The BMC Method for the Existential Part of RTCTLK and Interleaved Interpreted Systems

Bożena Woźna-Szcześniak, Agnieszka Zbrzezny, and Andrzej Zbrzezny

IMCS, Jan Długosz University. Al. Armii Krajowej 13/15, 42-200 Częstochowa, Poland
{b.wozna,a.zbrzezny,agnieszka.zbrzezny}@ajd.czest.pl

Abstract. In the paper, we focus on the formal verification of multi-agent systems – modelled by interleaved interpreted systems – by means of the bounded model checking (BMC) method, where specifications are expressed in the existential fragment of the Real-Time Computation Tree Logic augmented to include standard epistemic operators (RTECTLK). In particular, we define an improved SAT-based BMC for RTECTLK, and present performance evaluation of our newly developed BMC method by means of the well known train controller and generic pipeline systems.

1 Introduction

The problem of model checking [4] is to check automatically whether a structure M defines a model for a modal (temporal, epistemic, etc.) formula p. Bounded model checking (BMC) is a verification technique designed for finding counterexamples, and whose main idea is to consider a model curtailed to a specific depth. BMC via SAT was first proposed for linear-time temporal logic (LTL, LTL+Past) [1,2], and then it was extended, among others, to the universal fragment of CTL [18,22], and to the branching time epistemic logic, called ACTLK [13,17].

Multi-agent systems (MASs) are composed of many intelligent agents that interact with each other. The agents can share a common goal or they can pursue their own interests. Also, the agents may have deadlines or other timing constraints to achieve intended targets. As it was shown in [8], knowledge is a useful concept for analyzing the information state and the behaviour of agents in multi-agent systems. In particular, it is useful to reason about and to verify the evolution over time of epistemic states [10]. Thus reasoning about knowledge of agents and multi-agent systems has always been a core issue in artificial intelligence. Therefore, many logical formalisms and verification techniques, especially the one based on model checking, have been developed and refined over the years, among others, [8,9,11,13,14,19,20].

An existential fragment of the soft real-time CTL (RTECTL) [7] is a propositional branching-time temporal logic with bounded operators, which was introduced to permit specification and reasoning about time-critical correctness properties. More specifically, existential CTL formulae allow for the verification of properties such as "*there is a computation such that φ will eventually occur*", or "*there is a computation such that φ will never be asserted*". However, it is not possible to directly express bounded properties like for example "*there is a computation such that φ will occur in less than 30 unit*

L. Antunes and H.S. Pinto (Eds.): EPIA 2011, LNAI 7026, pp. 551–565, 2011.

time", or "*there is a computation such that φ will always be asserted between 10 and 30 unit time*". While it is true that properties like the above can be expressed using nested applications of the next state operators, the resulting CTL formula can be very complex and cumbersome to work with. RTECTL defeats this restriction by allowing bounds on all temporal operators to be specified, and provides a much more compact and convenient way of expressing time-bounded properties. The RTECTLK language is an epistemic soft real-time computation tree logic that is the fusion [3] of the two underlying languages: RTECTL and a multi-modal logic $S5_n$ for knowledge that satisfies the following properties: the truth axiom, the distribution axiom, the necessitation axiom, the positive introspection axiom, and the negative introspection axiom.

A version of the BMC method for specifications expressed in RTECTLK, and MASs modelled by interpreted systems [8], in which agents have time-limits or other explicit timing constraints to accomplish intended goals, has been published in pre-proceedings of CEE-SET'2009 ([21]). However, this method not only does not take into account the development related to the BMC algorithm for ECTL [22], but also it is based on unnecessarily complex bounded semantics. Moreover, it has not been implemented and experimentally evaluated.

The main idea of the [21] translation is the following. Given are a RTECTLK formula φ and a bound $k \in \mathbb{N}$. First the number of paths of length k (k-paths) that are sufficient for checking the formula φ is computed; this is done by means of the function f_k. Next, the set of k-paths of size $f_k(\varphi)$ is created and used to translate every subformula of the formula φ. This means that this translation uses all the $f_k(\varphi)$ k-paths both for the formula φ and for each of its proper subformulae. In the paper we propose an improvement of this translation, which is based on the BMC method for ECTL [22]. Its main idea consists in translating every subformula ψ of the formula φ using only $f_k(\psi)$ paths of length k. So, our new BMC algorithm uses a reduced number of paths, what results in significantly smaller and less complicated propositional formulae that encode the RTECTLK properties.

Specifically, in the paper we make three contributions: first, we present the BMC method (which is based on the improved ECTL translation [22]) for specifications expressed in RTECTLK, and multi-agent systems modelled by interleaved interpreted systems (IIS) introduced in [15] (a subclass of standard interpreted systems [8]); second, we implement under the same semantics, i.e., interleaved interpreted systems, both BMC translations the one presented in [21], and our new one; third, we present performance evaluation of the two implemented BMC algorithms for the verification of several properties expressed in RTECTLK.

The [21] BMC translation could be also improved by adopting the SAT-based BMC for ECTLK [13], which according to the authors significantly improves the [22] BMC encoding. However, for this paper we have decided not to make use of the method presented in [13], but we plan to research this possibility in the future.

The structure of the paper is as follows. In Section 2 we shortly introduce interleaved interpreted systems and the RTECTLK logic. In Section 3 we define an improved SAT-based BMC for RTECTLK, and we prove its correctness. In Section 4 we present performance evaluation of our newly developed SAT-based BMC algorithm for RTECTLK. In Section 5 we conclude the paper.

2 Preliminaries

In this section we first define *interleaved interpreted systems (IIS)*, and next we introduce syntax and semantics of RTECTLK. The formalism of IIS was introduced in [15] to model multi-agent system (MASs) that are composed of multiple agents, each of which is an independently operating entity, and to reason about the agents' epistemic and temporal properties. The interleaved interpreted system is a special subclass of standard interpreted systems introduced in [8], which allow for modelling asynchronous MASs. In this formalism, each agent is modelled using a set of local states, a set of actions, a local protocol, and a local (interleaved) evolution function. In IIS only one action at a time is performed in a global transition. Specifically, if several agents act in the global action, then they have to perform the same action, thereby synchronising at that particular time step. Thus, the local protocols are defined in such a way that if an agent has the action being performed in its set of actions, then it must be able to perform the action in order to allow the global evolution of the whole system.

Interleaved Interpreted Systems. We assume that a MAS consists of n agents[1], and by $Ag = \{1, \ldots, n\}$ we denote the non-empty set of agents. Further, we assume that each agent $c \in Ag$ is in some particular local state at a given point in time, and that a set L_c of local states for agent $c \in Ag$ is non-empty and finite (this is required by the model checking algorithms). An agent's local state encapsulates all the information the agent has access to, and which are required to completely characterise its state. Further, with each agent $c \in Ag$ we associate a finite set of *possible actions* Act_c such that a special action ϵ_c, called "null", belongs to Act_c; as it will be clear below the local state of agent c remains the same if the null action is performed. We do not assume that the sets Act_c (for all $c \in Ag$) are disjoint. We define the following sets $Act = \bigcup_{c \in Ag} Act_c$ and $Agent(a) = \{c \in Ag \mid a \in Act_c\}$. Next, for each agent $c \in Ag$ we associate a protocol that defines rules, according to which actions may be performed in each local state. The protocol for agent $c \in Ag$ is a function $P_c : L_c \to 2^{Act_c}$ such that $\epsilon_c \in P_c(l)$ for any $l \in L_c$, i.e., we insist on the null action to be enabled at every local state. Note that the above definition of the protocol enables more than one action to be performed for a given local state. This means that an agent selects non-deterministically which action to perform. Finally, for each agent c, there is defined a (partial) evolution function $t_c : L_c \times Act_c \to L_c$ such that for each $l \in L_c$ and for each $a \in P_c(l)$ there exists $l' \in L_c$ such that $t_c(l, a) = l'$; moreover, for each $l \in L_c$, $t_c(l, \epsilon_c) = l$. Note that the local evolution function considered here differs from the standard one (see [8]) by having the local action instead of the join action as the parameter.

A *global state* $g = (l_1, \ldots, l_n)$ is a tuple of n local states, one per each agent, corresponding to an instantaneous snapshot of the MAS at a given time. By $l_c(g)$ we denote the local component of agent $c \in Ag$ in a global state $g = (l_1, \ldots, l_n)$, and by G we denote a set of global states. It is assumed that, in every state, agents evolve simultaneously. Thus the *global interleaved evolution function* $t : G \times Act_1 \times \cdots \times Act_n \to G$ is defined as follows: $t(g, a_1, \ldots, a_n) = g'$ iff there exists an action $a \in$

[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.

$Act \setminus \{\epsilon_1, \ldots, \epsilon_n\}$ such that for all $c \in Agent(a)$, $a_c = a$ and $t_c(l_c(g), a) = l_c(g')$, and for all $c \in Ag \setminus Agent(a)$, $a_c = \epsilon_c$ and $t_c(l_c(g), a_c) = l_c(g)$. In brief we write the above as $g \xrightarrow{a} g'$. Now, for a given set of agents Ag and a set of propositional variables $\mathcal{PV}$, which can be either true or false, an *interleaved interpreted system* is a tuple: $IIS = (\iota, < L_c, Act_c, P_c, t_c >_{c \in Ag}, \mathcal{V})$, where $\iota \in G$ is an initial global state, and $\mathcal{V} : G \to 2^{\mathcal{PV}}$ is a valuation function. With such an interleaved interpreted system IIS it is possible to associate a *Kripke model* $M = (\iota, S, T, \{\sim_c\}_{c \in Ag}, \mathcal{V})$, where ι is the initial global state; $S \subseteq G$ is a set of reachable global states that is generated from ι by using the global interleaved evolution functions t; $T \subseteq S \times S$ is a global transition (temporal) relation on S defined by: sTs' iff there exists an action $a \in Act \setminus \{\epsilon_1, \ldots, \epsilon_n\}$ such that $s \xrightarrow{a} s'$. We assume that the relation is total, i.e., for any $s \in S$ there exists an $a \in Act \setminus \{\epsilon_1, \ldots, \epsilon_n\}$ such that $s \xrightarrow{a} s'$ for some $s' \in S$; $\sim_c \subseteq S \times S$ is an indistinguishability relation for agent c defined by: $s \sim_c s'$ iff $l_c(s') = l_c(s)$; and $\mathcal{V} : S \to 2^{\mathcal{PV}}$ is the valuation function of IIS subtracted to the set S. $\mathcal{V}$ assigns to each state a set of propositional variables that are assumed to be true at that state. For more details we refer to [8].

Syntax of RTECTLK. Let $p \in \mathcal{PV}$, $c \in Ag$, $\Gamma \subseteq Ag$, and I be an interval in $\mathbb{N} = \{0, 1, 2, \ldots\}$ of the form: $[a, b)$ and $[a, \infty)$, for $a, b \in \mathbb{N}$ [2]. Hereafter by $left(I)$ we denote the left end of the interval I, i.e., $left(I) = a$, and by $right(I)$ the right end of the interval I, i.e., $right([a, b)) = b - 1$ and $right([a, \infty)) = \infty$. The language RTECTLK is defined by the following grammar:

$$\varphi := \mathbf{true} \mid \mathbf{false} \mid p \mid \neg p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \mathrm{EX}\varphi \mid \mathrm{E}(\varphi \mathrm{U}_I \varphi) \mid \mathrm{EG}_I \varphi \mid \overline{\mathrm{K}}_c \varphi \mid \overline{\mathrm{D}}_\Gamma \varphi \mid \overline{\mathrm{E}}_\Gamma \varphi \mid \overline{\mathrm{C}}_\Gamma \varphi$$

U_I and G_I are the operators, resp., for bounded "Until" and "Always". The formula $\mathrm{E}(\alpha \mathrm{U}_I \beta)$ is read as "there exists a computation such that β holds in the interval I at least in one state and always earlier α holds", the formula $\mathrm{EG}_I \alpha$ is read as "there exists a computation such that α always holds in the interval I". The remaining bounded temporal operators are introduced in the standard way: $\mathrm{E}(\alpha \,\mathrm{R}_I\, \beta) \stackrel{def}{=} \mathrm{E}(\beta \,\mathrm{U}_I\, (\alpha \wedge \beta)) \vee \mathrm{EG}_I \beta$, $\mathrm{EF}_I \alpha \stackrel{def}{=} \mathrm{E}(\mathbf{true} \mathrm{U}_I \alpha)$. $\overline{\mathrm{K}}_c$ is the operator dual for the standard epistemic modality K_c ("agent c knows"), so $\overline{\mathrm{K}}_c \alpha$ is read as "agent c does not know whether or not α holds". Similarly, the modalities $\overline{\mathrm{D}}_\Gamma, \overline{\mathrm{E}}_\Gamma, \overline{\mathrm{C}}_\Gamma$ are the dual operators for $\mathrm{D}_\Gamma, \mathrm{E}_\Gamma, \mathrm{C}_\Gamma$ representing distributed knowledge in the group Γ, "everyone in Γ knows", and common knowledge among agents in Γ.

Semantics of RTECTLK. Let $M = (\iota, S, T, \{\sim_c\}_{c \in Ag}, \mathcal{V})$ be a model. Then, a *path* in M is an infinite sequence $\pi = (s_0, s_1, \ldots)$ of states such that $(s_j, s_{j+1}) \in T$ for each $j \in \mathbb{N}$. For a path $\pi = (s_0, s_1, \ldots)$, we take $\pi(j) = s_j$. By $\Pi(s)$ we denote the set of all the paths starting at $s \in S$. For the group epistemic modalities we also define the following. If $\Gamma \subseteq Ag$, then $\sim_\Gamma^E \stackrel{def}{=} \bigcup_{c \in \Gamma} \sim_c$, $\sim_\Gamma^C \stackrel{def}{=} (\sim_\Gamma^E)^+$ (the transitive closure of $\sim_\Gamma^E$), and $\sim_\Gamma^D \stackrel{def}{=} \bigcap_{c \in \Gamma} \sim_c$. Given the above, the formal semantics of RTECTLK is defined recursively as follows:

[2] Note that the remaining forms of intervals can be defined by means of $[a, b)$ and $[a, \infty)$.

- $M, s \models \mathbf{true}$, • $M, s \not\models \mathbf{false}$, • $M, s \models p$ iff $p \in \mathcal{V}(s)$, • $M, s \models \neg p$ iff $p \notin \mathcal{V}(s)$,
- $M, s \models \alpha \wedge \beta$ iff $M, s \models \alpha$ and $M, s \models \beta$,
- $M, s \models \alpha \vee \beta$ iff $M, s \models \alpha$ or $M, s \models \beta$,
- $M, s \models \mathrm{EX}\alpha$ iff $(\exists \pi \in \Pi(s))(M, \pi(1) \models \alpha)$,
- $M, s \models \mathrm{E}(\alpha \mathrm{U}_I \beta)$ iff $(\exists \pi \in \Pi(s))(\exists m \in I)[M, \pi(m) \models \beta$ and $(\forall j < m) M, \pi(j) \models \alpha]$,
- $M, s \models \mathrm{EG}_I \alpha$ iff $(\exists \pi \in \Pi(s))$ such that $(\forall m \in I)[M, \pi(m) \models \alpha]$,
- $M, s \models \overline{\mathrm{K}}_c \alpha$ iff $(\exists s' \in S)(s \sim_c s'$ and $M, s' \models \alpha)$,
- $M, s \models \overline{Y} \alpha$ iff $(\exists s' \in S)(s \sim s'$ and $M, s' \models \alpha)$, where $\overline{Y} \in \{\overline{\mathrm{D}}_\Gamma, \overline{\mathrm{E}}_\Gamma, \overline{\mathrm{C}}_\Gamma\}$, and $\sim \in \{\sim^D_\Gamma, \sim^E_\Gamma, \sim^C_\Gamma\}$.

We end the section by defining the notions of validity and the model checking problem. Namely, a RTECTLK formula φ is *valid* in M (denoted $M \models \varphi$) iff $M, \iota \models \varphi$, i.e., φ is true at the initial state of the model M. The *model checking problem* asks whether $M \models \varphi$.

3 SAT-Based BMC for RTECTLK

As it was already mentioned, the BMC method for RTECTLK presented in [21] is based on unnecessarily complicated bounded semantics, and it does not take into account the BMC encoding of [22]. In this section, we present a new bounded semantics for RTECTLK, show its equivalence to the unbounded one, and define an improved SAT-based BMC method for RTECTLK, which is based on the BMC encoding presented in [22]. As usual, we start by defining k-paths and (k, l)-*loops*. Next we define a bounded semantics, which is later used for translation to SAT.

Bounded Semantics. Let $M = (\iota, S, T, \{\sim_c\}_{c \in Ag}, \mathcal{V})$ be a model and $k \geq 0$. A k-*path* π_k in M is a finite sequence of states $(s_0, \ldots, s_k)$ such that $(s_j, s_{j+1}) \in T$ for each $0 \leq j < k$. By $\Pi_k(s)$ we denote the set of all the k-paths starting at s in M. A k-path π_k is a (k, l)-*loop* iff $\pi_k(l) = \pi_k(k)$ for some $0 \leq l < k$; note that (k, l)-loop π generates the infinite path of the following form: $u \cdot v^\omega$ with $u = (\pi(0), \ldots, \pi(l-1))$ and $v = (\pi(l), \ldots, \pi(k-1))$. Since in the bounded semantics we consider finite prefixes of paths only, the satisfiability of all the temporal operators depends on whether a considered k-path is a loop. Thus, as customary, we introduce a function $loop : \bigcup_{s \in S} \Pi_k(s) \to 2^{\mathbb{N}}$, which identifies these k-paths that are loops. The function is defined as: $loop(\pi_k) = \{l \mid 0 \leq l < k \text{ and } \pi_k(l) = \pi_k(k)\}$.

Definition 1. *Given are a bound $k \in \mathbb{N}$, a model M, and* RTECTLK *formulae α, β. $M, s \models_k \alpha$ denotes that α is $k-$true at the state s of M. The relation $\models_k$ is defined inductively as follows:*

- $M, s \models_k \mathbf{true}$, • $M, s \not\models_k \mathbf{false}$,
- $M, s \models_k p$ iff $p \in \mathcal{V}(s)$, • $M, s \models_k \neg p$ iff $p \notin \mathcal{V}(s)$,
- $M, s \models_k \alpha \vee \beta$ iff $M, s \models_k \alpha$ *or* $M, s \models_k \beta$,
- $M, s \models_k \alpha \wedge \beta$ iff $M, s \models_k \alpha$ *and* $M, s \models_k \beta$,
- $M, s \models_k \mathrm{EX}\alpha$ iff $k > 0$ and $(\exists \pi \in \Pi_k(s)) M, \pi(1) \models_k \alpha$,
- $M, s \models_k \mathrm{E}(\alpha \mathrm{U}_I \beta)$ iff $(\exists \pi \in \Pi_k(s))(\exists 0 \leq m \leq k)(m \in I$ and $M, \pi(m) \models_k \beta$ and $(\forall 0 \leq j < m) M, \pi(j) \models_k \alpha)$,

- $M, s \models_k \mathrm{EG}_I\alpha$ iff $(\exists \pi \in \Pi_k(s))((k \geq \mathit{right}(I)$ and $(\forall j \in I)\ M, \pi(j) \models_k \alpha)$ or $(k < \mathit{right}(I)$ and $(\exists l \in \mathit{loop}(\pi))(\forall \mathit{min}(\mathit{left}(I), l) \leq j < k)\ M, \pi(j) \models_k \alpha))$,
- $M, s \models_k \overline{\mathrm{K}}_c\alpha$ iff $(\exists \pi \in \Pi_k(\iota))(\exists 0 \leq j \leq k)(M, \pi(j) \models_k \alpha$ and $s \sim_c \pi(j))$,
- $M, s \models_k \overline{Y}\alpha$ iff $(\exists \pi \in \Pi_k(\iota))(\exists 0 \leq j \leq k)(M, \pi(j) \models_k \alpha$ and $s \sim \pi(j))$, *where* $\overline{Y} \in \{\overline{\mathrm{D}}_\Gamma, \overline{\mathrm{E}}_\Gamma, \overline{\mathrm{C}}_\Gamma\}$ *and* $\sim \in \{\sim^D_\Gamma, \sim^E_\Gamma, \sim^C_\Gamma\}$.

A RTECTLK formula φ is *valid in model M with bound k* (denoted $M \models_k \varphi$) iff $M, \iota \models_k \varphi$, i.e., φ is $k-$true at the initial state of the model M. The *bounded model checking problem* asks whether there exists $k \in \mathbb{N}$ such that $M \models_k \varphi$.

By straightforward induction on the length of a RTECTLK formula φ we can show that the following two lemmas hold.

Lemma 1. *Given are a model M, a* RTECTLK *formula φ, and a bound $k \geq 0$. Then, for each s in M, $M, s \models_k \varphi$ implies $M, s \models_{k+1} \varphi$.*

Lemma 2. *Given are a model M, a* RTECTLK *formula φ, and a bound $k \geq 0$. Then, for each s in M, $M, s \models_k \varphi$ implies $M, s \models \varphi$.*

Lemma 3. *Given are a model M, and a* RTECTLK *formula φ. Then, for each s in M, $M, s \models \varphi$ implies that there exists $k \geq 0$ such that $M, s \models_k \varphi$.*

Proof (By induction on the length of φ). The lemma follows directly for the propositional variables and their negations. Next, assume that the hypothesis holds for all the proper sub-formulae of φ. If φ is equal to either $\alpha \wedge \beta$, $\alpha \vee \beta$, or $\mathrm{EX}\alpha$, then it is easy to check that the lemma holds. For the epistemic operators, i.e., $\varphi = \overline{\mathrm{K}}_c\alpha, \overline{\mathrm{E}}_\Gamma\alpha, \overline{\mathrm{D}}_\Gamma\alpha, \overline{\mathrm{C}}_\Gamma\alpha$, the proof is like in [17] (see Lemma 2). So, consider φ to be of the following forms:

– Let $\varphi = \mathrm{EG}_I\alpha$ and $M, s \models \varphi$. By the definition of the unbounded semantics we have that there exists path $\pi \in \Pi(s)$ such that $(\forall j \in I)(M, \pi(j) \models \alpha)$. We have to consider two cases for the form of the interval I.

(a) $\mathit{right}(I) < \infty$. By the inductive assumption we have that for each $j \in I$ there exists k_j such that $M, \pi(j) \models_{k_j} \alpha$. Let $k = max\{\mathit{right}(I), max\{k_j \mid j \in I\}\}$. It follows by Lemma 1 that for each $j \in I, M, \pi(j) \models_k \alpha$. Now, consider the prefix π_k of length k of π. We have that $\pi_k \in \Pi_k(s)$, so by Definition 1 we conclude that $M, s \models_k \mathrm{EG}_I\alpha$.

(b) $\mathit{right}(I) = \infty$. Since number of states of M is finite, there exists $k' \geq \mathit{left}(I)$ such that for some $l < k'$, $\pi(k') = \pi(l)$. Now, let $u = (\pi(0), \ldots, \pi(l-1))$, $v = (\pi(l), \ldots, \pi(k'-1))$, and $\pi' = u \cdot v^\omega$. It is clear that for each $j \in I$, $M, \pi'(j) \models \alpha$. Therefore, taking into consideration the form of π', we get that for each j such that $l \leq j \leq k'$, $M, \pi'(j) \models \alpha$. Now, by the inductive assumption we get that for each $j \in I$ there exists k_j such that $M, \pi'(j) \models_{k_j} \alpha$. Then, let k be the least natural number such that for each $j \in \{l, \ldots, k'\}$, $k \geq k_j$, and moreover $k = l + m(k' - l)$, for some natural number $m > 0$. It follows by Lemma 1 that for each $j \in \{l, \ldots, k\}$, $M, \pi(j) \models_k \alpha$. Now, consider the prefix π'_k of length k of π'. Obviously, $l \in \mathit{loop}(\pi'_k)$ and $M, \pi'_k(j) \models_k \alpha$, for each j such that $\mathit{min}\{\mathit{left}(I), l\} \leq j < k$. We have that $\pi_k \in \Pi_k(s)$, so by Definition 1 we conclude that $M, s \models_k \mathrm{EG}_I\alpha$.

– Let $\varphi = \mathrm{E}(\alpha \mathrm{U}_I \beta)$ and $M, s \models \varphi$. By the definition of the unbounded semantics we have that there exists path $\pi \in \Pi(s)$ and $m \in I$ such that $M, \pi(m) \models \beta$ and $(\forall j < m) M, \pi(j) \models \alpha$. Thus, by the inductive assumption we have that there exists k' such that $M, \pi(m) \models_{k'} \beta$ and for each $0 \leq j < m$ there exists k_j such that $M, \pi(j) \models_{k_j} \alpha$. It follows by Lemma 1 that $M, \pi(m) \models_k \beta$ and $M, \pi(j) \models_k \alpha$, where $k = max\{k', k_0, \ldots, k_m\}$. Now, consider the prefix π_k of length k of π. We have that $\pi_k \in \Pi_k(s)$. Since $m \in I$, we conclude that $M, s \models_k \mathrm{E}(\alpha \mathrm{U}_I \beta)$.

The following theorem states that for a given model and formula there exists a bound k such that the model checking problem ($M \models \varphi$) can be reduced to the bounded model checking problem ($M \models_k \varphi$). Its proof follows from Lemmas 2 and 3.

Theorem 1. *Let M be a model and φ a* RTECTLK *formula. Then, the following equivalence holds: $M \models \varphi$ iff there exists $k \geq 0$ such that $M \models_k \varphi$.*

Now we show how to reduce BMC for RTECTLK to the propositional satisfiability problem. This reduction allows us to use efficient SAT solvers to perform model checking. We begin by introducing a function f_k that gives a bound on the number of k-paths of M, which are sufficient to validate a given RTECTLK formula. Namely, the function $f_k : \text{RTECTLK} \to \mathbb{N}$ is defined as follows:

- $f_k(\mathbf{true}) = f_k(\mathbf{false}) = f_k(p) = f_k(\neg p) = 0$, where $p \in \mathcal{PV}$,
- $f_k(\alpha \wedge \beta) = f_k(\alpha) + f_k(\beta)$,
- $f_k(\alpha \vee \beta) = max\{f_k(\alpha), f_k(\beta)\}$,
- $f_k(Y\alpha) = f_k(\alpha) + 1$, for $Y \in \{\mathrm{X}, \overline{\mathrm{K}}_c, \overline{\mathrm{D}}_\Gamma, \overline{\mathrm{E}}_\Gamma\}$,
- $f_k(\mathrm{E}(\alpha \mathrm{U}_I \beta)) = k \cdot f_k(\alpha) + f_k(\beta) + 1$,
- $f_k(\mathrm{EG}_I \alpha) = (k+1) \cdot f_k(\alpha) + 1$,
- $f_k(\overline{\mathrm{C}}_\Gamma \alpha) = f_k(\alpha) + k$.

By straightforward induction on the length of a RTECTLK formula φ we can show that φ is $k-$true in M if and only if φ is $k-$true in M with a number of $k-$paths reduced to $f_k(\varphi)$.

The New Translation of RTECTLK to Propositional Formulae. Now we present our translation of a RTECTLK formula into a propositional formula. Given are a model $M = (\iota, S, T, \{\sim_c\}_{c \in Ag}, \mathcal{V})$, a RTECTLK formula φ, and a bound $k \geq 0$. It is well known that the main idea of the BMC method consists in translating the bounded model checking problem, i.e., $M \models_k \varphi$, to the problem of checking the satisfiability of the following propositional formula:

$$[M, \varphi]_k := [M^{\varphi,\iota}]_k \ \wedge \ [\varphi]_{M,k} \tag{1}$$

The formula $[M^{\varphi,\iota}]_k$ constrains the $f_k(\varphi)$ symbolic k-paths to be valid k-paths of M, while the formula $[\varphi]_{M,k}$ encodes a number of constraints that must be satisfied on these sets of k-paths for φ to be satisfied. Once this translation is defined, checking satisfiability of a RTECTLK formula can be done by means of a SAT-solver.

In order to define the formula $[M, \varphi]_k$ we proceed as follows. We assume that each state s of M is encoded by a bit-vector whose length, say r, depends on the number of agents' local states. Thus, each state s of M we can represent by a vector $w = (u_1, \ldots, u_r)$ of propositional variables (usually called *state variables*), to which

we refer to as *symbolic states*. A finite sequence $(w_0, \ldots, w_k)$ of symbolic states is called a *symbolic k-path*. Since, in general, we may need to consider more than one symbolic k-path, we introduce a notion of the j-th symbolic k-path, which is denoted by $(w_{0,j}, \ldots, w_{k,j})$, where $w_{i,j}$ are symbolic states for $0 \leq j < f_k(\varphi)$ and $0 \leq i \leq k$. Note that the exact number of necessary symbolic k-paths depends on the checked formula φ, and it can be calculated by means of the function f_k.

For a given infinite set SV of state variables and a *valuation of state variables* $\sigma : SV \to \{0,1\}$ (a *valuation* for short), its extension to vectors of states variables $\boldsymbol{\sigma} : SV^r \to \{0,1\}^r$ is defined in the following way:

$$\boldsymbol{\sigma}((u_{j_1}, \ldots, u_{j_r})) = (\sigma(u_{j_1}), \ldots, \sigma(u_{j_r})).$$

In what follows for a symbolic state w, by $SV(w)$ we denote the set of all the state variables occurring in w.

Now, let w and w' be symbolic states such that $SV(w) \cap SV(w') = \emptyset$. We define the following propositional formulae:

- $I_s(w)$ is a formula over $SV(w)$ that is true for a valuation σ iff $\boldsymbol{\sigma}(w) = s$.
- $p(w)$ is a formula over w that is true for a valuation σ iff $p \in \mathcal{V}(\boldsymbol{\sigma}(w))$ (encodes a set of states of M in which $p \in \mathcal{PV}$ holds).
- $H(w, w')$ is a formula over $SV(w) \cup SV(w')$ that is true for a valuation σ iff $\boldsymbol{\sigma}(w) = \boldsymbol{\sigma}(w')$ (encodes equivalence of two global states).
- $H_c(w, w')$ is a formula over $SV(w) \cup SV(w')$ that is true for a valuation σ iff $l_c(\boldsymbol{\sigma}(w)) = l_c(\boldsymbol{\sigma}(w))$ (encodes equivalence of local states of agent c).
- $\mathcal{R}(w, w')$ is a formula over $SV(w) \cup SV(w')$ that is true for a valuation σ iff $(\boldsymbol{\sigma}(w), \boldsymbol{\sigma}(w')) \in T$ (encodes the transition relation of M).
- Let $j \in \mathbb{N}$, and I be an interval. Then $In(j, I) := \begin{cases} \mathbf{true}, \text{ if } j \in I \\ \mathbf{false}, \text{ if } j \notin I \end{cases}$.

The propositional formula $[M^{\varphi,\iota}]_k$ is defined over state variables in the set $\bigcup\{SV(w_{i,j}) \mid 0 \leq i \leq k \text{ and } 0 \leq j < f_k(\varphi)\}$, in the following way:

$$[M^{\varphi,\iota}]_k := I_\iota(w_{0,0}) \wedge \bigwedge_{j=0}^{f_k(\varphi)-1} \bigwedge_{i=0}^{k-1} \mathcal{R}(w_{i,j}, w_{i+1,j}) \tag{2}$$

The next step of the translation is the transformation of a RTECTLK formula φ into a propositional formula $[\varphi]_{M,k} := [\varphi]_k^{[0,0,F_k(\varphi)]}$, where $F_k(\varphi) = \{j \in \mathbb{N} \mid 0 \leq j < f_k(\varphi)\}$, and $[\varphi]_k^{[m,n,A]}$ denotes the translation of φ at the symbolic state $w_{m,n}$ using k-paths from the set A.

Following [22], to translate an RTECTLK formula with an operator Q (where $\mathrm{Q} \in \{\mathrm{EX}, \mathrm{EU}_I, \mathrm{EG}_I, \overline{\mathrm{K}}_1, \ldots, \overline{\mathrm{K}}_n, \overline{\mathrm{D}}_\Gamma, \overline{\mathrm{E}}_\Gamma\}$), we want exactly one path to be chosen for translating the operator Q, and the remaining k-paths to be used to translate arguments of Q. To accomplish this goal we need some auxiliary functions. However, before we define them, we first recall a definition of a relation $\prec$ that is defined on the power set of $\mathbb{N}$ as follows: $A \prec B$ iff for all natural numbers x and y, if $x \in A$ and $y \in B$, then $x < y$. Notice that from the definition of $\prec$ it follows that $A \prec B$ iff either $A = \emptyset$ or $B = \emptyset$ or $A \neq \emptyset$, $B \neq \emptyset$, $A \cap B = \emptyset$ and $max(A) < min(B)$. Now, let $A \subset \mathbb{N}$ be a finite nonempty set, $k, p \in \mathbb{N}$, and $m \in \mathbb{N}$ such that $m \leq |A|$. Then,

- $g_l(A, m)$ denotes the subset B of A such that $|B| = m$ and $B \prec A \setminus B$.
- $g_r(A, m)$ denotes the subset C of A such that $|C| = m$ and $A \setminus C \prec C$.
- $g_s(A)$ denotes the set $A \setminus \{min(A)\}$.
- If $k+1$ divides $|A|-1$, then $h_{\mathrm{G}}(A, k)$ denotes the sequence $(B_0, \ldots, B_k)$ of subsets of $A \setminus \{min(A)\}$ such that $\bigcup_{j=0}^{k} B_j = A \setminus \{min(A)\}$, $|B_0| = \ldots = |B_k|$, and $B_i \prec B_j$ for every $0 \leq i < j \leq k$. If $h_{\mathrm{G}}(A, k) = (B_0, \ldots, B_k)$, then $h_{\mathrm{G}}(A, k)(j)$ denotes the set B_j, for every $0 \leq j \leq k$.
- If k divides $|A| - 1 - p$, then $h_{\mathrm{U}}(A, k, p)$ denotes the sequence $(B_0, \ldots, B_k)$ of subsets of $A \setminus \{min(A)\}$ such that $\bigcup_{j=0}^{k} B_j = A \setminus \{min(A)\}$, $|B_0| = \ldots = |B_{k-1}|$, $|B_k| = p$, and $B_i \prec B_j$ for every $0 \leq i < j \leq k$. If $h_{\mathrm{U}}(A, k, p) = (B_0, \ldots, B_k)$, then $h_{\mathrm{U}}(A, k, p)(j)$ denotes the set B_j, for every $0 \leq j \leq k$.

In order to explain the purpose of the auxiliary functions defined above let us recall that each RTECTLK formula φ will be translated by using a set of exactly $f_k(\varphi)$ symbolic k-paths. In what follows we will say that a set A of positive natural numbers is used to translate a formula φ instead of saying that the set of symbolic k-paths with indices in A is used to translate the formula φ. Moreover, saying that a set A is used to translate a formula φ assumes that $|A| = f_k(\varphi)$.

The function g_l is used in the translation of the formulae with the main connective being a disjunction: for a given RTECTLK formula $\alpha \vee \beta$, if a set A is to be used to translate this formula, then the set $g_l(A, f_k(\alpha))$ is used to translate the subformula α and the set $g_l(A, f_k(\beta))$ is used to translate the subformula β.

The functions g_l and g_r are used in the translation of the formulae with the main connective being a conjunction: for a given RTECTLK formula $\alpha \wedge \beta$, if a set A is to be used to translate this formula, then the set $g_l(A, f_k(\alpha))$ is used to translate the subformula α and the set $g_r(A, f_k(\beta))$ is used to translate the subformula β.

The function g_s is used in the translation of the formulae with the main connective $\mathrm{Q} \in \{\mathrm{EX}, \overline{\mathrm{K}}_1, \ldots, \overline{\mathrm{K}}_n, \overline{\mathrm{D}}_\Gamma, \overline{\mathrm{E}}_\Gamma\}$: for a given RTECTLK formula $\mathrm{Q}\alpha$, if a set A is to be used to translate this formula, then the path of the number $min(A)$ is used to translate the operator Q and the set $g_s(A)$ is used to translate the subformula α.

The function h_{G} is used in the translation of the formulae with the main connective EG_I: for a given RTECTLK formula $\mathrm{EG}_I\alpha$, if a set A is to be used to translate this formula, then the path of the number $min(A)$ is used to translate the operator EG_I and the set $h_{\mathrm{G}}(A, k)(j)$, for every $0 \leq j \leq k$, is used to translate the formula α at the j-th symbolic state of the symbolic k-path of the number $min(A)$. Notice that if $k + 1$ does not divide $|A| - 1$, then $h_{\mathrm{G}}(A, k)$ is undefined. However, for every set A such that $|A| = f_k(\mathrm{EG}_I\alpha)$, it follows from the definition of f_k that $k + 1$ divides $|A| - 1$.

The function h_{U} is used in the translation of the formulae with the main connective EU_I: for a given RTECTLK formula $\mathrm{E}(\alpha \mathrm{U}_I \beta)$ if a set A is to be used to translate this formula, then the path of the number $min(A)$ is used to translate the operator EU_I, the set $h_{\mathrm{U}}(A, k, f_k(\beta))(j)$, for every $0 \leq j \leq k$, is used to translate the formula β at the symbolic state $w_{j,min(A)}$ and the set $h_{\mathrm{U}}(A, k, f_k(\beta))(i)$, for every $0 \leq i < j$, is used to translate the formula α at the symbolic state $w_{i,min(A)}$. Notice that if k does not divide $|A| - 1 - p$, then $h_{\mathrm{U}}(A, k, p)$ is undefined. However, for every set A such that $|A| = f_k(\mathrm{E}(\alpha \mathrm{U}_I \beta))$, it follows from the definition of f_k that k divides $|A| - 1 - f_k(\beta)$.

Definition 2 (Translation of RTECTLK formulae). *Let φ be a* RTECTLK *formula, and $k \geq 0$ a bound. We define inductively the translation of φ over path number $n \in F_k(\varphi)$ starting at symbolic state $w_{m,n}$ as shown below.*

- $[\mathbf{true}]_k^{[m,n,A]} := \mathbf{true}$, • $[\mathbf{false}]_k^{[m,n,A]} := \mathbf{false}$,
- $[p]_k^{[m,n,A]} := p(w_{m,n})$, • $[\neg p]_k^{[m,n,A]} := \neg p(w_{m,n})$,
- $[\alpha \wedge \beta]_k^{[m,n,A]} := [\alpha]_k^{[m,n,g_l(A,f_k(\alpha))]} \wedge [\beta]_k^{[m,n,g_r(A,f_k(\beta))]}$,
- $[\alpha \vee \beta]_k^{[m,n,A]} := [\alpha]_k^{[m,n,g_l(A,f_k(\alpha))]} \vee [\beta]_k^{[m,n,g_l(A,f_k(\beta))]}$,
- $[\mathrm{EX}\alpha]_k^{[m,n,A]} :=$
  (1) $H(w_{m,n}, w_{0,min(A)}) \wedge [\alpha]_k^{[1,min(A),g_s(A)]}$, if $k > 0$
  (2) $\mathbf{false}$, *otherwise*
- $[\mathrm{E}(\alpha \mathrm{U}_I \beta)]_k^{[m,n,A]} := H(w_{m,n}, w_{0,min(A)}) \wedge \bigvee_{i=0}^{k}([\beta]_k^{[i,min(A),h_{\mathrm{U}}(A,k,f_k(\beta))(k)]}$
  $\wedge\, In(i,I) \wedge \bigwedge_{j=0}^{i-1}[\alpha]_k^{[j,\,min(A),h_{\mathrm{U}}(A,k,f_k(\beta))(j)]})$,
- $[\mathrm{EG}_I\alpha]_k^{[m,n,A]} := H(w_{m,n}, w_{0,min(A)}) \wedge$
  (1) $\bigwedge_{j=left(I)}^{right(I)}[\alpha]_k^{[j,min(A),h_{\mathrm{G}}(A,k)(j)]}$, if $right(I) \leq k$
  (2) $\bigvee_{l=0}^{k-1}(H(w_{k,min(A)}, w_{l,min(A)}) \wedge \bigwedge_{j=min(left(I),l)}^{k-1}[\alpha]_k^{[j,min(A),h_{\mathrm{G}}(A,k)(j)]})$, otherwise.
- $[\overline{\mathrm{K}}_c\alpha]_k^{[m,n,A]} := I_\iota(w_{0,min(A)}) \wedge \bigvee_{j=0}^{k}([\alpha]_k^{[j,min(A),g_s(A)]} \wedge H_c(w_{m,n}, w_{j,min(A)}))$,
- $[\overline{\mathrm{D}}_\Gamma\alpha]_k^{[m,n,A]} := I_\iota(w_{0,min(A)}) \wedge \bigvee_{j=0}^{k}([\alpha]_k^{[j,min(A),g_s(A)]}$
  $\wedge \bigwedge_{c\in\Gamma} H_c(w_{m,n}, w_{j,min(A)}))$,
- $[\overline{\mathrm{E}}_\Gamma\alpha]_k^{[m,n,A]} := I_\iota(w_{0,min(A)}) \wedge \bigvee_{j=0}^{k}([\alpha]_k^{[j,min(A),g_s(A)]}$
  $\wedge \bigvee_{c\in\Gamma} H_c(w_{m,n}, w_{j,min(A)}))$,
- $[\overline{\mathrm{C}}_\Gamma\alpha]_k^{[m,n,A]} := [\bigvee_{j=1}^{k}(\overline{\mathrm{E}}_\Gamma)^j\alpha]_k^{[m,n,A]}$.

The following two lemmas state the correctness and the completeness of the new translation respectively. Lemma 4 claims that if some valuation V satisfies the translation of a RTECTLK formula for some k, then this formula is k-true at the state corresponding to the valuation V. On the other hand, Lemma 5 claims that if a RTECTLK formula is, for some k, k-true at some state of the model, then its translation is a satisfiable propositional formula.

Let $[M]_k^{F_k(\varphi)} = \bigwedge_{j=0}^{f_k(\varphi)-1}\bigwedge_{i=0}^{k-1}\mathcal{R}(w_{i,j}, w_{i+1,j})$. From now on, for every RTECTLK formula φ and every subformula α of φ, we denote by $[\alpha]_k^{[\varphi,m,n,A]}$ the propositional formula $[M]_k^{F_k(\varphi)} \wedge [\alpha]_k^{[m,n,A]}$. We shall write $\sigma \Vdash \xi$ if the valuation σ satisfies the propositional formula ξ. Moreover, we shall write $s \models_k \alpha$ instead of $M, s \models_k \alpha$ and $\boldsymbol{\sigma}_{i,j}$ instead of $\boldsymbol{\sigma}(w_{i,j})$.

Lemma 4 (Correctness of the translation). *Let M be a model, φ be a* RTECTLK *formula and $k \in \mathbb{N}$. Then for every subformula α of the formula φ, every $A \subseteq F_k(\varphi)$ such that $|A| = f_k(\alpha)$, every $(m,n) \in \{(0,0)\} \cup \{0,\ldots,k\} \times \mathbb{N}_+$ and every valuation σ such that $\boldsymbol{\sigma}_{m,n}$ is a state of M the following condition holds: if $\sigma \Vdash [\alpha]_k^{[\varphi,m,n,A]}$, then $M, \boldsymbol{\sigma}_{m,n} \models_k \alpha$.*

Proof. The proof is analogous to the proof of Lemma 3.1 from [22] and use induction on the complexity of α.

Lemma 5 (Completeness of the translation). *Let M be a model, φ be a* RTECTLK *formula and $k \in \mathbb{N}$. Then for every subformula α of the formula φ, every $A \subseteq F_k(\varphi)$ such that $|A| = f_k(\alpha)$, every $(m,n) \in \{(0,0)\} \cup \{0,\ldots,k\} \times (\mathbb{N}_+ \setminus A)$ and every state s of M the following condition holds: if $M, s \models_k \alpha$, then there exists a valuation σ such that $\boldsymbol{\sigma}_{m,n} = s$ and $\sigma \Vdash [\alpha]_k^{[\varphi,m,n,A]}$.*

Proof. The proof is analogous to the proof of Lemma 3.2 from [22] and use induction on the complexity of α.

The next theorem easily follows from Lemmas 4 and 5.

Theorem 2. *Let M be a model, and φ a* RTECTLK *formula. Then for every $k \in \mathbb{N}$, $M \models_k \varphi$ if, and only if, the propositional formula $[M,\varphi]_k$ is satisfiable.*

Proof. Recall that $[M,\varphi]_k = I(w_{0,0}) \wedge [M]_k^{F_k(\varphi)} \wedge [\varphi]_k^{[0,0,F_k(\varphi)]} = I(w_{0,0}) \wedge [\varphi]_k^{[\varphi,0,0,F_k(\varphi)]}$. If $M \models_k \varphi$, then $M, \iota \models_k \varphi$. By Lemma 5, there exists a valuation σ such that $\boldsymbol{\sigma}_{0,0} = \iota$ and $\sigma \Vdash [\varphi]_k^{[\varphi,0,0,F_k(\varphi)]}$. Hence, the propositional formula $[M,\varphi]_k$ is satisfiable.
If the propositional formula $[M,\varphi]_k$ is satisfiable, then there exists a valuation σ such that $\sigma \Vdash I(w_{0,0}) \wedge [\varphi]_k^{[\varphi,0,0,F_k(\varphi)]}$. Thus, $\boldsymbol{\sigma}_{0,0} = \iota$, and by Lemma 4 it follows that $M, \boldsymbol{\sigma}_{0,0} \models_k \varphi$. Hence, $M \models_k \varphi$.

Now, from Theorems 1 and 2 we get the following

Corollary 1. *Let M be a model, and φ a* RTECTLK *formula. Then, $M \models \varphi$ if, and only if, there exists $k \in \mathbb{N}$ such that the propositional formula $[M,\varphi]_k$ is satisfiable.*

4 Experimental Results

In this section we consider two scalable multi-agent systems, and present performance evaluation of both SAT-based BMC algorithms for RTECTLK, the new and the old one. Unfortunately, we cannot compare our experimental results to others, simply because, to the best of our knowledge, there is no tool implementing the BMC method for RTECTLK. In order to appraise the behaviour of our new algorithm, we have tested it on several RTECTLK properties. An evaluation of both BMC algorithms, which have been implemented in C++ under the same semantics (i.e., interleaved interpreted systems), is given by means of the running time, the memory used, the number of generated variables and clauses. All the benchmarks together with an instruction how to reproduce our results can be found at the webpage http://ajd.czest.pl/∼modelchecking/.

For the tests we have used a computer equipped with AMD phenom(tm) 9550 Quad-Core 2200 MHz processor and 4 GB of RAM, running Ubuntu Linux with kernel version 2.6.35-28-generic-pae, and we have set the timeout to 15000 seconds, and memory limit to 3072 MB. We have used the state of the art SAT-solver MiniSat 2 [6,5].

Generic Pipeline Paradigm. The first benchmark we consider is a generic pipeline paradigm (GPP) [16], which consists of three parts: Producer producing data, Consumer receiving data, and a chain of n intermediate Nodes that transmit data produced by Producer to Consumer. The local states for each agent (Producer, Consumer, and intermediate Nodes), and their protocols are shown on Fig. 1. The evaluation of both BMC algorithms for RTECTLK with respect to the GPP system has been done by means of the following RTECTLK specifications:

$\varphi_1 = \mathrm{EF}_{[0,\infty]}\,(ProdSend \wedge \mathrm{EG}_{[a,\infty]}\overline{\mathrm{K}}_C\overline{\mathrm{K}}_P(Received))$, where $a = 2n+1$, $n \geq 1$;
$\varphi_2 = \mathrm{EF}_{[0,\infty]}\overline{\mathrm{K}}_P(ProdSend \wedge \mathrm{EF}_{[0,3]}(Received))$;
$\varphi_3 = \mathrm{EF}_{[0,\infty]}\overline{\mathrm{K}}_P(ProdSend \wedge \mathrm{EF}_{[n,n+3]}(Received))$, where $n \geqslant 1$.

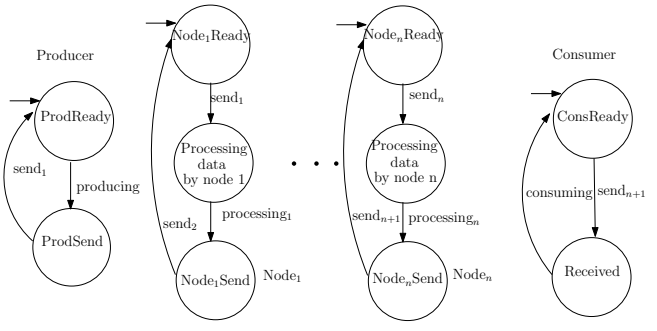


Fig. 1. The GPP system

The formula φ_1 states that it is not true that if Producer generates a product, then ultimately at time later or equal a Consumer knows that Producer does not know that Consumer has the product. The formula φ_2 expresses that it is not true that Producer knows that if he produces a product, then always within the first three time units later Consumer does not have the product. The formula φ_3 represents that it is not true that Producer knows that if he generates a product, then always within interval $[n, n+3]$ Consumer does not have the product. The above formulae are true in the model for GPP.

A train controller system (TC). The second benchmark we consider is a standard train controller (TC) system [12], which consists of n trains (for $n \geq 2$) and a controller. In the system it is assumed that each train uses its own circular track for travelling in one direction. At one point, all trains have to pass through a tunnel, but because there is only one track in the tunnel, trains arriving from each direction cannot use it simultaneously. There are traffic lights on both sides of the tunnel, which can be either red or green. All trains notify the controller when they request entry to the tunnel or when they leave the tunnel. The controller controls the colour of the traffic lights.

The local states for each agent (trains and the controller), and their protocols are shown on Fig. 2. The evaluation of both BMC algorithms for RTECTLK with respect to the TC system has been done by means of the following RTECTLK specifications:

$\varphi_1 = \mathrm{EF}_{[0,\infty]}(\overline{\mathrm{K}}_{Train_1}(InTunnel_1 \wedge \mathrm{EG}_{[1,\infty]}(\neg InTunnel_1)))$,
$\varphi_2 = \mathrm{EF}_{[0,\infty]}(InTunnel_1 \wedge \overline{\mathrm{K}}_{Train_1}(\mathrm{EG}_{[1,n+1]}(\bigwedge_{i=1}^{n}(\neg InTunnel_i))))$,

where n is the number of considered trains.

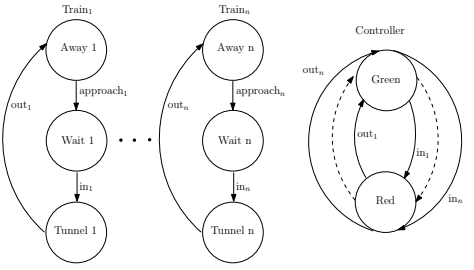
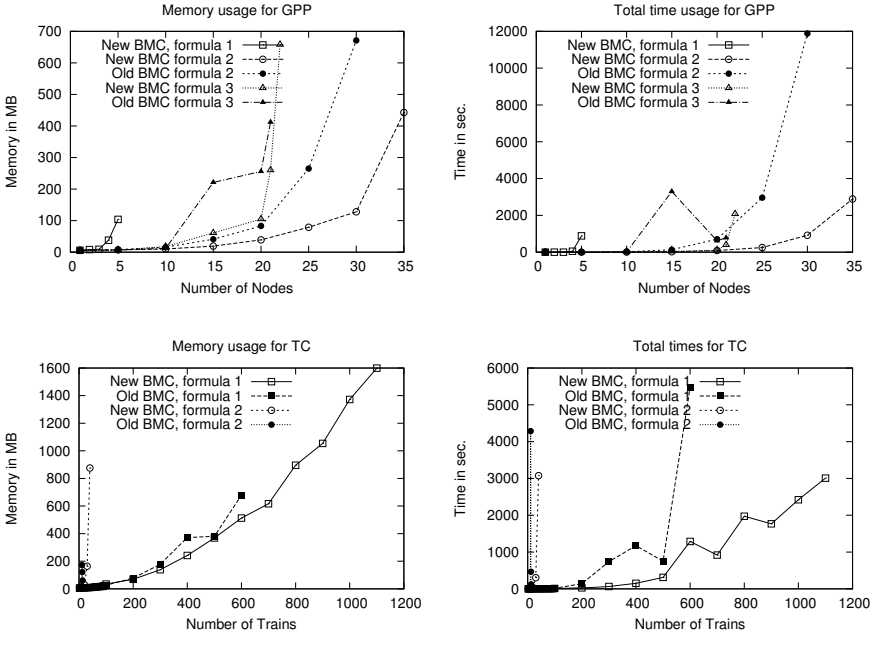


Fig. 2. Agents for train controller system

The formula φ_1 states that it is not true that it is always the case that agent Train 1 knows that whenever he is in the tunnel, it will be in the tunnel once again within a bounded period of time, i.e., within n time units for $n \geq 1$. The formula φ_2 represents that it is not true that it is always the case that if Train 1 is in the tunnel, then he knows that either he or other train will be in the tunnel during the next $n + 1$ time units. All above formulae are true in the model for TC.



Performance Evaluation. The experimental results show that the improved BMC method for RTECTLK outperforms the old BMC method for RTECTLK in both the memory consumption and the execution time (as shown below in the line plots). This is so, because the new method produces a significantly smaller set of clauses (as shown in Table 1), and the SAT solver is given this smaller set. The reason for this is that the new translation does not use all the $f_k(\varphi)$ k-paths both for the formula φ and for each of its proper subformulea. Moreover, the produced set of clauses is not only smaller,

but also easier for a SAT solver, which further boosts the performance of the improved BMC method. Therefore using the old translation only smaller systems can be model-checked. Notice, that for the GPP system it was even impossible to generate the translation of the formula φ_1 by using the old method.

As is well known, the SAT-based BMC method is divided into two steps: in the first step the set of clauses is generated that describes both the checked model and the formula under consideration, and in the second step the SAT solver checks the satisfiability of the generated set of clauses. Both methods, the new and the old, consumed comparable memory in the first step, but the old method took significantly more time in the second step. Also, the memory consumed by the SAT solver was significantly larger for the set of clauses generated by the old method, which suggests that the propositional formula obtained by the old method is bigger and more complicated in comparison with the one generated by the new method.

Table 1. Results for selected witnesses generated by the new and old BMC translations

Checked formula	Which translation	(Max) number of components	Length of the witnesses	Number of paths	Number of variables	Number of clauses
$gpp - \varphi_1$	new	1	5	14	3210	9153
$gpp - \varphi_1$	new	5	13	30	61926	178381
$gpp - \varphi_2$	old	30	63	3	443621	1359604
$gpp - \varphi_2$	new	30	63	3	301286	887929
$gpp - \varphi_2$	new	35	73	3	444741	1312844
$gpp - \varphi_3$	old	21	24	3	95015	290305
$gpp - \varphi_3$	new	21	24	3	65957	192865
$gpp - \varphi_3$	new	22	25	3	73532	215155
$tc - \varphi_1$	old	500	4	3	1716818	5159566
$tc - \varphi_1$	new	500	4	3	1647670	4918036
$tc - \varphi_1$	new	1100	4	3	7584670	22699036
$tc - \varphi_2$	old	13	14	3	29123	89905
$tc - \varphi_2$	new	13	14	3	17140	49765
$tc - \varphi_2$	new	40	41	3	217372	638392

5 Conclusions

In the paper we have presented a compact and elegant bounded semantics for RTECTLK, an improved SAT-based BMC algorithm for RTECTLK properties of interleaved interpreted systems, and we have shown the correctness of both, the new bounded semantics for RTECTLK and the new BMC encoding. Further, we have implemented, tested, and compared with each other on two standard benchmarks both BMC translations for RTECTLK, the new and the old one. Our experiments have shown that the new translation is clearly superior; it is much faster, and consumes less memory.

Our future work will involve an implementation of the method also for other models of multi-agent systems, for example for standard interpreted systems. Moreover, we are going to define a BDD-based BMC algorithm for RTECTLK, and compare it with the method presented in this paper.

References

1. Biere, A., Cimatti, A., Clarke, E., Fujita, M., Zhu, Y.: Symbolic model checking using SAT procedures instead of BDDs. In: Proc. of DAC 1999, pp. 317–320 (1999)
2. Biere, A., Heljanko, K., Junttila, T., Latvala, T., Schuppan, V.: Linear Encodings of Bounded LTL Model Checking. Logical Methods in Computer Science 2(5:5), 1–64 (2006)
3. Blackburn, P., de Rijke, M., Venema, Y.: Modal Logic. Cmbridge Tracts in Theoretical Computer Science, vol. 53. Cambridge University Press (2001)
4. Clarke, E.M., Grumberg, O., Peled, D.A.: Model Checking. The MIT Press, Cambridge (1999)
5. Eén, N., Sörensson, N.: MiniSat, `http://minisat.se/MiniSat.html`
6. Eén, N., Sörensson, N.: MiniSat - A SAT Solver with Conflict-Clause Minimization. In: Bacchus, F., Walsh, T. (eds.) SAT 2005. LNCS, vol. 3569. Springer, Heidelberg (2005)
7. Emerson, E.A., Sistla, A.P., Mok, A.K., Srinivasan, J.: Quantitative temporal reasoning. Real-Time Systems 4(4), 331–352 (1992)
8. Fagin, R., Halpern, J.Y., Moses, Y., Vardi, M.Y.: Reasoning about Knowledge. MIT Press, Cambridge (1995)
9. Fagin, R., Halpern, J.Y., Vardi, M.Y.: What can machines know? On the properties of knowledge in distributed systems. Journal of the ACM 39(2), 328–376 (1992)
10. Halpern, J.Y., Vardi, M.Y.: The complexity of reasoning about knowledge and time 1: lower bounds. Journal of Computer and System Sciences 38(1), 195–237 (1989)
11. van der Hoek, W., Wooldridge, M.J.: Model checking knowledge and time. In: Bošnački, D., Leue, S. (eds.) SPIN 2002. LNCS, vol. 2318, pp. 95–111. Springer, Heidelberg (2002)
12. van der Hoek, W., Wooldridge, M.: Cooperation, knowledge, and time: Alternating-time temporal epistemic logic and its applications. Studia Logica 75(1), 125–157 (2003)
13. Huang, X., Luo, C., van der Meyden, R.: Improved Bounded Model Checking for a Fair Branching-Time Temporal Epistemic Logic. In: van der Meyden, R., Smaus, J.-G. (eds.) MoChArt 2010. LNCS (LNAI), vol. 6572, pp. 95–111. Springer, Heidelberg (2011)
14. Kacprzak, M., Lomuscio, A., Niewiadomski, A., Penczek, W., Raimondi, F., Szreter, M.: Comparing BDD and SAT based techniques for model checking Chaum's dining cryptographers protocol. Fundamenta Informaticae 63(2,3), 221–240 (2006)
15. Lomuscio, A., Penczek, W., Qu, H.: Partial order reduction for model checking interleaved multi-agent systems. In: AAMAS, pp. 659–666. IFAAMAS Press (2010)
16. Peled, D.: All from one, one for all: On model checking using representatives. In: Courcoubetis, C. (ed.) CAV 1993. LNCS, vol. 697, pp. 409–423. Springer, Heidelberg (1993)
17. Penczek, W., Lomuscio, A.: Verifying epistemic properties of multi-agent systems via bounded model checking. Fundamenta Informaticae 55(2), 167–185 (2003)
18. Penczek, W., Woźna, B., Zbrzezny, A.: Bounded model checking for the universal fragment of CTL. Fundamenta Informaticae 51(1-2), 135–156 (2002)
19. Raimondi, F., Lomuscio, A.: Automatic verification of multi-agent systems by model checking via OBDDs. Journal of Applied Logic 5(2), 235–251 (2005); Special issue on Logic-based agent verification
20. van der Meyden, R., Su, K.: Symbolic model checking the knowledge of the dining cryptographers. In: Proc. of CSFW 2004, pp. 280–291. IEEE Computer Society, Los Alamitos (2004)
21. Woźna-Szcześniak, B.: Bounded model checking for the existential part of Real-Time CTL and knowledge. In: Pre-Proc. of CEE-SET 2009, pp. 178–191. AGH Krakow, Poland (2009)
22. Zbrzezny, A.: Improving the translation from ECTL to SAT. Fundamenta Informaticae 85(1-4), 513–531 (2008)