BDD-based Bounded Model Checking for LTLK over Two Variants of Interpreted Systems*

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

[1] Institute of Computer Science, PAS, Ordona 21, 01-237 Warsaw, Poland
{meski,penczek,mszreter}@ipipan.waw.pl
[2] University of Łódź FMCS, Banacha 22, 90-238 Łódź, Poland
[3] University of Natural Sciences and Humanities, Institute of Informatics, 3 Maja 54, 08-110 Siedlce, Poland

Abstract We present a novel approach to verification of multi-agent systems by bounded model checking for Linear Time Temporal Logic extended with the epistemic component (LTLK). The systems are modelled by two variants of interpreted systems: standard and interleaved ones. Our method is based on binary decision diagrams (BDD). We describe the algorithm and provide its experimental evaluation together with the comparison with another tool. This allows to draw some conclusions which semantics is preferable for bounded model checking LTLK properties of multi-agent systems.

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 [5], 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 [2,7,8,13,15,18,22,23].

When the verification is performed by searching directly through the state space of the MAS, its size is likely to grow exponentially with the number of agents, which is known as the *state-space explosion* problem. 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) [20,13] or with symbolic techniques based on *binary decision diagrams* (BDDs) [11].

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

* Partly supported by National Science Centre under the grant No. 2011/01/B/ST6/05317 and 2011/01/B/ST6/01477.

the epistemic component (LTLK, also called CKL_n [8]). The systems are modeled by two variants of Interpreted Systems: standard (IS) [6] and interleaved ones (IIS) [14]. IIS restrict IS by enforcing asynchronous semantics. This does not reduce the expressive power of IS, but modifies this popular modelling approach 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 [20,10] or on BDDs [11]. A SAT-based verification method for the LTLK properties of MAS, modeled by IIS, was put forward in [21]. Our technical report [16] 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. Section 2 provides the basic definitions and notations for LTLK and IS. Our BDD-based BMC method is described in Section 3. The last two sections contain the discussion of an experimental evaluation of the approach and the final remarks.

2 Preliminaries

In this section we introduce the basic definitions used in the paper. In particular, we define the semantics of interpreted systems, as well as the syntax and the semantics of LTLK.

2.1 Formalisms for Modelling Multi-Agent Systems

Interpreted Systems The semantics of *interpreted systems* provides a setting to reason about MAS by means of specifications based on knowledge and linear or branching time. We report here the basic setting as popularised in [6]. 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

[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.

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 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 $\rho = g_0 a_0 g_1 a_1 g_2 \ldots$ 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 (Interpreted Systems). *Given a set of propositions $\mathcal{PV}$ such that $\{true, false\} \subseteq \mathcal{PV}$, an interpreted system (IS), also referred to as a model, is a tuple $M = (G, \iota, \Pi, \{\sim_i\}_{i \in \mathcal{A}}, \mathcal{V})$, 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, $\Pi = \bigcup_{g \in G} \Pi(g)$ is the set of all the interleaved paths and runs originating from all the states in G, and $V : G \to 2^{\mathcal{PV}}$ is a valuation function.*

By Π^ω we denote the set of all the paths of Π.

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 $a = (a_1, \ldots, a_n) \in Act$, and an action $\alpha \in \bigcup_{i \in \mathcal{A}} Act_i \setminus \{\epsilon_1, \ldots, \epsilon_n\}$ such that: $a_i = \alpha$ for all $i \in Agent(\alpha)$, and $a_i = \epsilon_i$ for all $i \in \mathcal{A} \setminus Agent(\alpha)$. Similar to blocking synchronisation in automata, the above insists on all the agents performing the same non-epsilon action in a global evolution; additionally, note that if an agent has the action being performed in its repertoire it must be performed for the global evolution to be allowed. This assumes local protocols

are defined in such a way to permit this; if a local protocol does not allow this, the local action cannot be performed and therefore the global evolution does not comply with the above definition of interleaving.

2.2 Syntax and Semantics of LTLK

Combinations of linear time with knowledge have long been used in the analysis of temporal epistemic properties of systems [6]. We now recall the basic definitions here and adapt them to our purposes when needed.

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 *until* and *release* respectively, X is the next step operator. The epistemic operators K_q, $\mathrm{D}_\Gamma, \mathrm{E}_\Gamma$, and C_Γ 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 ones.

Typically, the semantics of LTLK is defined over paths of a model M only, whereas our semantics exploits paths and runs. This semantics can be conveniently applied also to submodels (to be defined later) in order to verify efficiently the existential fragment of LTLK over paths and runs.

Definition 3 (Semantics). *Given a model $M = (G, \iota, \Pi, \{\sim_q\}_{q\in\mathcal{A}}, \mathcal{V})$, where $\mathcal{V}(s)$ is the set of propositions that hold at s, let $\rho(i)$ 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* $M, \rho[i] \not\models \varphi$,
- $M, \rho[i] \models \varphi_1 \wedge \varphi_2$ *iff* $M, \rho[i] \models \varphi_1$ *and* $M, \rho[i] \models \varphi_2$,
- $M, \rho[i] \models \varphi_1 \vee \varphi_2$ *iff* $M, \rho[i] \models \varphi_1$ *or* $M, \rho[i] \models \varphi_2$,
- $M, \rho[i] \models \mathrm{X}\varphi$ *iff* $length(\rho) > i$ *and* $M, \rho[i+1] \models \varphi$;
- $M, \rho[i] \models \varphi_1 \mathrm{U} \varphi_2$ *iff* $(\exists k \geq i)[M, \rho[k] \models \varphi_2$ *and* $(\forall i \leq j < k)\ M, \rho[j] \models \varphi_1]$,
- $M, \rho[i] \models \varphi_1 \mathrm{R} \varphi_2$ *iff* $[(\rho \in \Pi^\omega(\iota)$ *and* $(\forall k \geq i)\ M, \rho[k] \models \varphi_2]$ *or* $(\exists k \geq i)[M, \rho[k] \models \varphi_1$ *and* $(\forall i \leq j \leq k)\ M, \rho[j] \models \varphi_2]$,
- $M, \rho[i] \models \mathrm{K}_q\varphi$ *iff* $(\forall \rho' \in \Pi^\omega(\iota))(\forall k \geq 0)[\rho'(k) \sim_q \rho(i)$ *implies* $M, \rho'[k] \models \varphi]$,
- $M, \rho[i] \models \overline{\mathrm{K}}_q\varphi$ *iff* $(\exists \rho' \in \Pi(\iota))(\exists k \geq 0)[\rho'(k) \sim_q \rho(i)$ *and* $M, \rho'[k] \models \varphi]$,
- $M, \rho[i] \models \mathrm{Y}_\Gamma\varphi$ *iff* $(\forall \rho' \in \Pi^\omega(\iota))(\forall k \geq 0)[\rho'(k) \sim_\Gamma^{\mathrm{Y}} \rho(i)$ *implies* $M, \rho'[k] \models \varphi]$,
- $M, \rho[i] \models \overline{\mathrm{Y}}_\Gamma\varphi$ *iff* $(\exists \rho' \in \Pi(\iota))(\exists k \geq 0)[\rho'(k) \sim_\Gamma^{\mathrm{Y}} \rho(i)$ *and* $M, \rho'[k] \models \varphi]$, *where* $\mathrm{Y} \in \{\mathrm{D}, \mathrm{E}, \mathrm{C}\}$.

Let $g \in G$ and φ be an LTLK formula. We use the following notations:

– $M, g \models \varphi$ iff $M, \rho[0] \models \varphi$ for all the paths $\rho \in \Pi^{\omega}(g)$;
– $M \models \varphi$ iff $M, \iota \models \varphi$;
– $Props(\varphi)$ is the set of atomic propositions appearing in φ.

LTL is the sublogic of LTLK which consists only of the formulae built without the epistemic operators. ELTLK is the existential fragment of LTLK, defined by the following grammar:

$$\varphi ::= p \mid \neg p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \mathrm{X}\varphi \mid \varphi \mathrm{U} \varphi \mid \varphi \mathrm{R} \varphi \mid \overline{\mathrm{K}}_q \varphi \mid \overline{\mathrm{E}}_\Gamma \varphi \mid \overline{\mathrm{D}}_\Gamma \varphi \mid \overline{\mathrm{C}}_\Gamma \varphi.$$

Moreover, an ELTLK formula φ holds in the model M, denoted $M \models_\exists \varphi$, iff $M, \rho[0] \models \varphi$ for some path or run $\rho \in \Pi(\iota)$. The intuition behind this definition is that ELTLK is obtained only by restricting the syntax of the epistemic operators while the temporal ones remain the same. We get the existential version of these operators by the change from the universal quantification over the paths ($\models$) to the existential quantification ($\models_\exists$) over the paths and the runs in the definition of the validity in the model M. Notice that this change is only necessary when φ contains a temporal operator, which is not nested in an epistemic operator.

Our semantics meets two important properties. Firstly, for LTL the definition of validity in a model M uses paths only. Secondly, if we replace each Π with Π^{ω}, the semantics does not change as our models have total transition relations (each run is a prefix of some path). The semantics applied to submodels of M does not have the above property, but it preserves ELTLK over M, which is shown in Lemma 1.

3 BDD-based BMC for ELTLK

In this section we show how to perform BMC of ELTLK using BDDs [5] by combining the standard approach for ELTL [4] with the method for the epistemic operators [22] in a similar manner to the solution for CTL* of [5].

Let $\mathcal{PV}$ be a set of propositions. For an ELTLK formula φ we define inductively the *number* $\gamma(\varphi)$ *of nested epistemic operators* in the formula:

– if $\varphi = p$, where $p \in \mathcal{PV}$, then $\gamma(\varphi) = 0$,
– if $\varphi = \odot\varphi'$ and $\odot \in \{\neg, \mathrm{X}\}$, then $\gamma(\varphi) = \gamma(\varphi')$,
– if $\varphi = \varphi' \odot \varphi''$ and $\odot \in \{\wedge, \vee, \mathrm{U}, \mathrm{R}\}$, then $\gamma(\varphi) = \gamma(\varphi') + \gamma(\varphi'')$,
– if $\varphi = \mathrm{Y}\varphi'$ and $\mathrm{Y} \in \{\overline{\mathrm{K}}_q, \overline{\mathrm{E}}_\Gamma, \overline{\mathrm{D}}_\Gamma, \overline{\mathrm{C}}_\Gamma\}$, then $\gamma(\varphi) = \gamma(\varphi') + 1$.

Let $\mathrm{Y} \in \{\overline{\mathrm{K}}_q, \overline{\mathrm{E}}_\Gamma, \overline{\mathrm{D}}_\Gamma, \overline{\mathrm{C}}_\Gamma\}$. If $\varphi = \mathrm{Y}\psi$ is an ELTLK formula, by $sub(\varphi)$ we denote the formula ψ nested in the epistemic operator Y. Moreover, for an arbitrary ELTLK formula φ we define inductively the set $\mathcal{Y}(\varphi)$ of its subformulae in the form $\mathrm{Y}\psi$:

– if $\varphi = p$, where $p \in \mathcal{PV}$, then $\mathcal{Y}(\varphi) = \emptyset$,
– if $\varphi = \odot\varphi'$ and $\odot \in \{\neg, \mathrm{X}\}$, then $\mathcal{Y}(\varphi) = \mathcal{Y}(\varphi')$,
– if $\varphi = \varphi' \odot \varphi''$ and $\odot \in \{\wedge, \vee, \mathrm{U}, \mathrm{R}\}$, then $\mathcal{Y}(\varphi) = \mathcal{Y}(\varphi') \cup \mathcal{Y}(\varphi'')$,
– if $\varphi = \mathrm{Y}\varphi'$ and $\mathrm{Y} \in \{\overline{\mathrm{K}}_q, \overline{\mathrm{E}}_\Gamma, \overline{\mathrm{D}}_\Gamma, \overline{\mathrm{C}}_\Gamma\}$, then $\mathcal{Y}(\varphi) = \mathcal{Y}(\varphi') \cup \{\varphi\}$.

Definition 4. *Let* $M = (G, \iota, \Pi, \{\sim_q\}_{q\in\mathcal{A}}, \mathcal{V})$ *and* $U \subseteq G$ *with* $\iota \in U$*. The* submodel *generated by* U *is a tuple* $M|_U = (U, \iota, \Pi', \{\sim'_q\}_{q\in\mathcal{A}}, \mathcal{V}')$*, where:* $\sim'_q = \sim_q \cap U^2$ *for each* $q \in \mathcal{A}$*,* $\mathcal{V}' = \mathcal{V} \cap U^2$*, and* Π' *is the set of the paths and runs of* M *having all the states in* U*, formally,* $\Pi' = \{\rho \in \Pi \mid (\forall 0 \leq i \trianglelefteq_\rho length(\rho))\ \rho(i) \in U\}$*.*

For ELTLK formulae φ, ψ, and ψ', by $\varphi[\psi \leftarrow \psi']$ we denote the formula φ in which every occurrence of ψ is replaced with ψ'. Let $M = (G, \iota, \Pi, \{\sim_q\}_{q\in\mathcal{A}}, \mathcal{V})$ be a model, then by $\mathcal{V}_M$ we understand the valuation function $\mathcal{V}$ of the model M, and by $G_R \subseteq G$ the set of its reachable states. Moreover, we define $[\![M, \varphi]\!] = \{g \in G_R \mid M, g \models^\exists \varphi\}$.

Reducing ELTLK to ELTL. Given a model $M = (G, \iota, \Pi, \{\sim_q\}_{q\in\mathcal{A}}, \mathcal{V})$, and an ELTLK formula φ, Algorithm 1 is used to compute the set $[\![M, \varphi]\!]$, 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 [4] and [22], respectively). In order to obtain this set, we construct a new model M_c together with an ELTL formula φ_c, as described in Algorithm 1, 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. We begin by choosing some epistemic subformula ψ of φ_c, which consists of exactly one epistemic operator, and process it in two stages. First, we modify the valuation function of M_c 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. 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 we label every state initialising some path or run along which the modified simple epistemic formula ψ holds with a new variable p_ψ. Similarly to the previous stage, we replace every occurrence of ψ in φ_c with p_ψ. 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, 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). The correctness of the substitution used in Algorithm 1 is stated by the following proposition:

Proposition 1. *Let* $M = (G, \iota, \Pi, \{\sim_q\}_{q\in\mathcal{A}}, \mathcal{V})$ *be a model,* φ *an* ELTLK *formula, and* $\rho \in \Pi$ *some path or run with an evaluation position such that* $m \trianglelefteq_\rho length(\rho)$*. We define* $p \in \mathcal{PV}$ *such that* $M, \rho'[m'] \models p$ *iff* $M, \rho'[m'] \models \varphi$ *for all* $\rho' \in \Pi(\iota)$*, where* $m' \trianglelefteq_{\rho'} length(\rho')$*. Then,* $M, \rho[m] \models \mathrm{Y}\varphi$ *iff* $M, \rho[m] \models \mathrm{Y}p$*, where* $\mathrm{Y} \in \{\overline{\mathrm{K}}_q, \overline{\mathrm{E}}_\Gamma, \overline{\mathrm{D}}_\Gamma, \overline{\mathrm{C}}_\Gamma\}$*, and* $q \in \mathcal{A}$*,* $\Gamma \subseteq \mathcal{A}$*.*

Proof. Straightforward from the semantics of ELTLK.

Algorithm 1. Computation of $[\![M, \varphi]\!]$

1: $M_c := M$, $\varphi_c := \varphi$
2: **while** $\gamma(\varphi_c) \neq 0$ **do**
3: pick $\psi \in \mathcal{Y}(\varphi_c)$ such that $\gamma(\psi) = 1$
4: **for all** $g \in [\![M_c, sub(\psi)]\!]$ **do**
5: $\mathcal{V}_{M_c}(g) := \mathcal{V}_{M_c}(g) \cup \{p_{sub(\psi)}\}$
6: **end for**
7: $\psi := \psi[sub(\psi) \leftarrow p_{sub(\psi)}]$
8: **for all** $g \in [\![M_c, \psi]\!]$ **do**
9: $\mathcal{V}_{M_c}(g) := \mathcal{V}_{M_c}(g) \cup \{p_\psi\}$
10: **end for**
11: $\varphi_c := \varphi_c[\psi \leftarrow p_\psi]$
12: **end while**
13: **return** $[\![M_c, \varphi_c]\!]$

Algorithm 2. BMC algorithm

1: $Reach := \{\iota\}$,$New := \{\iota\}$
2: **while** $New \neq \emptyset$ **do**
3: $Next := New_{\rightsquigarrow}$
4: **if** $\iota \in [\![M|_{Reach}, \varphi]\!]$ **then**
5: **return** $true$
6: **end if**
7: $New := Next \setminus Reach$
8: $Reach := Reach \cup New$
9: **end while**
10: **return** $false$

BMC Algorithm. To perform bounded model checking of an ELTLK formula, we use Algorithm 2. Given a model M and an ELTLK formula φ, the algorithm checks if there exists a path or run initialised in ι on which φ holds, i.e., if $M, \iota \models^{\exists} \varphi$. For any $X \subseteq G$ by $X_{\rightsquigarrow} \stackrel{def}{=} \{g' \in G \mid (\exists g \in X)(\exists \rho \in \Pi(g))\ g' = \rho(1)\}$ we define the set of the immediate successors of all the states in X. 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 (line 8). The algorithm operates on submodels $M|_{Reach}$ generated by the set *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 set the of 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 $false$.

The correctness of the results obtained by the bounded model checking algorithm is formulated by the following lemma:

Lemma 1. *Let $M = (G, \iota, \Pi, \{\sim_q\}_{q \in \mathcal{A}}, \mathcal{V})$ be a model, φ an* ELTLK *formula, and $\rho \in \Pi$ a path or run with an evaluation position m such that $m \trianglelefteq_\rho length(\rho)$. Then, $M, \rho[m] \models \varphi$ iff exists $G' \subseteq G$ such that $\iota \in G'$, and $M|_{G'}, \rho[m] \models \varphi$.*

Proof. "$\Rightarrow$" This way the proof is obvious as we simply take $G' = G$.
"$\Leftarrow$" This way the proof is more involved. It is by induction on the length of a formula φ. The base case is straightforward, as the lemma follows directly for the propositional variables and their negations. Assume, the statement holds for all the proper subformulae of φ. Let $G' \subseteq G$ be a set of states such that $M|_{G'}$ contains ρ, and (*) $M|_{G'}, \rho[i] \models \varphi$, where $i \in \mathbb{N}$.

1. Let $\varphi = \alpha \vee \beta$. By the semantics and the assumption (*), $M|_{G'}, \rho[i] \models \alpha$ or $M|_{G'}, \rho[i] \models \beta$. Using the induction hypothesis and the definition of

submodel (Def. 4), ρ exists also in the model M, and $M, \rho[i] \models \alpha$ or $M, \rho[i] \models \beta$, thus $M, \rho[i] \models \alpha \vee \beta$.

2. Let $\varphi = \alpha \wedge \beta$. By the semantics and the assumption (*), $M|_{G'}, \rho[i] \models \alpha$ and $M|_{G'}, \rho[i] \models \beta$. Using the induction hypothesis and the definition of submodel, ρ exists also in the model M. Therefore, $M, \rho[i] \models \alpha$ and $M, \rho[i] \models \beta$, thus $M, \rho[i] \models \alpha \wedge \beta$.
3. Let $\varphi = \mathrm{X}\alpha$. By the semantics and the assumption (*), $length(\rho) > i$, and $M|_{G'}, \rho[i+1] \models \alpha$. Using the induction hypothesis and the definition of submodel, we get that ρ exists also in M, and $M, \rho[i+1] \models \alpha$, therefore $M, \rho[i] \models \mathrm{X}\alpha$.
4. Let $\varphi = \alpha \mathrm{U} \beta$. By the semantics and the assumption (*), there exists $k \geq i$, such that $M|_{G'}, \rho[k] \models \beta$, and $M|_{G'}, \rho[j] \models \alpha$, for all $i \leq j < k$. Using the induction hypothesis and the definition of submodel, we get that ρ exists also in M. Therefore, from $M, \rho[k] \models \beta$, and $M, \rho[j] \models \alpha$ for all $i \leq j < k$, it follows that $M, \rho[i] \models \alpha \mathrm{U} \beta$.
5. Let $\varphi = \alpha \mathrm{R} \beta$. By the semantics and the assumption (*) we have one or both of the following cases:
   (a) ρ is a path of $M|_{G'}$, and $M|_{G'}, \rho[k] \models \beta$ for all $k \geq i$, then from the definition of submodel, ρ exists also in M, and $\rho \in \Pi^{\omega}$. Using the induction hypothesis, we have that $M, \rho[k] \models \beta$ for all $k \geq i$. Therefore, it follows that $M, \rho[i] \models \alpha \mathrm{R} \beta$.
   (b) There exists $k \geq i$ such that $M|_{G'}, \rho[k] \models \alpha$, and $M|_{G'}, \rho[j] \models \beta$ for all $i \leq j \leq k$. From the definition of submodel, ρ also exists in M, and using the induction hypothesis we get that $M, \rho[k] \models \alpha$, and $M, \rho[j] \models \beta$ for all $i \leq j \leq k$. Thus, $M, \rho[i] \models \alpha \mathrm{R} \beta$.
6. Let $q \in \mathcal{A}$, and $\varphi = \overline{\mathrm{K}}_q \alpha$. By the semantics and the assumption (*), there exists such a path or run ρ' in $M|_{G'}$ that $\rho'(k) \sim_q \rho(i)$ for some $k \geq 0$, and $M|_{G'}, \rho'[k] \models \alpha$. From the definition of submodel, ρ and ρ' also exist in M. Using the induction hypothesis, we get that $M, \rho'[k] \models \alpha$ and $\rho'(k) \sim_q \rho(i)$. Thus, $M, \rho[i] \models \overline{\mathrm{K}}_q \alpha$.
7. Let $\Gamma \subseteq \mathcal{A}$, and $\varphi = \overline{\mathrm{Y}}_{\Gamma} \alpha$, where $\mathrm{Y} \in \{\mathrm{D}, \mathrm{E}, \mathrm{C}\}$. By the semantics and the assumption (*), there exists such a path or run ρ' in $M|_{G'}$ that $\rho'(k) \sim_{\Gamma}^{\mathrm{Y}} \rho(i)$ for some $k \geq 0$, and $M|_{G'}, \rho'[k] \models \alpha$. From the definition of submodel, ρ and ρ' also exist in M. Using the induction hypothesis, we get that $M, \rho'[k] \models \alpha$ and $\rho'(k) \sim_{\Gamma}^{\mathrm{Y}} \rho(i)$. Thus, $M, \rho[i] \models \overline{\mathrm{Y}}_{\Gamma} \alpha$.

Model Checking ELTL. To compute the sets of states corresponding to the ELTL formulae, needed in Algorithm 1, we use the method described in [4] and based on checking the non-emptiness of Büchi automata. Given a model M and an ELTL formula φ, we begin with constructing the tableau for φ (as described in [4]), that is then combined with M to obtain their product, which contains these paths of M where φ potentially holds. Next, the product is verified in terms of the CTL model checking of EG$true$ formula under fairness constraints. Those constraints, corresponding to sets of states, allow to choose only the paths of the model, along which at least one state in each set representing fairness constraints

appears in a cycle. In case of ELTL model checking, fairness guarantees that $\varphi \mathrm{U} \psi$ really holds, i.e., eliminates the paths where φ holds continuously, but ψ never holds. Finally, we choose only these reachable states of the product that belong to some particular set of states computed for the formula. The corresponding states of the verified system that are in this set, comprise the set $[\![M, \varphi]\!]$, i.e., the reachable states where the verified formula holds. As we are unable to include more details (due to the page limit), we refer the reader to [4].

The method described above has some limitations when used for BMC, where it is preferable to detect counterexamples using not only the paths but also the runs of the submodel. As totality of the transition relation of the verified model is assumed, counterexamples are found only along the paths of the model. However, this remains correct even if the final submodel only has the total transition relation: in the worst case the detection of the counterexample is delayed to the last iteration, i.e., when all the reachable states are computed. Nonetheless, this should not keep us from assessing the potential efficiency of our approach.

Model Checking Epistemic Modalities. In order to verify the formulae of 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 described in [22]. The procedures simply follow from the semantics of ELTLK. The algorithm for $\overline{\mathrm{C}}_\Gamma$ involves a fix point computation, whereas for the remaining operators the algorithms are based on simple non-iterative computations.

4 Experimental Results

In this section we consider three scalable systems which we use to evaluate the efficiency of our BDD-based BMC for LTLK over two variants of Interpreted Systems: IS and IIS. We also compare our results with the ones obtained using MCK. The tool MCK[2] enables fair comparisons for IS semantics, as according to the manual it supports SAT-based BMC for CTL*K. Unfortunately, no theory behind this implementation has ever been published. The paper [10], which describes SAT-based BMC for CTLK, does not discuss how this approach can be extended to CTL*K.

The tests have been performed on a computer fitted with Intel Xeon 2 GHz processor and 4 GB of RAM, running Linux 2.6. Our methods are implemented with reordering, and with the fixed interleaving order of the BDD variables. The reordering is performed by the Rudell's sifting algorithm available in CUDD library, used for manipulating BDDs.

The specifications for the described benchmarks are given in the universal form, for which we verify the corresponding counterexample formula, i.e., the formula which is negated and interpreted existentially. Moreover, for every specification given, there exists a counterexample. With $i(n)$ and $i^{\mathrm{I}}(n)$ we denote the number of iterations needed by our algorithms for IS and IIS, respectively, to find

[2] `http://cgi.cse.unsw.edu.au/~mck/mcks/docDownload/manual`, version 0.5.1 was used as the newer 1.0.0 is provided only for 32-bit machines.

the counterexample, where n is the scaling parameter. The detailed descriptions of our experiments together with the specifications for the systems used, can be found at the web page of VerICS.[3] The memory and the time consumption are shown in the respective figures as the functions of the scaling parameter for each benchmark. Note that the figures are presented in a logarithmic scale.

4.1 Benchmarks

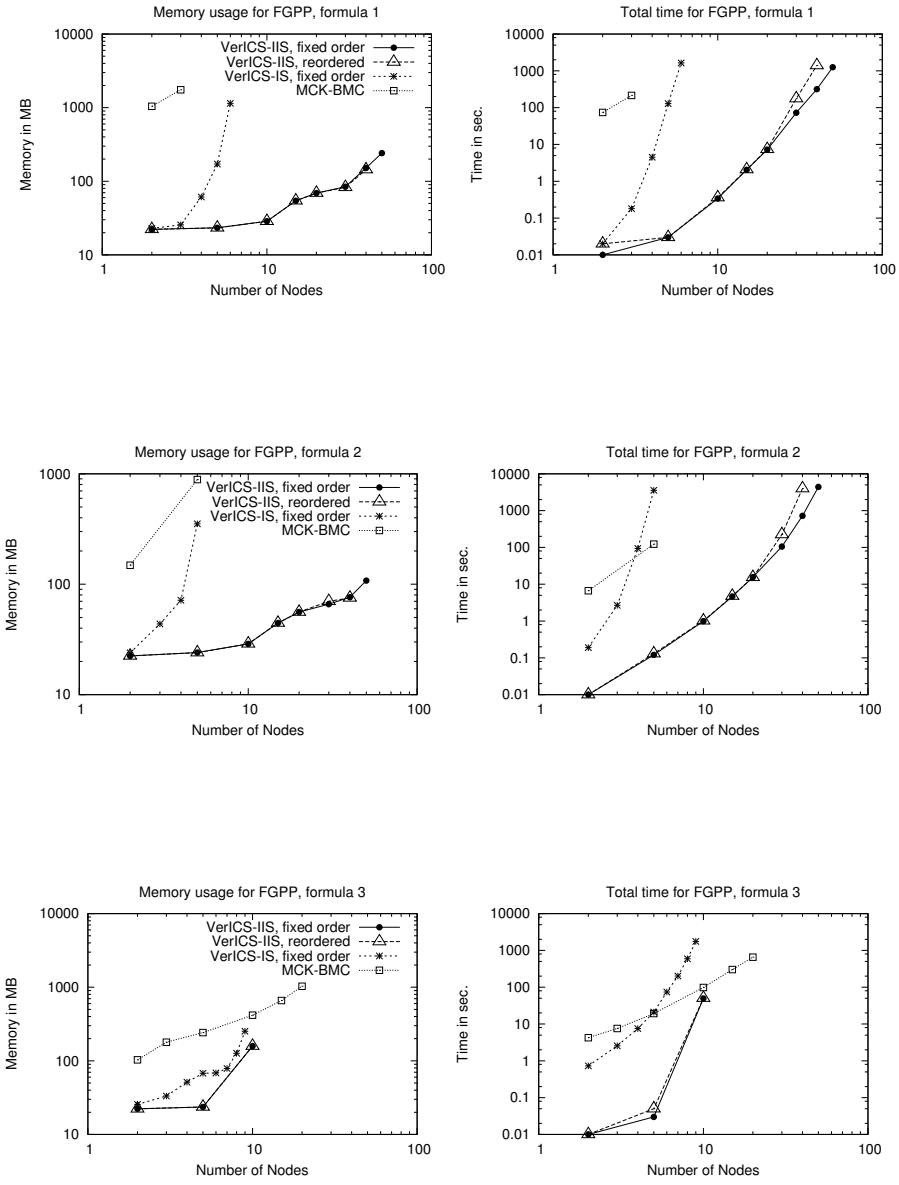
Faulty Generic Pipeline Paradigm (FGPP) (adapted from [19]) consists of Producer, Consumer, and a chain of n intermediate Nodes transmitting data, together with a chain of n Alarms enabled when some error occurs. We consider the following specifications:
$\varphi_1 = \mathrm{G}(ProdSend \rightarrow \mathrm{K}_C\mathrm{K}_P ConsReady))$, $\varphi_2 = \mathrm{G}(Problem_n \rightarrow (\mathrm{F}(Repair_n) \vee \mathrm{G}(Alarm_n Send)))$, $\varphi_3 = \bigwedge_{i=1}^{n} \mathrm{G}(Problem_i \rightarrow (\mathrm{F}(Repair_i) \vee \mathrm{G}(Alarm_i Send)))$, and $\varphi_4 = \bigwedge_{i=1}^{n} \mathrm{G}(\mathrm{K}_P(Problem_i \rightarrow (\mathrm{F}(Repair_i) \vee \mathrm{G}(Alarm_i Send))))$. The formula φ_1 ($i(n) = i^{\mathtt{I}}(n) = 2n + 3$) states that if Producer produces a commodity, then Consumer knows that Producer does not know that Consumer has the commodity. The formula φ_2 ($i(n) = i^{\mathtt{I}}(n) = 2n + 4$) expresses that each time a problem occurs at node n, then either it is repaired or the alarm of node n rings. The formula φ_3 ($i^{\mathtt{I}}(n) = 8$, $i(n) = 2n + 4$) expresses that each time a problem occurs on a node, then either it is repaired or the alarm rings. The formula, φ_4 ($i^{\mathtt{I}}(n) = 5$, $i(n) = 8$) expresses that Producer knows that each time a problem occurs on a node, then either it is repaired or the alarm rings.

A faulty train controller system (FTC) (adapted from [9]) consists of a controller and n trains (for $n \geq 2$), one of which is dysfunctional. We consider the following specifications: $\varphi_1 = \mathrm{G}(InTunnel_1 \rightarrow \mathrm{K}_{Train_1}(\bigwedge_{i=2}^{n} \neg InTunnel_i))$, and $\varphi_2 = \mathrm{G}(\mathrm{K}_{Train_1} \bigwedge_{i=1,j=2,i<j}^{n} \neg(InTunnel_i \wedge InTunnel_j)$. The formula φ_1 ($i^{\mathtt{I}}(n) = 5$, $i(n) = 8$) expresses that whenever a train is in the tunnel, it knows that the other trains are not. The formula φ_2 ($i^{\mathtt{I}}(n) = 5$, $i(n) = 7$) represents that trains are aware of the fact that they have exclusive access to the tunnel.

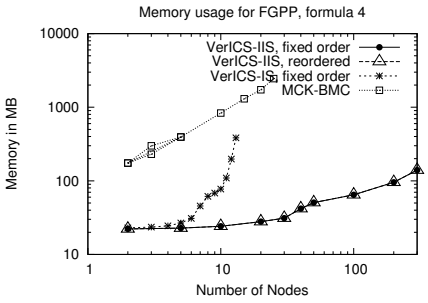
Dining Cryptographers (DC) [3] is a scalable anonymity protocol, which has been formalised and analysed in many works, e.g., [12,15]. We consider the following specifications: $\varphi_1 = \mathrm{G}(odd \wedge \neg paid_1 \rightarrow \bigvee_{i=2}^{n} \mathrm{K}_1(paid_i))$, $\varphi_2 = \mathrm{G}(\neg paid_1 \rightarrow \mathrm{K}_1(\bigvee_{i=2}^{n} paid_i))$, $\varphi_3 = \mathrm{G}(odd \rightarrow \mathrm{C}_{1,\ldots,n}\neg(\bigvee_{i=1}^{n} paid_i))$. The formula φ_1 ($i^{\mathtt{I}}(n) = 4n + 2$, $i(n) = 3$) expresses that always when the number of uttered differences is odd and the first cryptographer has not paid for dinner, then he knows the cryptographer who paid for dinner. The formula φ_2 ($i^{\mathtt{I}}(n) = 2$, $i(n) = 3$) states that it is always true that if the first cryptographer has not paid for dinner, then he knows that some other cryptographer pays. The formula φ_3 ($i^{\mathtt{I}}(n) = 4n + 2$, $i(n) = 3$) states that always when the number of uttered differences is odd, then it is common knowledge of all the cryptographers that none of the cryptographers has paid for dinner.

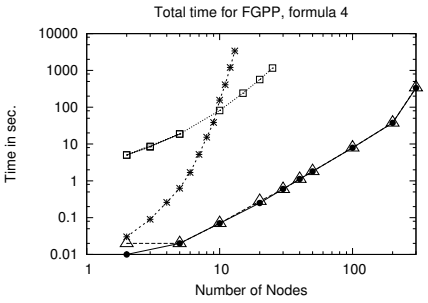
[3] `http://verics.ipipan.waw.pl/r/2is`

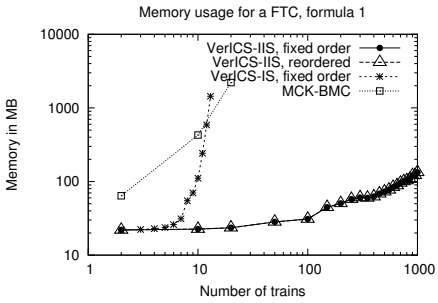
4.2 Performance Evaluation

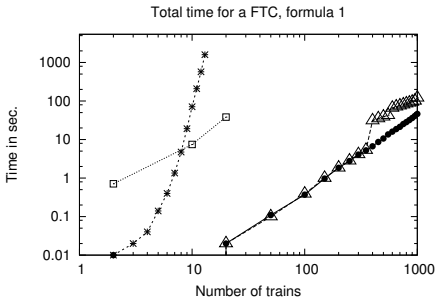


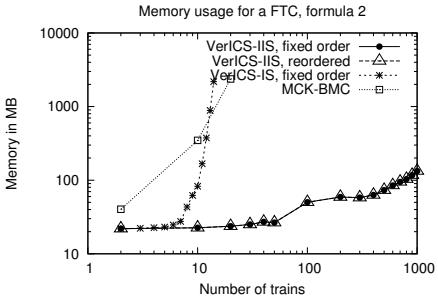
Comparing IS algorithms, in most cases MCK is better than VerICS-IS, but remains close when looking at the orders of magnitude. The reason for better performance of MCK may come from the fact that it is based on the translation

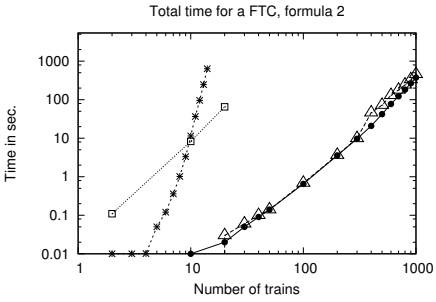

Memory usage for FGPP, formula 4
VerICS-IIS, fixed order
VerICS-IIS, reordered
VerICS-IS, fixed order
MCK-BMC
Memory in MB
Number of Nodes
10000
1000
100
10
1
10
100

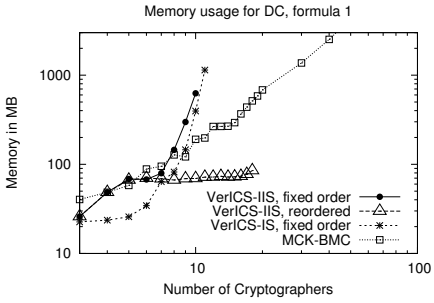

Total time for FGPP, formula 4
Time in sec.
Number of Nodes
10000
1000
100
10
1
0.1
0.01
1
10
100

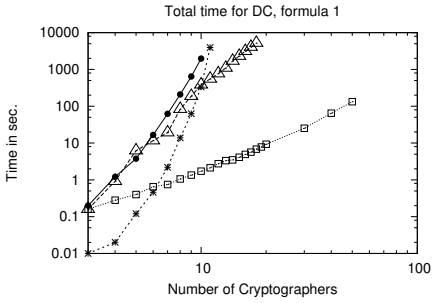

Memory usage for a FTC, formula 1
VerICS-IIS, fixed order
VerICS-IIS, reordered
VerICS-IS, fixed order
MCK-BMC
Memory in MB
Number of trains
10000
1000
100
10
1
10
100
1000

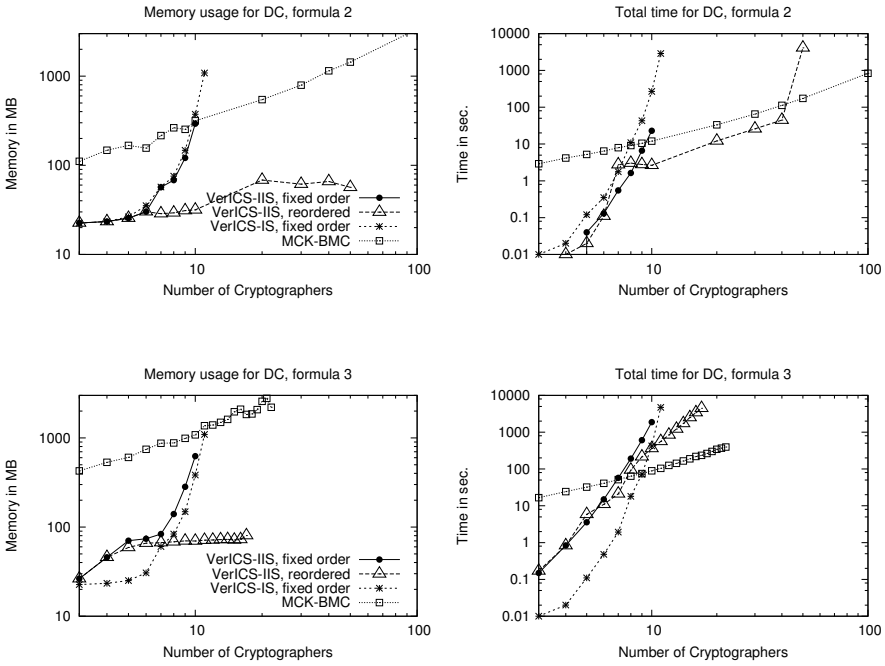

Total time for a FTC, formula 1
Time in sec.
Number of trains
1000
100
10
1
0.1
0.01
1
10
100
1000


Memory usage for a FTC, formula 2
VerICS-IIS, fixed order
VerICS-IIS, reordered
VerICS-IS, fixed order
MCK-BMC
Memory in MB
Number of trains
10000
1000
100
10
1
10
100
1000


Total time for a FTC, formula 2
Time in sec.
Number of trains
1000
100
10
1
0.1
0.01
1
10
100
1000


Memory usage for DC, formula 1
VerICS-IIS, fixed order
VerICS-IIS, reordered
VerICS-IS, fixed order
MCK-BMC
Memory in MB
Number of Cryptographers
1000
100
10
10
100


Total time for DC, formula 1
Time in sec.
Number of Cryptographers
10000
1000
100
10
1
0.1
0.01
10
100



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 φ_3 with no epistemic modalities, MCK proved to be more efficient, but for the formula φ_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.

5 Final Remarks

We have proposed, implemented, and experimentally evaluated our BDD-based BMC algorithms for LTLK over two variants of Interpreted Systems: standard and interleaved ones. 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 can expect to obtain similar results for other specification formalisms. Moreover, contrary to the SAT-based method of MCK and of [21], our BDD-based BMC is complete, i.e., it can also be easily used to verify that existential properties are false in the considered model.

In the future we are going to extend the presented algorithms to handle also the CTL*K properties. Since our implementation is in its preliminary stage, we also need to improve it in many ways, e.g., it should be investigated in the case of the non-interleaving semantics whether a different strategy for finding good BDD variables ordering would improve the results.

Our results are preliminary and the comparison is by no means complete. It ignores the fact that some formulae can give different verification results for each of the considered semantics, e.g., in the presence of the next-state operator X. However, we believe our results can be viewed as a justification and a starting point for further research on the subject.

References

1. A. Biere, A. Cimatti, E. Clarke, O. Strichman, and Y. Zhu. Bounded model checking. In *Highly Dependable Software*, volume 58 of *Advances in Computers*. Academic Press, 2003. Pre-print.
2. R. Bordini, M. Fisher, C. Pardavila, W. Visser, and M. Wooldridge. Model checking multi-agent programs with CASP. In *Proc. of the 15th Int. Conf. on Computer Aided Verification (CAV'03)*, volume 2725 of *LNCS*, pages 110–113. Springer-Verlag, 2003.
3. D. Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of Cryptology*, 1(1):65–75, 1988.
4. E. Clarke, O. Grumberg, and K. Hamaguchi. Another look at LTL model checking. In *Proc. of the 6th Int. Conf. on Computer Aided Verification (CAV'94)*, volume 818 of *LNCS*, pages 415–427. Springer-Verlag, 1994.
5. E. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, 1999.
6. R. Fagin, J. Y. Halpern, Y. Moses, and M. Vardi. *Reasoning about Knowledge*. MIT Press, Cambridge, 1995.
7. 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.

8. W. van der Hoek and M. Wooldridge. Model checking knowledge and time. In *Proc. of the 9th Int. SPIN Workshop (SPIN'02)*, volume 2318 of *LNCS*, pages 95–111. Springer-Verlag, 2002.
9. W. van der Hoek and M. Wooldridge. Cooperation, knowledge, and time: Alternating-time temporal epistemic logic and its applications. *Studia Logica*, 75(1):125–157, 2003.
10. 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.
11. A. V. Jones and A. Lomuscio. Distributed bdd-based bmc for the verification of multi-agent systems. In *AAMAS*, pages 675–682, 2010.
12. M. Kacprzak, A. Lomuscio, A. Niewiadomski, W. Penczek, F. Raimondi, and M. Szreter. Comparing BDD and SAT based techniques for model checking Chaum's dining cryptographers protocol. *Fundam. Inform.*, 72(1-2):215–234, 2006.
13. 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.
14. Alessio Lomuscio, Wojciech Penczek, and Hongyang Qu. Partial order reduction for model checking interleaved multi-agent systems. In *AAMAS, IFAAMAS Press.*, pages 659–666, 2010.
15. R. van der Mayden and K. Su. Symbolic model checking the knowledge of the dining cryptographers. In *Proc. of the 17th IEEE Computer Security Foundations Workshop (CSFW-17)*, pages 280–291. IEEE Computer Society, June 2004.
16. A. Męski, W. Penczek, and M. Szreter. Bounded model checking linear time and knowledge using decision diagrams. In *Proc. of the Int. Workshop on Concurrency, Specification and Programming (CS&P'11)*, pages 363–375, 2011.
17. A. Męski, W. Penczek, M. Szreter, B. Woźna-Szcześniak, and A. Zbrzezny. Bounded model checking for knowledge and linear time. In *Proceedings of the 11th AAMAS*. IFAAMAS Press, 2012. To appear.
18. R. van der Meyden and N. V. Shilov. Model checking knowledge and time in systems with perfect recall. In *Proc. of the 19th Conf. on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'99)*, volume 1738 of *LNCS*, pages 432–445. Springer-Verlag, 1999.
19. D. Peled. All from one, one for all: On model checking using representatives. In *Proc. of the 5th Int. Conf. on Computer Aided Verification (CAV'93)*, volume 697 of *LNCS*, pages 409–423. Springer-Verlag, 1993.
20. W. Penczek and A. Lomuscio. Verifying epistemic properties of multi-agent systems via bounded model checking. *Fundam. Inform.*, 55(2):167–185, 2003.
21. W. Penczek, B. Woźna-Szcześniak, and A. Zbrzezny. Towards SAT-based BMC for LTLK over interleaved interpreted systems. In *Proc. of the Int. Workshop on Concurrency, Specification and Programming (CS&P'11)*, pages 565–576, 2011.
22. 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.
23. K. Su, Abdul Sattar, and Xiangyu Luo. Model checking temporal logics of knowledge via OBDDs. *The Computer Journal*, 50(4):403–420, 2007.