Comparing SAT-based bounded model checking RTECTL and ECTL properties

Agnieszka M. Zbrzezny

IMCS, Jan Długosz University. Al. Armii Krajowej 13/15, 42-200 Częstochowa, Poland.

Abstract

We compare two SAT-based bounded model checking algorithms for properties expressed in the existential fragment of soft real time computation tree logic (RTECTL) and in the existential fragment of computation tree logic (ECTL). To this end we use a faulty train controller system (FTC) and the generic pipeline paradigm (GPP), the classic concurrency problems, which we formalise by means of a finite transition system. We consider several properties of the problems that can be expressed in both RTECTL and ECTL, and we present the performance evaluation of the mentioned BMC methods by means of the running time and the memory used.

1 Introduction

The problem of model checking [1] is to check automatically whether a structure M defines a model for a modal (temporal, epistemic, etc.) formula α. The practical applicability of model checking is strongly limited by the state explosion problem, which means that the number of model states grows exponentially in the size of the system representation. To avoid this problem a number of state reduction techniques and symbolic model checking approaches have been developed, among others, [2, 3, 4, 5].

The SAT-based bounded model checking (BMC) is one of the symbolic model checking technique designed for finding witnesses for existential properties or counterexamples for universal properties. Its main idea is to consider a model reduced to a specific depth. The first BMC method was proposed in [6], and it was designed for linear time properties. Next in [7] the method has been extended to handle branching time properties. Further extensions of the SAT-based BMC method for real-time systems and multi-agent systems can be found, among others, in [8, 9].

The existential fragment of the computation tree logic (ECTL) [7] is a formalism that allow for specification of properties such as "*there is a computation such that α will eventually request*", or "*there is a computation such that α will never be true*", but it is impossible to to directly express bounded properties like for example "*there is a computation such that α will be true in less than 15 unit time*", or "*there is a computation such that α will never be averted after 15 unit time*", or "*there is a computation such that α will always be averted between 10 and 20 unit time*". Note however that this bounded properties can be formalised in ECTL by using nested applications of the next state operators, but the resulting ECTL formulae can be very complicated and problematic to work with. An existential fragment of the soft real-time CTL (RTECTL) [10] defeats this restriction

by introducing time-bounded temporal operators, and it supplies a much more compact and convenient way of expressing time-bounded properties.

The purpose of this paper is to compare the SAT-based bounded model checking of RTECTL properties and the SAT-based bounded model checking of equivalent ECTL properties and to present a correct and complete translation from RTECTL to ECTL.

We have expected that BMC for RTECTL would give better performance than BMC for the ECTL formulae that result from the translation of RTECTL formulae. This is because the size of these ECTL formulae is exponential in the size of the original RTECTL formulae. However, our experiments have shown that for certain RTECTL formulae BMC can be less effective than BMC for their translation into ECTL formulae. These are the formulae for which the interval at the operator $\mathbf{EG}$ is finite: then after the translation to ECTL, $\mathbf{EG}$ operator disappears, and there are more paths, but they are much shorter.

The structure of the paper is as follows. In Section 2 we shortly recall definitions of transition systems and their parallel composition. Then, we present syntax and semantics of RTECTL and ECTL. In Section 3 we define a translation from RTECTL to ECTL, and show its correctness. In section 4 we shortly present BMC technique for RTECTL and ECTL. In Section 5 we present experimental evaluation of the SAT-based BMC for RTECTL [11] and ECTL [12] on equivalent formulae and for a faulty train controller system (FTC) and generic pipeline paradigm (GPP). In Section 6 we conclude the paper.

2 Preliminaries

In this section we first define Transition System (TS), then we recall syntax and semantics of two logics ECTL and RTECTL.

2.1 Transition System

A Transition System [13] (also called a *model*) is a tuple $\mathcal{M} = (S, Act, \longrightarrow, s^0, AP, L)$ where:

- S is a set of states,
- Act is a set of actions,
- $\longrightarrow \subseteq S \times Act \times S$ is a transition relation,
- $s^0 \in S$ is the initial state,
- AP is a set of atomic propositions, and
- $L : S \to 2^{AP}$ is a labelling function.

Transition system is called finite if S, Act, and AP are finite. For convenience, we write $s \xrightarrow{\sigma} s'$ instead of $(s, \sigma, s) \in \longrightarrow$. Moreover, we write $s \longrightarrow s'$ if $s \xrightarrow{\sigma} s'$, for some $\sigma \in Act$.

From now on we assume that a considered model has no terminal states, i.e. for every $s \in S$ there exist $s' \in S$ such that $s \longrightarrow s'$. The set of all natural numbers is denoted by $\mathbb{N}$, and the set of all positive natural numbers by $\mathbb{N}_+$. A *path* in $\mathcal{M}$ is an infinite sequence $\pi = (s_0, s_1, \ldots)$ of states such that $s_i \longrightarrow s_{i+1}$ for each $i \in \mathbb{N}$. For a path $\pi = (s_0, s_1, \ldots)$ and $i \in \mathbb{N}$, the i-th state of π is defined as $\pi(i) = s_i$. The i-th prefix of π, denoted by $\pi[..i]$ is defined as $\pi[..i] = (s_0, s_1, \ldots, s_i)$, and the i-th suffix of π, denoted by π^i, is defined as $\pi^i = (s_i, s_{i+1}, \ldots)$. Note that if π is a path in $\mathcal{M}$ then the suffix π^i is also

a path in $\mathcal{M}$. By $\Pi(s)$ we denote the set of all the paths starting at $s \in S$, and by $\Pi(\mathcal{M})$ we denote the set of all the paths in $\mathcal{M}$.

2.2 Models and parallel composition

The concurrent systems are designed as collections of interacting computational processes that may be executed in parallel. Therefore, we assume that a concurrent system is modelled as a network of models that run in parallel and communicate with each other via executing shared actions. There are several ways of defining a parallel composition. We adapt the standard definition, namely, in the parallel composition the transitions not corresponding to a shared action are interleaved, whereas the transitions labelled with a shared action are synchronised.

We formalise the above by the following definition of parallel composition. Let $\mathcal{M}_i = (S_i, Act_i, \longrightarrow_i, s_i^0, AP_i, L_i)$ be models such that for all $i, j \in \{1, \ldots, m\}$, if $i \neq j$, then $AP_i \cap AP_j = \emptyset$. We take $Act = \bigcup_{i=1}^{m} Act_i$, and for $\sigma \in Act$ we define a set $Act(\sigma) = \{1 \leq i \leq m \mid \sigma \in Act_i\}$ that gives the indices of the components that synchronise at σ. *A parallel composition of m models* $\mathcal{M}_i$ is the model $\mathcal{M} = (S, Act, \longrightarrow, s^0, AP, L)$, where $S = \prod_{i=1}^{m} S_i$, $Act = \bigcup_{i=1}^{m} Act_i$, a transition $((s_1, \ldots, s_m), \sigma, (s'_1, \ldots, s'_m)) \in \longrightarrow$ iff $(\forall j \in Act(\sigma))\ (s_j, \sigma, s'_j) \in \longrightarrow_j$, $(\forall i \in \{1, \ldots, m\} \setminus Act(\sigma))\ s'_i = s_i$, $s^0 = (s_1^0, \ldots, s_m^0)$, $AP = \bigcup_{i=1}^{m} AP_i$, and $L((s_1, \ldots, s_m)) = \bigcup_{i=1}^{m} L_i(s_i)$.

2.3 The ECTL and RTECTL logics

2.3.1 Syntax of ECTL

The syntax of ECTL formulae over the set AP of atomic propositions is defined by the following grammar:

$$\varphi ::= \mathbf{true} \mid \mathbf{false} \mid p \mid \neg p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \mathbf{EX}\varphi \mid \mathbf{E}(\varphi \mathbf{U} \varphi) \mid \mathbf{EG}\varphi$$

where $p \in AP$ and φ is a formula. The symbols $\mathbf{X}$, $\mathbf{U}$ and $\mathbf{G}$ are the modal operators for "neXt time", "Until" and "Globally", respectively. The symbol $\mathbf{E}$ is the existential path quantifier.

The derived basic modalities are defined as follows:

$$\mathbf{EF}\alpha \stackrel{df}{=} \mathbf{E}(\mathbf{true}\,\mathbf{U}\,\alpha), \qquad \mathbf{E}(\alpha\,\mathbf{R}\,\beta) \stackrel{df}{=} \mathbf{E}(\beta\,\mathbf{U}\,(\alpha \wedge \beta)) \vee \mathbf{EG}\beta.$$

2.3.2 Semantics of ECTL

Let $\mathcal{M}$ be a model, and φ an ECTL formula. An ECTL formula φ is *true* in the model $\mathcal{M}$ (in symbols $\mathcal{M} \models \varphi$) iff $\mathcal{M}, s^0 \models \varphi$ (i.e., φ is true at the initial state of the model $\mathcal{M}$), where

$$\begin{aligned}
&\mathcal{M}, s \models \mathbf{true}, \\
&\mathcal{M}, s \not\models \mathbf{false}, \\
&\mathcal{M}, s \models p && \text{iff } p \in L(s), \\
&\mathcal{M}, s \models \neg p && \text{iff } p \notin L(s), \\
&\mathcal{M}, s \models \alpha \wedge \beta && \text{iff } \mathcal{M}, s \models \alpha \text{ and } \mathcal{M}, s \models \beta,
\end{aligned}$$

$$
\begin{array}{ll}
\mathcal{M}, s \models \alpha \vee \beta & \text{iff } \mathcal{M}, s \models \alpha \text{ or } \mathcal{M}, s \models \beta, \\
\mathcal{M}, s \models \mathbf{EX}\alpha & \text{iff } (\exists \pi \in \Pi(s))(\mathcal{M}, \pi(1) \models \alpha), \\
\mathcal{M}, s \models \mathbf{E}(\alpha \mathbf{U} \beta) & \text{iff } (\exists \pi \in \Pi(s))(\exists m \geq 0)(\mathcal{M}, \pi(m) \models \beta \text{ and} \\
 & \quad (\forall j < m)\mathcal{M}, \pi(j) \models \alpha), \\
\mathcal{M}, s \models \mathbf{EG}\alpha & \text{iff } (\exists \pi \in \Pi(s))(\forall m \geq 0)(\mathcal{M}, \pi(m) \models \alpha).
\end{array}
$$

2.3.3 Syntax of RTECTL

Let $p \in AP$ and I be an interval in $\mathbb{N} = \{0, 1, 2, \ldots\}$ of the form: $[a, b)$ and $[a, \infty)$, for $a, b \in \mathbb{N}$. Note that the remaining forms of intervals can be defined by means of $[a, b)$ and $[a, \infty)$.

The language RTECTL is defined by the following grammar:

$$\varphi ::= \mathbf{true} \mid \mathbf{false} \mid p \mid \neg p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \mathbf{EX}\varphi \mid \mathbf{E}(\varphi \mathbf{U}_I \varphi) \mid \mathbf{EG}_I \varphi$$

The derived basic modalities are defined as follows:

$$\mathbf{EF}_I \alpha \stackrel{df}{=} \mathbf{E}(\mathbf{true}\, \mathbf{U}_I\, \alpha), \qquad \mathbf{E}(\alpha\, \mathbf{R}_I\, \beta) \stackrel{df}{=} \mathbf{E}(\beta\, \mathbf{U}_I\, (\alpha \wedge \beta)) \vee \mathbf{EG}_I \beta.$$

2.3.4 Semantics of RTECTL

Let $\mathcal{M}$ be a model and φ an RTECTL formula. An RTECTL formula φ is *true* in the model $\mathcal{M}$ (in symbols $\mathcal{M} \models_{rt} \varphi$) iff $\mathcal{M}, s^0 \models_{rt} \varphi$ (i.e., φ is true at the initial state of the model $\mathcal{M}$), where:

$$
\begin{array}{ll}
\mathcal{M}, s \models_{rt} \mathbf{true}, & \\
\mathcal{M}, s \not\models_{rt} \mathbf{false}, & \\
\mathcal{M}, s \models_{rt} p & \text{iff } p \in L(s), \\
\mathcal{M}, s \models_{rt} \neg p & \text{iff } p \notin L(s), \\
\mathcal{M}, s \models_{rt} \alpha \wedge \beta & \text{iff } \mathcal{M}, s \models_{rt} \alpha \text{ and } \mathcal{M}, s \models_{rt} \beta, \\
\mathcal{M}, s \models_{rt} \alpha \vee \beta & \text{iff } \mathcal{M}, s \models_{rt} \alpha \text{ or } \mathcal{M}, s \models_{rt} \beta, \\
\mathcal{M}, s \models_{rt} \mathbf{EX}\alpha & \text{iff } (\exists \pi \in \Pi(s))(\mathcal{M}, \pi(1) \models_{rt} \alpha), \\
\mathcal{M}, s \models_{rt} \mathbf{E}(\alpha \mathbf{U}_I \beta) & \text{iff } (\exists \pi \in \Pi(s))(\exists m \in I)(\mathcal{M}, \pi(m) \models_{rt} \beta \text{ and} \\
 & \quad (\forall i < m)\mathcal{M}, \pi(i) \models_{rt} \alpha), \\
\mathcal{M}, s \models_{rt} \mathbf{EG}_I \alpha & \text{iff } (\exists \pi \in \Pi(s))(\forall m \in I)(\mathcal{M}, \pi(m) \models_{rt} \alpha).
\end{array}
$$

3 Translation from RTECTL into ECTL

In this section we present a translation from the RTECTL language to the ECTL language that is based on the intuitive description of such a translation presented in [10, 14]. We focused on the existential part of RTCTL only, because we use the BMC method. Moreover, we would like to stress that our semantics of the operator $\mathbf{G}_I$ is different than the one presented in [10, 14]. Further, unlike in [10, 14], we present the translation for all types of intervals that can appear together with the temporal operators $\mathbf{U}_I$ and $\mathbf{G}_I$, and for both operators.

Let $p \in AP$, α and β be formulae of RTECTL, $a \in \mathbb{N}$, $b \in \mathbb{N} \cup \{\infty\}$ and $a < b$. We define the translation from RTECTL into ECTL as a function tr : RTECTL $\rightarrow$ ECTL in the following way:

$$\begin{aligned} tr(p) &= p \\ tr(\neg p) &= \neg p \\ tr(\alpha \wedge \beta) &= tr(\alpha) \wedge tr(\beta) \\ tr(\alpha \vee \beta) &= tr(\alpha) \vee tr(\beta) \end{aligned}$$

$$tr(\mathbf{E}(\alpha \mathbf{U}_{[a,b)}\beta)) = \begin{cases} tr(\alpha) \wedge \mathbf{EX}(tr(\mathbf{E}(\alpha \mathbf{U}_{[a-1,b-1)}\beta))) & \text{if } a > 0 \text{ and } 1 < b < \infty \\ tr(\beta) \vee tr(\alpha) \wedge \mathbf{EX}(tr(\mathbf{E}(\alpha \mathbf{U}_{[0,b-1)}\beta))) & \text{if } a = 0 \text{ and } 1 < b < \infty \\ tr(\alpha) \wedge \mathbf{EX}(tr(\mathbf{E}(\alpha \mathbf{U}_{[a-1,\infty)}\beta))) & \text{if } a > 0 \text{ and } b = \infty \\ \mathbf{E}(tr(\alpha)\mathbf{U}tr(\beta)) & \text{if } a = 0 \text{ and } b = \infty \\ tr(\beta) & \text{if } a = 0 \text{ and } b = 1 \end{cases}$$

$$tr(\mathbf{EG}_{[a,b)}\alpha) = \begin{cases} \mathbf{EX}(tr(\mathbf{EG}_{[a-1,b-1)}\alpha)) & \text{if } a > 0 \text{ and } 1 < b < \infty \\ tr(\alpha) \wedge \mathbf{EX}(tr(\mathbf{EG}_{[a,b-1)}\alpha)) & \text{if } a = 0 \text{ and } 1 < b < \infty \\ \mathbf{EX}(tr(\mathbf{EG}_{[a-1,\infty)}\alpha)) & \text{if } a > 0 \text{ and } b = \infty \\ \mathbf{EG}tr(\alpha) & \text{if } a = 0 \text{ and } b = \infty \\ tr(\alpha) & \text{if } a = 0 \text{ and } b = 1 \end{cases}$$

$$tr(\mathbf{EX}\alpha) = \mathbf{EX}tr(\alpha)$$

Because $\mathbf{EF}_{[a,b)}\alpha \stackrel{df}{=} \mathbf{E}(true\mathbf{U}_{[a,b)}\alpha)$, the translation for $\mathbf{EF}_{[a,b)}\alpha$ can be defined using the translation for $\mathbf{E}(\alpha \mathbf{U}_{[a,b)}\beta)$:

$$tr(\mathbf{EF}_{[a,b)}\alpha) = \begin{cases} \mathbf{EX}(tr(\mathbf{EF}_{[a-1,b-1)}\alpha)) & \text{if } a > 0 \text{ and } 1 < b < \infty \\ tr(\alpha) \vee \mathbf{EX}(tr(\mathbf{EF}_{[0,b-1)}\alpha)) & \text{if } a = 0 \text{ and } 1 < b < \infty \\ \mathbf{EX}(tr(\mathbf{EF}_{[a-1,\infty)}\alpha)) & \text{if } a > 0 \text{ and } b = \infty \\ \mathbf{EF}tr(\alpha) & \text{if } a = 0 \text{ and } b = \infty \\ tr(\alpha) & \text{if } a = 0 \text{ and } b = 1 \end{cases}$$

The following theorem, which can be proven by induction on the length of an RTECTL formula, states correctness of the above translation.

Proposition 1. *Let α be an* RTECTL *formula, and $a \in \mathbb{N}$. Then,*

$$tr(\mathbf{EG}_{[a,\infty)}\alpha)) = \underbrace{\mathbf{EX} \ldots \mathbf{EX}}_{a}(\mathbf{EG}tr(\alpha)).$$

Example 1. $tr(\mathbf{EG}_{[3,\infty)}\alpha) = \mathbf{EXEXEX}(\mathbf{EG}tr(\alpha))$.

Proposition 2. *Let α be an* RTECTL *formula, and $a, b \in \mathbb{N}$. If $a < b < \infty$, then*

$$tr(\mathbf{EG}_{[a,b)}\alpha)) = \underbrace{\mathbf{EX}\ldots\mathbf{EX}}_{a}(tr(\alpha) \wedge \mathbf{EX}(tr(\alpha)) \wedge \ldots \wedge \underbrace{\mathbf{EX}\ldots\mathbf{EX}}_{b-a}(tr(\alpha))).$$

Example 2. $tr(\mathbf{EG}_{[3,6)}\alpha) = \mathbf{EXEXEX}(tr(\alpha) \wedge \mathbf{EX}(tr(\alpha) \wedge \mathbf{EX}(tr(\alpha))))$.

Proposition 3. *Let α and β be* RTECTL *formulae, and $a \in \mathbb{N}$. If $a > 0$, then*

$$tr(\mathbf{E}(\alpha\mathbf{U}_{[a,\infty)}\beta)) = tr(\alpha) \wedge \underbrace{\mathbf{EX}(tr(\alpha) \wedge \mathbf{EX}(tr(\alpha)\ldots}_{a-1} \wedge \mathbf{E}(tr(\alpha)\mathbf{U}tr(\beta)))).$$

Example 3. $tr(\mathbf{E}(\alpha\mathbf{U}_{[3,\infty)}\beta)) = tr(\alpha) \wedge \mathbf{EX}(tr(\alpha) \wedge \mathbf{EX}(tr(\alpha) \wedge \mathbf{E}(tr(\alpha)\mathbf{U}tr(\beta))))$.

Proposition 4. *Let α and β be* RTECTL *formulae, and $a, b \in \mathbb{N}$. If $a < b < \infty$, then*

$$tr(\mathbf{E}(\alpha\mathbf{U}_{[a,b)}\beta)) = tr(\alpha) \wedge \underbrace{\mathbf{EX}(tr(\alpha) \wedge \mathbf{EX}(tr(\alpha) \wedge \ldots}_{a}$$
$$\wedge \underbrace{\mathbf{EX}(tr(\beta) \vee tr(\alpha) \wedge \mathbf{EX}(tr(\beta) \vee tr(\alpha)\ldots}_{b-a-1} \wedge \mathbf{EX}(tr(\beta)))))))).$$

Example 4. $tr(\mathbf{E}(\alpha\mathbf{U}_{[3,6)}\beta)) = tr(\alpha) \wedge \mathbf{EX}(tr(\alpha) \wedge \mathbf{EX}(tr(\alpha) \wedge \mathbf{EX}(tr(\alpha) \wedge$
$\mathbf{EX}(tr(\beta) \vee tr(\alpha) \wedge \mathbf{EX}(tr(\beta) \vee tr(\alpha) \wedge \mathbf{EX}(tr(\beta)))))))$.

Lemma 1. *Let $\mathcal{M}$ be a model, and φ an* RTECTL *formula. Then* $(\forall s \in S)(\mathcal{M}, s \models_{rt} \varphi \Rightarrow \mathcal{M}, s \models tr(\varphi))$

Proof. We proceed by induction on the length of formulae. Let $s \in S$, and assume that $\mathcal{M}, s \models_{rt} \varphi$. Now consider the following cases:

1. $\varphi \in AP$. Then, since $tr(\varphi) = \varphi$, we have that $tr(\varphi) \in AP$. Therefore, $\mathcal{M}, s \models_{rt} \varphi \iff \varphi \in L(s) \iff tr(\varphi) \in L(s) \iff \mathcal{M}, s \models tr(\varphi)$.
2. $\varphi = \neg p$, where $p \in AP$. Then $tr(\varphi) = \varphi$. Therefore, $\mathcal{M}, s \models_{rt} \varphi \iff \mathcal{M}, s \models_{rt} \neg p \iff p \notin L(s) \iff \mathcal{M}, s \models \neg p \iff \mathcal{M}, s \models \varphi \iff \mathcal{M}, s \models tr(\varphi)$.
3. $\varphi = \alpha \wedge \beta$. By the definition of the satisfiability relation we have that $\mathcal{M}, s \models_{rt} \alpha$ and $\mathcal{M}, s \models_{rt} \beta$. By the inductive hypothesis, we get $\mathcal{M}, s \models tr(\alpha)$ and $\mathcal{M}, s \models tr(\beta)$. Thus, $\mathcal{M}, s \models tr(\alpha) \wedge tr(\beta)$, and therefore $\mathcal{M}, s \models tr(\alpha \wedge \beta)$.
4. $\varphi = \alpha \vee \beta$. By the definition of the satisfiability relation we have that $\mathcal{M}, s \models_{rt} \alpha$ or $\mathcal{M}, s \models_{rt} \beta$. By the inductive hypothesis, we get $\mathcal{M}, s \models tr(\alpha)$ or $\mathcal{M}, s \models tr(\beta)$. Thus, $\mathcal{M}, s \models tr(\alpha) \vee tr(\beta)$, and therefore $\mathcal{M}, s \models tr(\alpha \vee \beta)$.
5. $\varphi = \mathbf{EX}\alpha$. By the definition of the satisfiability relation we have that there exists $\pi \in \Pi(s)$ such that $\mathcal{M}, \pi(1) \models_{rt} \alpha$. By the inductive hypothesis, we conclude that $\mathcal{M}, \pi(1) \models tr(\alpha)$. Thus, $\mathcal{M} \models \mathbf{EX}tr(\alpha)$, since $\pi(0) = s$. Therefore $\mathcal{M} \models tr(\mathbf{EX}\alpha)$, since $tr(\mathbf{EX}\alpha) = \mathbf{EX}tr(\alpha)$.

6. $\varphi = \mathbf{EG}_I\alpha$. By the definition of the satisfiability relation we have that there exists $\pi \in \Pi(s)$ such that $(\forall m \in I)(\mathcal{M}, \pi(m) \models_{rt} \alpha)$. By the inductive hypothesis, we conclude that there exists $\pi \in \Pi(s)$ such that

$$(\forall m \in I)(\mathcal{M}, \pi(m) \models tr(\alpha)). \tag{1}$$

Now consider the following cases:

(a) $I = [a, \infty)$. From (1) we conclude that $\mathcal{M}, s \models \underbrace{\mathbf{EX} \ldots \mathbf{EX}}_{a}(\mathbf{EG}tr(\alpha))$. From this, by Proposition 1, it follows that $\mathcal{M}, s \models tr(\mathbf{EG}_{[a,\infty)}\alpha)$.

(b) $I = [a, b)$, where $a < b < \infty$. From (1) we conclude that

$$\mathcal{M}, s \models \underbrace{\mathbf{EX} \ldots \mathbf{EX}}_{a}(tr(\alpha) \wedge \mathbf{EX}(tr(\alpha)) \wedge \ldots \wedge \underbrace{\mathbf{EX} \ldots \mathbf{EX}}_{b-a}(tr(\alpha))).$$

From this, by Proposition 2, we get $\mathcal{M}, s \models tr(\mathbf{EG}_{[a,b)}\alpha))$.

7. $\varphi = \mathbf{E}(\alpha \mathbf{U}_I \beta)$. By the definition of the satisfiability relation we have that there exists $\pi \in \Pi(s)$ such that $(\exists m \in I)(\mathcal{M}, \pi(m) \models_{rt} \beta$ and $(\forall i < m)(\mathcal{M}, \pi(i) \models_{rt} \alpha)$. By the inductive hypothesis, we conclude that

$$(\exists m \in I)(\mathcal{M}, \pi(m) \models tr(\beta) \text{ and } (\forall i < m)(\mathcal{M}, \pi(i) \models tr(\alpha))) \tag{2}$$

Now consider the following cases:

(a) $I = [a, \infty)$. From (2) we conclude that

$$\mathcal{M}, s \models tr(\alpha) \wedge \underbrace{\mathbf{EX}(tr(\alpha) \wedge \mathbf{EX}(tr(\alpha) \ldots}_{a-1} \wedge \mathbf{E}(tr(\alpha) \mathbf{U} tr(\beta)))).$$

From this, by Proposition 3, it follows that $\mathcal{M}, s \models tr(\mathbf{E}(\alpha \mathbf{U}_{[a,\infty)}\beta))$.

(b) $I = [a, b)$, where $a < b < \infty$. From (2) we conclude that

$$\mathcal{M}, s \models tr(\alpha) \wedge \underbrace{\mathbf{EX}(tr(\alpha) \wedge \mathbf{EX}(tr(\alpha) \wedge \ldots}_{a}$$

$$\wedge \underbrace{\mathbf{EX}(tr(\beta) \vee tr(\alpha) \wedge \mathbf{EX}(tr(\beta) \vee tr(\alpha) \ldots}_{b-a-1} \wedge \mathbf{EX}(tr(\beta)))))))).$$

From this, by Proposition 4, it follows that $\mathcal{M}, s \models tr(\mathbf{E}(\alpha \mathbf{U}_{[a,b)}\beta))$.

□

Lemma 2. *Let $\mathcal{M}$ be a model, and φ an* RTECTL *formula. Then* $(\forall s \in S)(\mathcal{M}, s \models tr(\varphi) \Rightarrow \mathcal{M}, s \models_{rt} \varphi)$.

Proof. We proceed by induction on the length of formulae. Let $s \in S$, and assume that $\mathcal{M}, s \models tr(\varphi)$. Now consider the following cases:

1. $\varphi \in AP$. Then, since $\varphi = tr(\varphi)$, we have that $tr(\varphi) \in AP$. Therefore, $\mathcal{M}, s \models tr(\varphi) \iff tr(\varphi) \in L(s) \iff \varphi \in L(s) \iff \mathcal{M}, s \models_{rt} \varphi$.
2. $\varphi = \neg p$, where $p \in AP$. Then $tr(\varphi) = \varphi$. Therefore, $\mathcal{M}, s \models tr(\varphi) \iff \mathcal{M}, s \models tr(\neg p) \iff p \notin L(s) \iff \mathcal{M}, s \models_{rt} \neg p \iff \mathcal{M}, s \models_{rt} \varphi$.
3. $\varphi = \alpha \wedge \beta$. Then, $tr(\varphi) = tr(\alpha \wedge \beta) = tr(\alpha) \wedge tr(\beta)$. By the definition of the satisfiability relation for ECTL we have that $\mathcal{M}, s \models tr(\alpha)$ and $\mathcal{M}, s \models tr(\beta)$. By the inductive hypothesis, we get $\mathcal{M}, s \models_{rt} \alpha$ and $\mathcal{M}, s \models_{rt} \beta$, and therefore $\mathcal{M}, s \models_{rt} (\alpha \wedge \beta)$.

4. $\varphi = \alpha \vee \beta$. Then, $tr(\varphi) = tr(\alpha \vee \beta) = tr(\alpha) \vee tr(\beta)$. By the definition of the satisfiability relation for ECTL we have that $\mathcal{M}, s \models tr(\alpha)$ or $\mathcal{M}, s \models tr(\beta)$. By the inductive hypothesis, we get $\mathcal{M}, s \models_{rt} \alpha$ or $\mathcal{M}, s \models_{rt} \beta$, and therefore $\mathcal{M}, s \models_{rt} (\alpha \vee \beta)$.
5. $\varphi = \mathbf{EX}\alpha$. Then, $tr(\varphi) = tr(\mathbf{EX}\alpha) = \mathbf{EX}tr(\alpha)$. By the definition of the satisfiability relation we have that there exists $\pi \in \Pi(s)$ such that $\mathcal{M}, \pi(1) \models tr(\alpha)$. By the inductive hypothesis, we conclude that $\mathcal{M}, \pi(1) \models_{rt} \alpha$. Thus, $\mathcal{M}, s \models_{rt} \mathbf{EX}\alpha$, since $\pi(0) = s$.
6. $\varphi = \mathbf{EG}_I\alpha$. Consider the following cases:
   (a) $I = [a, \infty)$. From Proposition 1, we get that $\mathcal{M}, s \models \underbrace{\mathbf{EX} \ldots \mathbf{EX}}_{a}(\mathbf{EG}tr(\alpha))$.
   From this it follows that there exists $\pi \in \Pi(s)$ such that $(\forall m \geqslant a)$ $(\mathcal{M}, \pi(m) \models tr(\alpha))$. By the inductive hypothesis, we conclude that $(\forall m \geqslant a)(\mathcal{M}, \pi(m) \models_{rt} \alpha)$, and therefore $\mathcal{M}, s \models_{rt} \mathbf{EG}_I\alpha$.
   (b) $I = [a, b)$, where $a < b < \infty$. From Proposition 2, we get that
   $$\mathcal{M}, s \models \underbrace{\mathbf{EX} \ldots \mathbf{EX}}_{a}(tr(\alpha) \wedge \mathbf{EX}(tr(\alpha)) \wedge \ldots \wedge \underbrace{\mathbf{EX} \ldots \mathbf{EX}}_{b-a}(tr(\alpha))).$$
   From this it follows that there exists $\pi \in \Pi(s)$ such that $(\forall m \in [a, b))$ $(\mathcal{M}, \pi(m) \models tr(\alpha))$. By the inductive hypothesis, we conclude that $(\forall m \in [a, b))(\mathcal{M}, \pi(m) \models_{rt} \alpha)$. Therefore $\mathcal{M}, s \models_{rt} \mathbf{EG}_I\alpha$.
7. $\varphi = \mathbf{E}(\alpha \mathbf{U}_I \beta)$. Consider the following cases:
   (a) $I = [a, \infty)$. From Proposition 3, we get that $\mathcal{M}, s \models tr(\alpha) \wedge \underbrace{\mathbf{EX}(tr(\alpha) \wedge \mathbf{EX}(tr(\alpha) \ldots}_{a-1} \wedge \mathbf{E}(tr(\alpha)\mathbf{U}tr(\beta)))$. From this it follows that there exists $\pi \in \Pi(s)$ such that $(\forall m < a)(\mathcal{M}, \pi(m) \models tr(\alpha))$ and $(\exists m \geqslant a)(\mathcal{M}, \pi(m) \models \mathbf{E}(tr(\alpha)\mathbf{U}tr(\beta))$ By the inductive hypothesis, we conclude that $(\forall m < a)(\mathcal{M}, \pi(m) \models_{rt} \alpha)$ and $(\exists m \geqslant a)(\mathcal{M}, \pi(m) \models_{rt} \beta)$, and therefore $\mathcal{M}, s \models \mathbf{E}(\alpha \mathbf{U}_I \beta)$.
   (b) $I = [a, b)$, where $a < b < \infty$. From Proposition 4, we get that
   $$\mathcal{M}, s \models tr(\alpha) \wedge \underbrace{\mathbf{EX}(tr(\alpha) \wedge \mathbf{EX}(tr(\alpha) \wedge \ldots}_{a}$$
   $$\wedge \underbrace{\mathbf{EX}(tr(\beta) \vee tr(\alpha) \wedge \mathbf{EX}(tr(\beta) \vee tr(\alpha) \ldots}_{b-a-1} \wedge \mathbf{EX}(tr(\beta)))))))).$$
   From this it follows that there exists $\pi \in \Pi(s)$ such that $(\forall m < a)$ $(\mathcal{M}, \pi(m) \models tr(\alpha))$ and $(\exists b - a - 1 > m > a)(\mathcal{M}, \pi(m) \models tr(\alpha) \vee tr(\beta))$ and $(\exists m < b)(\mathcal{M}, \pi(m) \models tr(\beta))$ By the inductive hypothesis, we conclude that $(\forall m < a)(\mathcal{M}, \pi(m) \models_{rt} \alpha)$ and $(\exists b - a - 1 > m > a)(\mathcal{M}, \pi(m) \models_{rt} \alpha \vee \beta)$ and $(\exists m < b)(\mathcal{M}, \pi(m) \models_{rt} \beta)$, and therefore $\mathcal{M}, s \models \mathbf{E}(\alpha \mathbf{U}_I \beta)$.

□

From the Lemma 1 and Lemma 2 we get the following theorem.

Theorem 1 (Correctness of the translation)**.** *Let $\mathcal{M}$ be a model, and φ an* RTECTL *formula. Then $\mathcal{M} \models_{rt} \varphi$ if, and only if $\mathcal{M} \models tr(\varphi)$.*

4 Bounded Model Checking

The SAT-based Bounded Model Checking (BMC) is a popular model checking technique for the verification of concurrent systems. Given a model $\mathcal{M}$, an existential modal formula φ, and a non-negative bound k, the SAT-based BMC consists in searching for a non-empty set of paths of length k that constitute a witness for the checked property φ. In particular, the BMC algorithms generate a propositional formula which is satisfiable if and only if the mentioned set of paths exist. The propositional formula is usually obtained as a combination of a propositional encoding of the unfolding of the transition relation of the given model, and a propositional encoding of the property in question. If the generated propositional formula is not satisfiable, then k is incremented until either the problem becomes intractable due to the complexity of solving the corresponding SAT instance, or k reaches the upper bound of the bounded model checking problem for the language under consideration.

All the SAT-based BMC, so the one for ECTL and RTECTL as well, are based on so called bounded semantics, which are the base of translations of specifications to the SAT-problem. In definitions of the bounded semantics one needs to represent cycles in models in a special way. To this aim k-paths, i.e., finite paths of length k, and loops are defined. These definitions have evolved over the last decade, and they have had a major impact on the effectiveness of the BMC encodings.

The SAT-based BMC method for ECTL was introduced in [7], and then it was improved in [12]. In this paper we use the definition and implementation of the SAT-based BMC for ECTL that was presented in [12]. The SAT-based BMC method for RTECTL was introduced in [15], and then it was improved in [11]. In this paper we use the definition and implementation of the SAT-based BMC for RTECTL that was presented in [11].

5 Experimental Results

In this section we present a comparison of a performance evaluation of two SAT-based BMC algorithms: for RTECTL [11, 16], and for ECTL [12]. In order to evaluate the behaviour of the algorithms, we have tested it on several RTECTL properties and equivalent ECTL properties.

An evaluation of both BMC algorithms, which have been implemented in C++ is given by means of the running time, the memory used, and the number of generated variables and clauses.

5.1 A Faulty Train Controller System

To evaluate the BMC technique for RTECTL and ECTL we analyse a scalable concurrent system, which is a faulty train controller system (FTC) (adapted from [17]). The system consists of a controller, and n trains (for $n \geq 2$), and it is assumed that each train uses its own circular track for travelling in one direction. All trains have to pass through a tunnel, but because there is only one track in the tunnel, arriving trains cannot use it simultaneously. There are signal 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 signal lights, however it can

be faulty, and does not fulfil its task. The controller does not ensure the mutual exclusion property: two trains never occupy the tunnel at the same time.

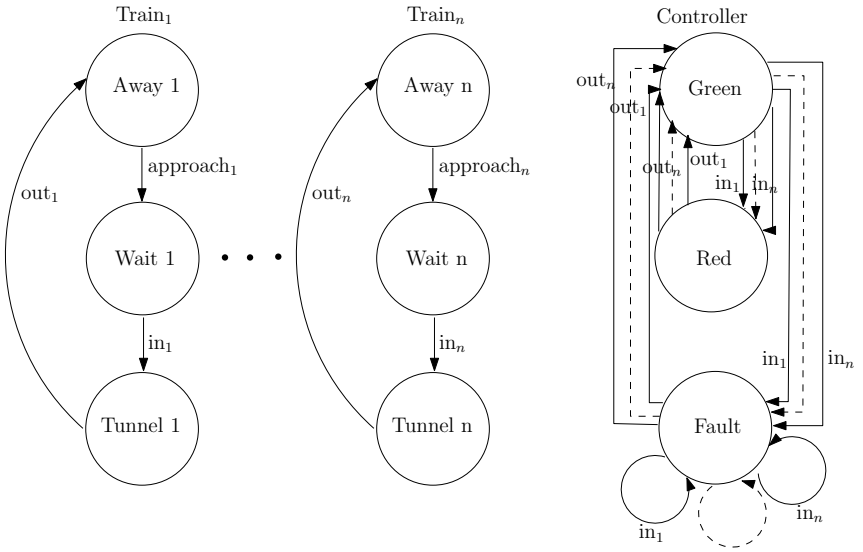


FIGURE 1: A network of automata for train controller system ([16])

An automata model of the FTC system is shown on Figure 1. The specifications for it are given in the existential form, i.e., they are expressed in the RTECTL language:

- $\varphi_1 = \mathbf{EF}_{[0,\infty)}\left(InTunnel_1 \wedge \mathbf{EG}_{[1,n+2)}(\bigwedge_{i=1}^{n} \neg InTunnel_i)\right)$, where n is the number of trains.
- $\varphi_2 = \mathbf{EF}_{[0,\infty)}\left(InTunnel_1 \vee \mathbf{EG}_{[1,n+2)}(\bigvee_{i=1}^{n} \neg InTunnel_i)\right)$, where n is the number of trains.

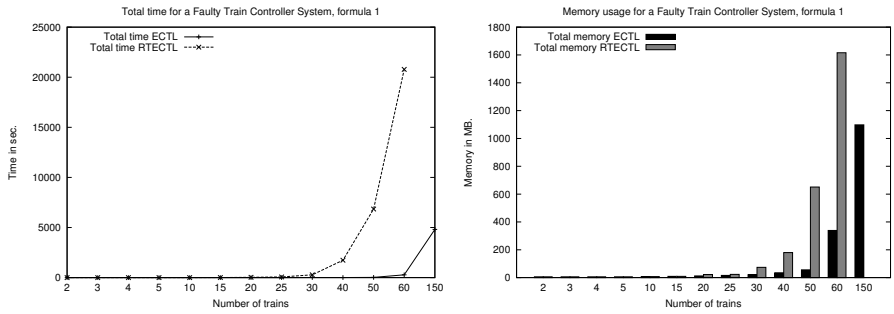
The equivalent ECTL formulae are:

- $tr(\varphi_1) = \mathbf{EF}(InTunnel_1 \wedge \mathbf{EX}((\neg InTunnel_1 \wedge \neg InTunnel_2) \wedge \mathbf{EX}((\neg InTunnel_1 \wedge \neg InTunnel_2) \wedge \mathbf{EX}(\neg InTunnel_1 \wedge \neg InTunnel_2)))$, for $n = 2$.
- $tr(\varphi_2) = \mathbf{EF}(InTunnel_1 \vee \mathbf{EX}((\neg InTunnel_1 \vee \neg InTunnel_2) \wedge \mathbf{EX}((\neg InTunnel_1 \vee \neg InTunnel_2) \wedge \mathbf{EX}(\neg InTunnel_1 \vee \neg InTunnel_2)))$, for $n = 2$.

The formula φ_1 states that there exists the case that Train 1 is in the tunnel and either it and other train will not be in the tunnel during the next $n + 1$ time units. The formula φ_2 expresses that there exists the case that Train 1 is in the tunnel or either it or other train will not be in the tunnel during the next $n + 1$ time units.

In all of the following tables, the amount of time used by BMC and SAT is given in the penultimate column, and maximum of memory usage of BMC and SAT is given in the

last column. In Tables 1 and 2 we present experimental results for the formulae φ_1 and φ_2, respectively. In Figures 2(a) and 2(b) we present a comparison of total time usage and total memory usage for the formulae φ_1.



(a) (b)

FIGURE 2: A comparison of total time usage and total memory usage for the formulae φ_1.

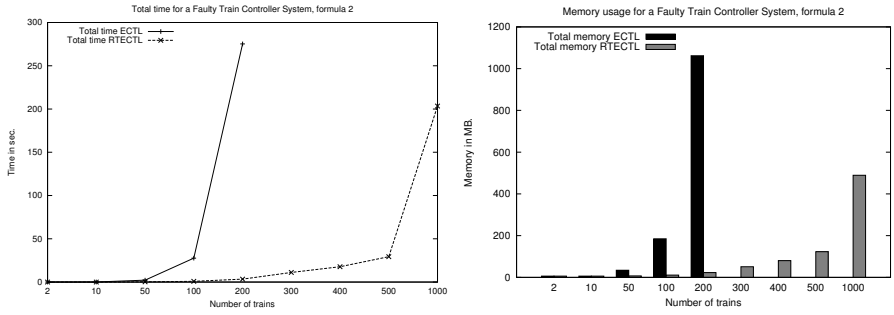
An analysis of experimental results for formula φ_1 leads to the conclusion that BMC for ECTL uses less time and memory comparing to BMC for RTECTL. The reason is that although BMC needs much more paths for verification in case of ECTL, but these paths are significantly shorter.

				BMC		**SAT**		**BMC + SAT**	
Logic	**n**	**k**	**LL**	**sec**	**MB**	**sec**	**MB**	**sec**	**MB**
RTECTL	2	3	2	0.01	1.70	0.00	6.00	0.01	6.00
ECTL	2	2	4	0.01	1.70	0.00	6.00	0.01	6.00
RTECTL	3	4	2	0.01	1.70	0.01	6.00	0.02	6.00
ECTL	3	2	5	0.00	1.70	0.00	6.00	0.00	6.00
RTECTL	4	5	2	0.02	1.83	0.00	6.00	0.03	6.00
ECTL	4	2	6	0.02	1.83	0.00	6.00	0.02	6.00
RTECTL	5	6	2	0.03	1.83	0.05	6.00	0.08	6.00
ECTL	5	2	7	0.02	1.96	0.01	6.00	0.03	6.00
RTECTL	10	11	2	0.28	2.47	0.52	7.00	0.80	7.00
ECTL	10	2	12	0.05	2.60	0.03	7.00	0.08	7.00
RTECTL	15	16	2	0.94	3.63	3.00	9.00	3.94	9.00
ECTL	15	2	17	0.21	3.76	0.12	9.00	0.33	9.00
RTECTL	20	21	2	2.93	5.31	23.78	22.00	26.71	22.00
ECTL	20	2	22	0.35	5.70	0.14	12.00	0.49	12.00
RTECTL	25	26	2	6.06	7.76	56.78	23.00	62.84	23.00
ECTL	25	2	27	0.90	8.26	0.69	16.00	1.60	16.00
RTECTL	30	31	2	11.56	10.98	269.14	74.00	280.70	74.00
ECTL	30	2	32	1.22	11.62	2.70	21.00	3.92	21.00

RTECTL	40	41	2	33.81	20.26	1698.67	180.00	1732.48	180.00
ECTL	40	2	42	2.87	21.16	5.88	35.00	8.74	35.00
RTECTL	50	51	2	102.58	33.93	6738.59	651.00	6841.17	651.00
ECTL	50	2	52	5.11	35.22	14.94	56.00	20.05	56.00
RTECTL	60	61	2	217.43	52.62	20563.46	1616.00	20780.89	1616.00
ECTL	100	2	102	29.58	199.40	247.63	339.00	277.21	339.00
ECTL	150	2	152	100.04	181.10	4709.48	1098.00	4809.52	1098.00

Table 1: Experimental results for formula φ_1 - RTECTL vs. ECTL. k is the bound, LL is the number of k-paths.

In Figures 3(a) and 3(b) we present a comparison of total time usage and total memory usage for the formulae φ_2.



(a) (b)

FIGURE 3: A comparison of total time usage and total memory usage for the formulae φ_2.

An observation of experimental results for formula φ_2 leads to the conclusion that BMC for RTECTL uses less time and memory comparing to BMC for ECTL. The reason is that BMC needs much more paths for verification in case of ECTL.

				BMC		**SAT**		**BMC + SAT**	
Logic	**n**	**k**	**LL**	**sec**	**MB**	**sec**	**MB**	**sec**	**MB**
RTECTL	2	2	2	0.00	1.70	0.00	6.00	0.00	6.00
ECTL	2	1	2	0.00	1.70	0.00	6.00	0.00	6.00
RTECTL	10	2	2	0.01	1.83	0.01	6.00	0.02	6.00
ECTL	10	1	10	0.05	2.21	0.02	6.00	0.07	6.00
RTECTL	50	2	2	0.16	3.12	0.06	7.00	0.23	7.00
ECTL	50	1	50	1.51	20.78	0.67	34.00	2.18	34.00

RTECTL	100	2	2	0.56	5.82	0.26	11.00	0.82	11.00
ECTL	100	1	100	11.20	109.00	16.42	185.00	27.61	185.00
RTECTL	200	2	2	2.36	15.11	0.99	23.00	3.36	23.00
ECTL	200	1	200	81.60	213.50	193.61	1062.00	275.21	1062.00
RTECTL	300	2	2	6.10	29.54	5.00	51.00	11.10	51.00
RTECTL	400	2	2	9.73	49.27	7.97	80.00	17.70	80.00
RTECTL	500	2	2	12.73	74.15	16.63	123.00	29.37	123.00
RTECTL	1000	2	2	54.94	276.40	148.54	489.00	203.47	489.00

Table 2: Experimental results for φ_2 - RTECTL vs. ECTL. k is the bound, LL is the number of k-paths.

5.1.1 Generic Pipeline Paradigm.

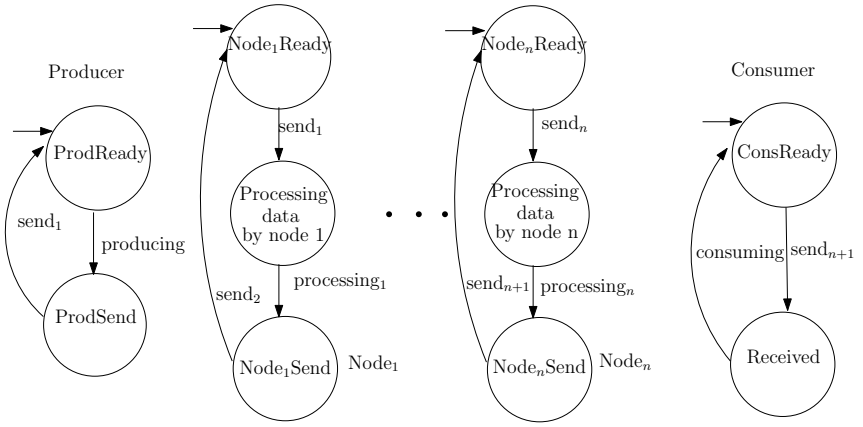


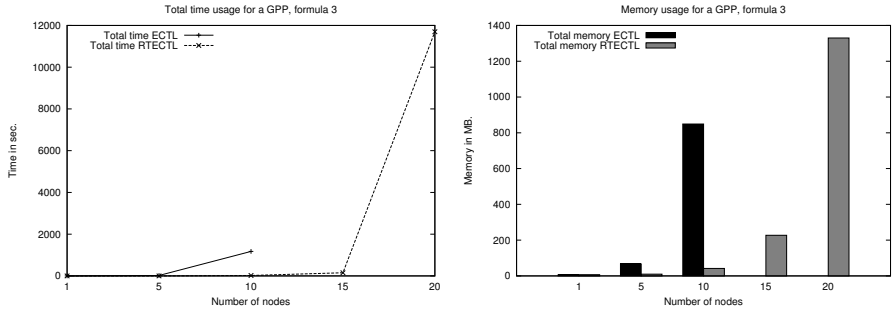
FIGURE 4: The GPP system [11]

The benchmark we consider is a generic pipeline paradigm (GPP) [18], 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 component (Producer, Consumer, and intermediate Nodes), and their protocols are shown on Fig. 4. The comparison of both BMC algorithms for RTECTL and ECTL with respect to the GPP system has been done by means of the following RTECTL specification:
$\varphi_3 = \mathbf{EG}_{[0,\infty)}(\neg ProdSend \vee \mathbf{EF}_{[2n+1,2n+2)}(Received))$, where n is number of nodes.
The equivalent ECTL formula for $n = 2$ is:
$tr(\varphi_3) = \mathbf{EG}(\neg ProdSend \vee \mathbf{EX}(\mathbf{EX}(\mathbf{EX}(\mathbf{EX}(\mathbf{EX}(Received))))))$.
$\varphi_4 = \mathbf{EG}_{[0,n^2+2n+1)}(\neg Received))$, where n is number of nodes.

The equivalent ECTL formula for $n = 2$ is:
$tr(\varphi_4) = \mathbf{EX}(\neg Received \wedge \mathbf{EX}(\neg Received \wedge \mathbf{EX}(\neg Received \wedge \mathbf{EX}(\neg Received \wedge \mathbf{EX}(\neg Received \wedge \mathbf{EX}(\neg Received \wedge \mathbf{EX}(\neg Received \wedge \mathbf{EX}(\neg Received \wedge \neg Received)))))))))$

In Tables 3 and 4 we present experimental results for the formula φ_3 and φ_4, respectively. Also in the case of this benchmark we can observe that time and memory usage depends on number of paths.

In Figures 5(a) and 5(b) we present a comparison of total time usage and total memory usage for the formulae φ_3.



(a) (b)

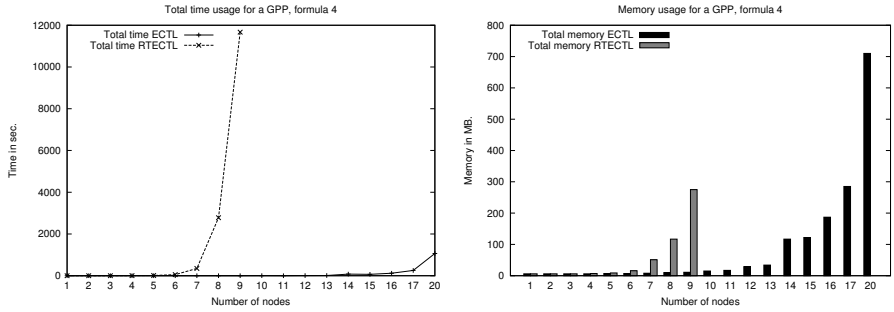
FIGURE 5: A comparison of total time usage and total memory usage for the formulae φ_3.

An observation of experimental results for formula φ_3 leads to the conclusion that BMC for RTECTL uses less time and memory comparing to BMC for ECTL. The reason is that although BMC needs much more paths for verification in case of ECTL.

				BMC		**SAT**		**BMC + SAT**	
Logic	**n**	**k**	**LL**	**sec**	**MB**	**sec**	**MB**	**sec**	**MB**
RTECTL	1	5	7	0.01	1.83	0.00	6.00	0.01	6.00
ECTL	1	5	19	0.04	2.09	0.01	6.00	0.05	6.00
RTECTL	5	13	15	0.89	4.92	0.56	10.00	1.44	10.00
ECTL	5	13	155	10.34	35.73	9.36	67.00	19.70	67.00
RTECTL	10	23	25	10.95	23.74	9.70	42.00	20.65	42.00
ECTL	10	23	505	258.28	445.00	916.79	849.00	1175.07	849.00
RTECTL	15	33	35	55.66	78.27	96.52	227.00	152.18	227.00
RTECTL	20	43	45	234.62	196.00	11461.90	1330.00	11696.52	1330.00

Table 3: Experimental results for formula φ_3 - RTECTL vs. ECTL. k is the bound, LL is the number of k-paths.

In Figures 6(a) and 6(b) we present a comparison of total time usage and total memory usage for the formulae φ_4.



(a) (b)

FIGURE 6: A comparison of total time usage and total memory usage for the formulae φ_4.

An analysis of experimental results for formula φ_4 leads to the conclusion that BMC for ECTL uses less time and memory comparing to BMC for RTECTL. The reason is that although BMC needs much more paths for verification in case of ECTL, but these paths are significantly shorter.

				BMC		**SAT**		**BMC + SAT**	
Logic	**n**	**k**	**LL**	**sec**	**MB**	**sec**	**MB**	**sec**	**MB**
RTECTL	1	4	1	0.00	1.70	0.00	6.00	0.00	6.00
ECTL	1	1	4	0.00	1.70	0.00	6.00	0.00	6.00
RTECTL	2	9	1	0.01	1.70	0.01	6.00	0.02	6.00
ECTL	2	1	9	0.00	1.70	0.00	6.00	0.00	6.00
RTECTL	3	16	1	0.02	1.83	0.28	6.00	0.29	6.00
ECTL	3	1	16	0.00	1.83	0.01	6.00	0.01	6.00
RTECTL	4	25	1	0.29	2.09	1.90	7.00	2.18	7.00
ECTL	4	1	25	0.02	2.09	0.03	6.00	0.05	6.00
RTECTL	5	36	1	0.58	2.34	10.63	9.00	11.20	9.00
ECTL	5	1	36	0.03	2.47	0.03	7.00	0.06	7.00
RTECTL	6	49	1	1.63	2.86	49.99	16.00	51.62	16.00
ECTL	6	1	49	0.07	2.99	0.08	7.00	0.15	7.00
RTECTL	7	64	1	2.97	3.50	346.04	51.00	349.01	51.00
ECTL	7	1	64	0.09	3.76	0.12	8.00	0.21	8.00
RTECTL	8	81	1	5.52	4.41	2775.31	117.00	2780.83	117.00
ECTL	8	1	81	0.15	4.66	0.79	10.00	0.94	10.00
RTECTL	9	100	1	11.55	5.44	11659.16	275.00	11670.71	275.00
ECTL	9	1	100	0.25	5.95	0.64	11.00	0.88	11.00

ECTL	10	1	121	0.32	7.62	1.92	15.00	2.24	15.00
ECTL	11	1	144	0.44	9.56	0.99	17.00	1.43	17.00
ECTL	12	1	169	0.52	12.01	6.97	29.00	7.49	29.00
ECTL	13	1	196	0.66	14.97	9.57	34.00	10.23	34.00
ECTL	14	1	225	0.90	18.46	75.31	117.00	76.21	117.00
ECTL	15	1	256	1.22	22.59	65.80	122.00	67.02	122.00
ECTL	16	1	289	1.46	27.50	116.33	187.00	117.80	187.00
ECTL	17	1	324	1.72	33.06	250.14	285.00	251.86	285.00
ECTL	20	1	441	3.07	55.17	1062.39	710.00	1065.46	710.00

Table 4: Experimental results for formula φ_4 - RTECTL vs. ECTL. k is the bound, LL is the number of k-paths.

6 Conclusions

For the tests we have used a computer equipped with AMD Phenom™ 9550 Quad-Core 2200 MHz processor and 8 GB of RAM, running Ubuntu Linux with kernel version 3.5.0-17-generic. We have used the state of the art SAT-solver MiniSat 2 [19, 20], which is one of the best SAT-solver. MiniSat 2 also has been used in the experimental results in many papers concerning the BMC method, for example [11, 16, 12], and many others.

In this paper we have presented a comparison between the BMC method for ECTL and the BMC method for RTECTL. Moreover, we have presented a correct translation for operator $\mathbf{EG}_I$, which was not defined in [10, 14]. Further, we have tested and compared with each other on to standard benchmarks the BMC translations for RTECTL and ECTL incremented in [11, 12].

Acknowledgements. The author wishes to thank Bożena Woźna-Szcześniak and Andrzej Zbrzezny for valuable comments and numerous suggestions which helped to improve this paper.

References

[1] Clarke, E., Grumberg, O., Peled, D.: Model Checking. MIT Press (1999)

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

[3] Bryant, R.: Graph-based algorithms for boolean function manipulation. IEEE Transaction on Computers **35(8)** (1986) 677–691

[4] Lomuscio, A., Penczek, W., Qu, H.: Partial order reduction for model checking interleaved multi-agent systems. In: Proceedings of the 9th International Conference on Autonomous Agents and Multi-Agent systems (AAMAS'2010)., IFAAMAS Press (2010) 659–666

[5] Clarke, E., Emerson, E.A., Sistla, A.P.: Automatic verification of finite state concurrent systems using temporal logic specifications: A practical approach. ACM Transactions on Programming Languages and Systems **8(2)** (1986) 244–263

[6] Biere, A., Cimatti, A., Clarke, E., Zhu, Y.: Symbolic model checking without BDDs. In: Proceedings of the 5th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'99). Volume 1579 of LNCS., Springer-Verlag (1999) 193–207

[7] Penczek, W., Woźna, B., Zbrzezny, A.: Bounded model checking for the universal fragment of ctl. Fundamenta Informaticae **51**(1-2) (2002) 135–156

[8] Woźna, B., Zbrzezny, A.: Checking ACTL* properties of discrete timed automata via bounded model checking. In: Proceedings of the 1st Int. Workshop on Formal Analysis and Modeling of Timed Systems (FORMATS'03). Volume 2791 of LNCS., Springer-Verlag (2004) 18–33

[9] Penczek, W., Lomuscio, A.: Verifying epistemic properties of multi-agent systems via bounded model checking. In: Proceedings of the 2nd Int. Conf. on Autonomous Agents and Multi-Agent Systems (AAMAS'03), ACM (2003) 209–216

[10] Emerson, E.A., Mok, A.K., Sistla, A.P., Srinivasan, J.: Quantitative temporal reasoning. Real-Time Systems **4**(4) (1992) 331–352

[11] Woźna-Szcześniak, B., Zbrzezny, A.M., Zbrzezny, A.: The BMC method for the existential part of RTCTLK and interleaved interpreted systems. In: Proceedings of the 15th Portuguese Conference on Artificial Intelligence (EPIA'2011). Volume 7026 of LNAI., Springer-Verlag (2011) 551–565

[12] Zbrzezny, A.: Improving the translation from ECTL to SAT. Fundamenta Informaticae **85(1-4)** (2008) 513–531

[13] Baier, C., Katoen, J.P.: Principles of model checking. MIT Press (2008)

[14] Campos, S.V.A.: A quantitative approach to the formal verification of real-time systems. PhD thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA (1996)

[15] Woźna-Szcześniak, B.: Bounded model checking for the existential part of real-time CTL and knowledge. In: Advances in Software Engineering Techniques - 4th IFIP TC 2 Central and East European Conference on Software Engineering Techniques, CEE-SET'2009. Revised Selected Papers. Volume 7054 of LNCS., Springer (2009) 164–178

[16] Woźna-Szcześniak, B., Zbrzezny, A.M., Zbrzezny, A.: Verifying RTECTL Properties of a Train Controller System. Scientific Issues, Jan Długosz University in Częstochowa, Mathematics **XV** (2011) 153–162

[17] Hoek, W., Wooldridge, M.: Cooperation, knowledge, and time: Alternating-time temporal epistemic logic and its applications. Studia Logica **75**(1) (2003) 125–157

[18] Peled, D.: All from one, one for all: On model checking using representatives. In: Proceedings of the 5th Int. Conf. on Computer Aided Verification (CAV'93). Volume 697 of LNCS., Springer-Verlag (1993) 409–423

[19] Eén, N., Sörensson, N. MiniSat http://minisat.se/MiniSat.html.

[20] Eén, N., Sörensson, N.: MiniSat - A SAT Solver with Conflict-Clause Minimization. In: Proceedings of 8th International Conference on Theory and Applications of Satisfiability Testing(SAT'05). LNCS, Springer-Verlag (2005)