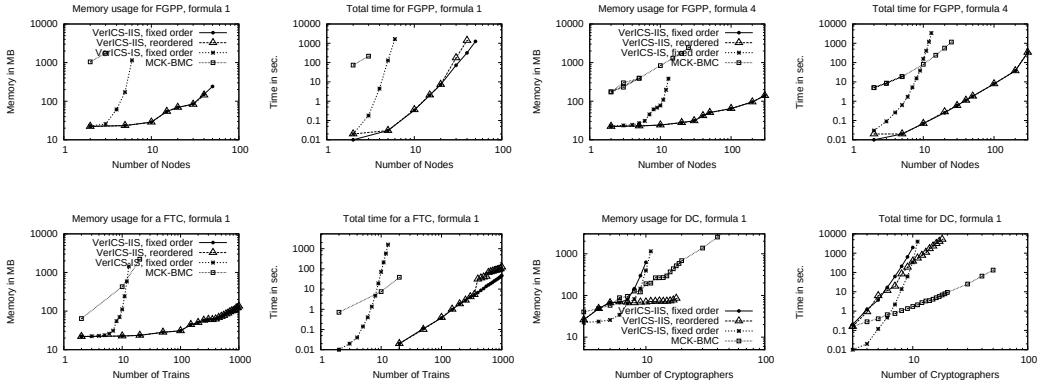
Algorithm 2 BMC algorithm

```
1:  Reach := {ι}, New := {ι}
2:  while New ≠ ∅ do
3:     Next := New_⇝
4:     if ι ∈ [[M|_Reach, φ]] then
5:        return true
6:     end if
7:     New := Next \ Reach
8:     Reach := Reach ∪ New
9:  end while
10: return false
```

BMC algorithm. Given a model M and an ELTLK formula φ, the algorithm checks if there exists a path or run initialised in the initial state ι along which φ holds. The algorithm starts with the set *Reach* of reachable states that initially contains only the state ι. With each iteration the verified formula is checked (line 4), and the set *Reach* is extended with new states reachable in one step from old states in *Reach* (line 8). The algorithm operates on submodels $M|_{Reach}$ generated by the set *Reach* (i.e., models restricted to contain only the states of *Reach*) to check if the initial state ι is in the set of states from which there is a path or run on which φ holds. The loop terminates if there is such a path or run in the obtained submodel, and the algorithm returns *true* (line 5). The search continues until no new states can be reached from the states in *Reach*. When we obtain the complete set of the reachable states, and a path or run from the initial state on which φ holds could not be found in any of the obtained submodels, the algorithm terminates returning *false*.

4 Experimental Evaluation

We have considered three scalable systems to evaluate the efficiency of our BDD-based BMC for LTLK: Faulty Generic Pipeline Paradigm (FGPP), Faulty Train Controller (FTC), and Dining Cryptographers (DC). The systems were modelled using two semantics, and the benchmarks were performed with several formulae. For the detailed descriptions of the benchmarks see [15]. Our method was implemented as two separate prototype modules of VerICS [11] for IS and IIS semantics (named VerICS-IS and VerICS-IIS, respectively). We have also compared our results with those obtained using MCK [6], another model checker for multi-agent systems, implementing standard IS semantics. Results for some of the performed benchmarks are included in the figures below.



Comparing algorithms for IS, in most cases MCK is better than VerICS-IS, but remains close when looking at the orders of magnitude. The reason for better performance of MCK may come from the fact that it is based on the translation to SAT, and SAT-based BMC does not need to store the whole examined part of the state space.

For most of the considered benchmarks the VerICS-IIS method is superior to the two IS approaches: MCK and VerICS-IS, sometimes even by several orders of magnitude. This can be observed especially in the case of FTC. However, in the case of FGPP and formula 3 with no epistemic modalities, MCK proved to be more efficient, but for the formula 4 containing the K operator, VerICS-IIS was superior. This can be justified by the fact that introducing epistemic modalities partitions the ELTL verification task into several smaller ones.

In the case of IIS, the reordering of the BDD variables does not cause any significant change of the performance in the case of FGPP and FTC, but for DC it reduces the memory consumption. Therefore, for IIS the fixed interleaving order we used can often be considered optimal. The penalty in the verification time to reorder the variables, in favour of reducing memory consumption, is also not significant and can be worth the tradeoff. However, in the case of IS the performance did not change, thus we include only the results for the fixed order of the variables for VerICS-IS.

It is important to note that from our comparison of [17] it follows that in the case of IIS, the general performance of BDD-based approach is superior to the SAT-based one. Therefore, we can conclude now that BMC for LTLK is less efficient for IS when comparing with IIS. This could be explained by the different structure of the state space, which for IS is more dense, i.e., more states are explored at every iteration of the BMC algorithm. The case of DC shows that this factor can be more important than the lengths of the counterexamples, which can be shorter for IS, or may even be of constant length when scaling the system.

The experimental results show that the approach based on the interleaved interpreted systems can greatly improve the practical applicability of the bounded model checking method. Although, we have tested only properties of LTLK, we suspect this to also be true for similar specification formalisms, e.g., CTLK.

5 Final Remarks

In this paper, we have presented a BDD-based method for bounded model checking of LTLK over models of multi-agent systems. We evaluated the methodology in two different settings: interleaved interpreted systems and synchronous interpreted systems. The results are preliminary and the comparison is by no means complete. It ignores the fact that for some formulae the choice of the semantics influences the existence of a witness in the model.

References

1 A. Biere, A. Cimatti, E. Clarke, and Y. Zhu. Symbolic model checking without BDDs. In *Proc. of the 5th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'99)*, volume 1579 of *LNCS*, pages 193–207. Springer-Verlag, 1999.

2 R. Bordini, M. Fisher, C. Pardavila, W. Visser, and M. Wooldridge. Model checking multi-agent programs with CASP. In *Proc. of CAV'03*, volume 2725 of *LNCS*, pages 110–113. Springer-Verlag, 2003.

3 E. Clarke, O. Grumberg, and K. Hamaguchi. Another look at LTL model checking. In *Proc. of CAV'94*, volume 818 of *LNCS*, pages 415–427. Springer-Verlag, 1994.

4 E. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, 1999.

5 R. Fagin, J. Y. Halpern, Y. Moses, and M. Vardi. *Reasoning about Knowledge*. MIT Press, Cambridge, 1995.

6 P. Gammie and R. van der Meyden. MCK: Model checking the logic of knowledge. In *Proc. of the 16th Int. Conf. on Computer Aided Verification (CAV'04)*, volume 3114 of *LNCS*, pages 479–483. Springer-Verlag, 2004.

7 W. van der Hoek and M. Wooldridge. Model checking knowledge and time. In *Proc. of SPIN'02*, volume 2318 of *LNCS*, pages 95–111. Springer-Verlag, 2002.

8 X. Huang, C. Luo, and R. van der Meyden. Improved bounded model checking for a fair branching-time temporal epistemic logic. In *Proc. of 6th Int. Workshop on Model Checking and Artificial Intelligence 2010*, LNAI. Springer, 2011.

9 A. V. Jones and A. Lomuscio. Distributed bdd-based bmc for the verification of multi-agent systems. In *Proc. of AAMAS'10*, pages 675–682, 2010.

10 M. Kacprzak, A. Lomuscio, and W. Penczek. From bounded to unbounded model checking for temporal epistemic logic. *Fundam. Inform.*, 63(2-3):221–240, 2004.

11 M. Kacprzak, Wojciech Nabiałek, A. Niewiadomski, W. Penczek, A. Półrola, M. Szreter, B. Woźna, and A. Zbrzezny. Verics 2006 - a model checker for real-time and multi-agent systems. In *Proc. of the Int. Workshop on Concurrency, Specification and Programming (CS&P'07)*, pages 345–356. Warsaw University, 2007.

12 A. Lomuscio, W. Penczek, and H. Qu. Partial order reduction for model checking interleaved multi-agent systems. In *Proc. of AAMAS'10*, pages 659–666, 2010.

13 R. van der Mayden and K. Su. Symbolic model checking the knowledge of the dining cryptographers. In *Proc. of CSFW-17*, pages 280–291. IEEE Computer Society, June 2004.

14 R. van der Meyden and N. V. Shilov. Model checking knowledge and time in systems with perfect recall. In *Proc. of FSTTCS'99*, volume 1738 of *LNCS*, pages 432–445. Springer-Verlag, 1999.

15 A. Męski, W. Penczek, and M. Szreter. Bounded model checking linear time and knowledge using decision diagrams. In *Proc. of CS&P'11*, pages 363–375, 2011.

16 A. Męski, W. Penczek, and M. Szreter. BDD-based bounded model checking for LTLK over two variants of interpreted systems. In *Proc. of LAM'12*, pages 35–49, 2012.

17 A. Męski, W. Penczek, M. Szreter, B. Woźna-Szcześniak, and A. Zbrzezny. Bounded model checking for knowledge and linear time. In *Proc. of AAMAS'12*. IFAAMAS Press, 2012.

18 W. Penczek and A. Lomuscio. Verifying epistemic properties of multi-agent systems via bounded model checking. *Fundam. Inform.*, 55(2):167–185, 2003.

19 W. Penczek, B. Woźna-Szcześniak, and A. Zbrzezny. Towards SAT-based BMC for LTLK over interleaved interpreted systems. In *Proc. of CS&P'11*, pages 565–576, 2011.

20 F. Raimondi and A. Lomuscio. Automatic verification of multi-agent systems by model checking via OBDDs. *Journal of Applied Logic*, 5(2):235–251, 2007.

21 K. Su, A. Sattar, and X. Luo. Model checking temporal logics of knowledge via OBDDs. *The Computer Journal*, 50(4):403–420, 2007.