Jan Długosz University in Częstochowa
Scientific Issues, Mathematics XVI, Częstochowa 2011

VERIFYING RTECTL PROPERTIES OF A TRAIN CONTROLLER SYSTEMS

Bożena Woźna-Szcześniak, Agnieszka Zbrzezny, Andrzej Zbrzezny

Institute of Mathematics and Computer Science
Jan Długosz University in Częstochowa
al. Armii Krajowej 13/15, 42-200 Częstochowa, Poland
e-mail: {b.wozna, agnieszka.zbrzezny, a.zbrzezny}@ajd.czest.pl

Abstract. In the paper we deal with a classic concurrency problem – a *faulty train controller system* (FTC). In particular, we formalise it by means of finite automata, and consider several properties of the problem, which can be expressed as formulae of a soft real-time branching time temporal logic, called RTECTL. Further, we verify the RTECTL properties of FTC by means of SAT-based bounded model checking (BMC) method, and present the performance evaluation of the BMC method with respect to the considered problem. The performance evaluation is given by means of the running time and the memory used.

1. Introduction

Concurrency is a property of systems that allows to perform multiple computations in parallel and it is ubiquitous in computer science today, for example, it is the core feature of today operating systems. Concurrency is widespread but error prone - typical error includes race conditions and mutual exclusion violations; errors that are unknown in sequential computations. Traditional reliability measures such as simulation and testing fail in the presence of concurrency, due to the difficulties of reproducing erroneous behaviour.

Model checking [3] is an automated technique designed to establish in a formal and precise way that specific properties are satisfied by a given system. Its main idea consists in representing a (finite) state system as a Kripke structure M, expressing a specification by a logical formula φ, and checking automatically whether the formula φ holds in the model M. Unfortunately, the practical applicability of model checking is strongly restricted by the state-space explosion problem, which is mainly caused by representing concurrency of operations by their interleaving. Therefore, there are many different reduction techniques aimed at minimising models. The major methods include application of partial order reductions [10], symmetry reductions [6], abstraction techniques [4], OBDD-based symbolic storage methods [1], and SAT-based bounded [2, 9] and unbounded [8] model checking.

The RTCTL language [5] is a propositional branching-time temporal logic with bounded operators, which was introduced to allow specification and reasoning about time-critical correctness properties. It makes possible to directly express bounded properties like, for example, "property φ will occur in less than 10 unit time", or "property φ will always be asserted between 2 and 8 unit time". Note that properties like above can be expressed using nested applications of the next state operators, however the resulting CTL formula can be very complex and cumbersome to work with. RTCTL, by allowing bounds on all temporal operators to be specified, provides a much more compact and convenient way of expressing time-bounded properties.

In the paper we investigate a finite state systems modelled via a network of finite automata. In particular, we deal with a faulty train controller system (adapted from [7]) – a classic concurrency problem. We model it as a network of finite automata, and verify using a SAT-based bounded model checking (BMC) method for RTCTL properties.

The rest of the article is structured as follows. In the next section we provide the main formalisms used throughout the paper, i.e., finite automata, the RTCTL language together with its universal and existential subsets, and SAT-based BMC for the existential part of RTCTL (RTECTL). In section 3 we show how our SAT-based BMC for RTECTL works by means of the faulty train controller system. In section 4 we conclude our paper.

2. Preliminaries

2.1. Finite automata and parallel composition

Given is a set PV of propositional variables, each of which represents fundamental properties of the system in question. A *finite automaton*, we consider in the paper, is a mathematical structure $\mathcal{A} = (\Sigma, S, s^0, T, V)$ that consists of a finite set of actions (Σ), a finite set of states (S), an initial state (s^0), a transition relation ($T \subseteq S \times S$) defining rules for going from one state to another depending upon the input action, and a valuation function ($V : S \rightarrow 2^{PV}$)

which assigns to every state a set of propositional variables that are assumed to be true at this state.

Typically 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 automata that run in parallel and communicate with each other via executing shared actions. There are several ways of defining a parallel composition of a few automata. 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.

The following definition formalises the above discussion. Let $\mathcal{A}_i = (\Sigma_i, S_i, s_i^0, T_i, V_i)$ be an automaton, for $i = 1, \ldots, m$. We take $\Sigma = \bigcup_{i=1}^{m} \Sigma_i$, and for $\sigma \in \Sigma$ we define a set $\Sigma(\sigma) = \{1 \leq i \leq m \mid \sigma \in \Sigma_i\}$ that gives the indices of the components that synchronise at σ. *A parallel composition of m automata $\mathcal{A}_i$* is the automaton $\mathcal{A} = (\Sigma, S, s^0, T, V)$, where $\Sigma = \bigcup_{i=1}^{m} \Sigma_i$, $S = \prod_{i=1}^{m} S_i$, $s^0 = (s_1^0, \ldots, s_m^0)$, $V((s_1, \ldots, s_m)) = \bigcup_{i=1}^{m} V_i(s_i)$, and a transition $((s_1, \ldots, s_m), \sigma, (s_1', \ldots, s_m')) \in T$ iff $(\forall j \in \Sigma(\sigma))\ (s_j, \sigma, s_j') \in T_j$ and $(\forall i \in \{1, \ldots, m\} \setminus \Sigma(\sigma))\ s_i' = s_i$.

2.2. The RTCTL language

Let $p \in PV$, and I be an interval in $\mathbb{N} = \{0, 1, \ldots\}$ of the form: $[a, b)$ and $[a, \infty)$, for $a, b \in \mathbb{N}$[1]. 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 RTCTL is defined by the following grammar:

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

U_I and G_I are the operators for bounded "until" and "globally", respectively. E and A are the existential and universal path quantifiers, respectively. The remaining bounded temporal operators are defined in the standard way: $\mathrm{O}(\alpha \mathrm{R}_I \beta) \overset{def}{=} \mathrm{O}(\beta \mathrm{U}_I(\alpha \wedge \beta)) \vee \mathrm{OG}_I \beta$, $\mathrm{OF}_I \alpha \overset{def}{=} \mathrm{O}(\mathbf{true} \mathrm{U}_I \alpha)$, where $\mathrm{O} \in \{\mathrm{E}, \mathrm{A}\}$.

RTACTL is the fragment of RTCTL such that the formulae are restricted to the positive Boolean combinations of $\mathrm{AX}\varphi$, $\mathrm{AG}\varphi$ and $\mathrm{A}(\varphi \mathrm{U} \psi)$. Negation can be applied to propositions only.

RTECTL is the fragment of RTCTL such that the formulae are restricted to the positive Boolean combinations of $\mathrm{EX}\varphi$, $\mathrm{EG}\varphi$ and $\mathrm{E}(\varphi \mathrm{U} \psi)$. Negation can be applied to propositions only.

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

A *model* for RTCTL is the automaton $\mathcal{A} = (\Sigma, S, s^0, T, V)$ as defined in the previous section. Note that this finite automaton can be viewed as a standard Kripke structure or labelled transition system.

Let $\mathcal{A} = (\Sigma, S, s^0, T, V)$ be a model. A *path* of $\mathcal{A}$ is an infinite sequence $\pi = (s_0, s_1, \dots)$ of states such that $(s_j, s_{j+1}) \in T$ for each $j \in \mathbb{N}$. For a path $\pi = (s_0, s_1, \dots)$, we take $\pi(j) = s_j$. By $\Pi(s)$ we denote the set of all the paths starting at $s \in S$, $\mathrm{O} \in \{\mathrm{E}, \mathrm{A}\}$, and $\# = \exists\pi \in \Pi(s)$ if $\mathrm{O} = \mathrm{E}$, otherwise $\# = \forall\pi \in \Pi(s)$. Given the above, the formal semantics of RTCTL is defined recursively as follows:

- $\mathcal{A}, s \models \mathbf{true}$, • $\mathcal{A}, s \not\models \mathbf{false}$,
- $\mathcal{A}, s \models p$ iff $p \in V(s)$, • $\mathcal{A}, s \models \neg p$ iff $p \notin V(s)$,
- $\mathcal{A}, s \models \alpha \wedge \beta$ iff $\mathcal{A}, s \models \alpha$ and $\mathcal{A}, s \models \beta$,
- $\mathcal{A}, s \models \alpha \vee \beta$ iff $\mathcal{A}, s \models \alpha$ or $\mathcal{A}, s \models \beta$,
- $\mathcal{A}, s \models \mathrm{OX}\alpha$ iff$\#(\mathcal{A}, \pi(1) \models \alpha)$,
- $\mathcal{A}, s \models \mathrm{O}(\alpha \mathrm{U}_I \beta)$ iff $\#(\exists m \in I)[\mathcal{A}, \pi(m) \models \beta$ and $(\forall j < m)\mathcal{A}, \pi(j) \models \alpha]$,
- $\mathcal{A}, s \models \mathrm{OG}_I\alpha$ iff $\#(\forall m \in I)[\mathcal{A}, \pi(m) \models \alpha]$.

We end the section by defining the notions of validity and the model checking problem. Namely, a RTCTL formula φ is *valid* in $\mathcal{A}$ (denoted $\mathcal{A} \models \varphi$) iff $\mathcal{A}, s^0 \models \varphi$, i.e., φ is true at the initial state of the model $\mathcal{A}$. The *model checking problem* asks whether $\mathcal{A} \models \varphi$.

2.3. SAT-based BMC for RTECTL

In this section we give an overview of a SAT-based BMC method for the existential fragment of RTCTL (RTECTL) [11]. As usual, we start by defining k-paths, and (k, l)-*loops*, and then in turn we define a *bounded semantics* for RTECTL, which is later used for translation to SAT.

Given are a model $\mathcal{A} = (\Sigma, S, s^0, T, V)$ and a bound $k \geq 0$. A k-path is the prefix of length k of a path in Π. By P_k we denote a set of all the k-paths. By $P_k(s)$ we denote a set of all the k-paths π_k with $\pi_k(0) = s$. A (k, l)-*loop* is a k-path $\pi_k = (\pi_k(0), \dots, \pi_k(l), \dots, \pi_k(k))$ such that $\pi_k(l) = \pi_k(k)$, for some $0 \leq l < k$. A function *loop* : $\Pi_k \to 2^{\mathbb{N}}$ identifies these k-paths that are loops and it is defined as: $\mathit{loop}(\pi_k) = \{l \mid 0 \leq l < k \text{ and } \pi_k(l) = \pi_k(k)\}$.

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

- $\mathcal{A}, s \models_k \mathbf{true}$, • $\mathcal{A}, s \not\models_k \mathbf{false}$,
- $\mathcal{A}, s \models_k p$ iff $p \in V(s)$, • $\mathcal{A}, s \models_k \neg p$ iff $p \notin V(s)$,
- $\mathcal{A}, s \models_k \alpha \vee \beta$ iff $\mathcal{A}, s \models_k \alpha$ *or* $\mathcal{A}, s \models_k \beta$,

- $\mathcal{A}, s \models_k \alpha \wedge \beta$ iff $\mathcal{A}, s \models_k \alpha$ *and* $\mathcal{A}, s \models_k \beta$,
- $\mathcal{A}, s \models_k \mathrm{EX}\alpha$ iff $k > 0$ and $(\exists \pi \in \Pi_k(s))\mathcal{A}, \pi(1) \models_k \alpha$,
- $\mathcal{A}, s \models_k \mathrm{E}(\alpha \mathrm{U}_I \beta)$ iff $(\exists \pi \in \Pi_k(s))(\exists 0 \leq m \leq k)(m \in I$ and $\mathcal{A}, \pi(m) \models_k \beta$ and $(\forall 0 \leq j < m)\mathcal{A}, \pi(j) \models_k \alpha)$,
- $\mathcal{A}, s \models_k \mathrm{EG}_I\alpha$ iff $(\exists \pi \in \Pi_k(s))((k \geq \mathit{right}(I)$ and $(\forall j \in I)\ \mathcal{A}, \pi(j) \models_k \alpha)$ or $(k < \mathit{right}(I)$ and $(\exists l \in \mathit{loop}(\pi))(\forall \mathit{min}(\mathit{left}(I), l) \leq j < k)\mathcal{A}, \pi(j) \models_k \alpha))$.

A RTECTL formula φ is *valid in model $\mathcal{A}$ with bound k* (denoted $\mathcal{A} \models_k \varphi$) iff $\mathcal{A}, s^0 \models_k \varphi$, i.e., φ is $k-$true at the initial state of the model $\mathcal{A}$. The *bounded model checking problem* asks whether $\mathcal{A} \models_k \varphi$.

The following theorem, which can be proven by induction on the length of a RTECTL formula, states that there exists a bound such that bounded and unbounded semantics are equivalent. This implies that the model checking problem ($\mathcal{A} \models \varphi$) can be reduced to the bounded model checking problem ($\mathcal{A} \models_k \varphi$).

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

We can also show even the stronger property, namely, we can prove that φ is $k-$true in $\mathcal{A}$ if and only if φ is $k-$true in $\mathcal{A}$ with a number of $k-$paths reduced to $f_k(\varphi)$, where the function $f_k : \mathit{RTECTL} \to \mathbb{N}$ is defined as follows:

- $f_k(\mathbf{true}) = f_k(\mathbf{false}) = f_k(p) = f_k(\neg p) = 0$, where $p \in PV$,
- $f_k(\alpha \wedge \beta) = f_k(\alpha) + f_k(\beta)$,
- $f_k(\alpha \vee \beta) = max\{f_k(\alpha), f_k(\beta)\}$,
- $f_k(\mathrm{X}\alpha) = f_k(\alpha) + 1$,
- $f_k(\mathrm{E}(\alpha \mathrm{U}_I \beta)) = k \cdot f_k(\alpha) + f_k(\beta) + 1$,
- $f_k(\mathrm{EG}_I\alpha) = (k + 1) \cdot f_k(\alpha) + 1$.

Given are a model $\mathcal{A} = (\Sigma, S, s^0, T, V)$, a bound $k \geq 0$, and a RTECTL formula φ. The problem of checking whether $\mathcal{A} \models_k \varphi$ holds can be translated to the satisfiability problem of the following propositional formula:

$$[\mathcal{A}, \varphi]_k := [\mathcal{A}^{\varphi, s^0}]_k \wedge [\varphi]_{\mathcal{A},k} \tag{1}$$

where, the formula $[\mathcal{A}^{\varphi, s^0}]_k$ constrains the $f_k(\varphi)$ symbolic k-paths to be valid k-paths of $\mathcal{A}$, while the formula $[\varphi]_{\mathcal{A},k}$ 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 RTECTL formula can be done by means of a SAT-solver.

In order to define the formula $[\mathcal{A},\varphi]_k$ we proceed as follows. We assume that each state s of $\mathcal{A}$ is encoded by a bit-vector whose length, say r, depends on the number of local states of "component" automata. Thus, each state s of $\mathcal{A}$ we can represent by a vector $w = (u_1, \dots, u_r)$ of propositional variables (usually called *state variables*), to which we refer to as *symbolic states*. A finite sequence $(w_0, \dots, w_k)$ 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 $(w_{0,j}, \dots, w_{k,j})$, where $w_{i,j}$ are symbolic states for $0 \le j < f_k(\varphi)$ and $0 \le i \le k$. 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.

The propositional formula $[\mathcal{A}^{\varphi,s^0}]_k$ is defined over symbolic states $w_{i,j}$, for $0 \le i \le k$ and $0 \le j < f_k(\varphi)$, in the following way:

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

where $I_{s^0}(w)$ is a formula that encodes the initial state s^0 of $\mathcal{A}$, and $\mathcal{R}(w,w')$ is a formula that encodes the transition relation of $\mathcal{A}$.

The next step is the translation of a RTECTL formula φ into a propositional formula $[\varphi]_{\mathcal{A},k} := [\varphi]_k^{[0,0]}$, where $k \ge 0$ is a bound, $[\varphi]_k^{[m,n]}$ denotes the translation of φ at the symbolic state $w_{m,n}$, and it is defined inductively as follows:

- $[\mathbf{true}]_k^{[m,n]} := \mathbf{true}$,
- $[\mathbf{false}]_k^{[m,n]} := \mathbf{false}$,
- $[p]_k^{[m,n]} := p(w_{m,n})$,
- $[\neg p]_k^{[m,n]} := \neg p(w_{m,n})$,
- $[\alpha \wedge \beta]_k^{[m,n]} := [\alpha]_k^{[m,n]} \wedge [\beta]_k^{[m,n]}$,
- $[\alpha \vee \beta]_k^{[m,n]} := [\alpha]_k^{[m,n]} \vee [\beta]_k^{[m,n]}$,
- $[\mathrm{EX}\alpha]_k^{[m,n]} := \begin{array}{ll} (1) \displaystyle\bigvee_{ll=0}^{f_k(\varphi)-1} H(w_{m,n}, w_{0,ll}) \wedge [\alpha]_k^{[1,ll]}, & \text{if } k > 0 \\ (2) \ \mathbf{false}, & \textit{otherwise} \end{array}$
- $[\mathrm{E}(\alpha \mathrm{U}_I \beta)]_k^{[m,n]} := \displaystyle\bigvee_{ll=0}^{f_k(\varphi)-1} \Big(H(w_{m,n}, w_{0,ll}) \wedge \bigvee_{i=0}^{k} \big([\beta]_k^{[i,ll]} \wedge In(i,I) \wedge \bigwedge_{j=0}^{i-1} [\alpha]_k^{[j,ll]}\big)\Big)$,
- $[\mathrm{EG}_I\alpha]_k^{[m,n]} := \displaystyle\bigvee_{ll=0}^{f_k(\varphi)-1} H(w_{m,n}, w_{0,ll}) \wedge$

  (1) $\displaystyle\bigwedge_{j=left(I)}^{right(I)} [\alpha]_k^{[j,ll]}$, if $right(I) \le k$

  (2) $\displaystyle\bigvee_{l=0}^{k-1} \Big(H(w_{k,ll}, w_{l,ll}) \wedge \bigwedge_{j=min(left(I),l)}^{k-1} [\alpha]_k^{[j,ll]}\Big)$, otherwise.

The following theorem, which can be proven by induction on the length of a RTECTL formula, expresses the correctness and the completeness of the translation presented above.

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

3. A faulty train controller system

To evaluate the BMC technique for RTECTL, we analyse a scalable concurrent system, which is a faulty train controller system (FTC) (adapted from [7]). 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. 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 colour light signals 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 colour light signals, however it can be faulty, and thereby it does not serve its purpose. Namely, the controller does not ensure the mutual exclusion property: two trains never occupy the tunnel at the same time.

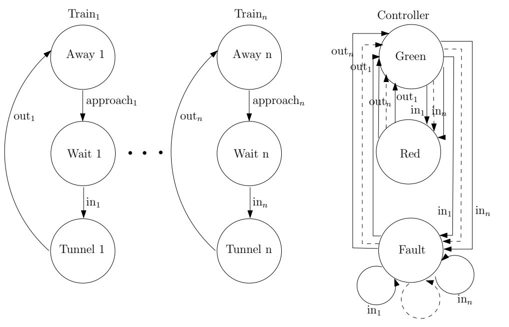

Figure 1: A network of automata for train controller system

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

$$\begin{aligned}
\varphi_1 &= \mathrm{AG}_{[0,\infty]}\big(InTunnel_1 \rightarrow \mathrm{AF}_{[1,\infty]}(InTunnel_1)\big),\\
\varphi_2 &= \mathrm{AG}_{[0,\infty]}\big(\textstyle\bigwedge_{i=1}^{n-1}\bigwedge_{j=i+1}^{n}\neg(InTunnel_i \wedge InTunnel_j)\big),\\
\varphi_3 &= \mathrm{AG}_{[0,\infty]}\big(InTunnel_1 \rightarrow \mathrm{AF}_{[1,n+1]}\big(\textstyle\bigvee_{i=1}^{n} InTunnel_i\big)\big).
\end{aligned}$$

The formula φ_1 states that it is always the case that whenever Train 1 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 the fact that trains have exclusive access to the tunnel. The formula φ_3 expresses that it is always the case that if Train 1 is in the tunnel, then either he or other train will be in the tunnel during the next $n+1$ time units.

All the above formulae are not true in the model for FTC, and for every specification given, there exists a counterexample. This was shown by means of the BMC method for RTECTL and testing the formulae $\psi_i = \neg\varphi_i$ (for $i = 1, 2, 3$), which are the negations of the assumed universal specifications and are interpreted existentially.

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 3600 seconds, and memory limit to 3072 MB. We have used the state of the art SAT-solver MiniSat 2. The experimental results are shown in Table 1. In particular, we present there the results for the formulae φ_1, φ_2, and φ_3, and the maximum number of trains we were able to model check by means of our BMC method for RTECTL.

To get the experimental results in Table 1, we started with a propositional encoding of a network of automata that models FTC. To this end we have encoded the states of the network, in particular the initial state, and the transition relation. An example of such encoding for two trains and a controller, we present below.

Let $SV = \{p_1, p_2, \dots\}$ be an infinite set of state variables. A Boolean encoding of all the local states of the two automata representing trains is the following:

$Train\ 1$				$Train\ 2$			
$state$	bit_2	bit_1	$formula$	$state$	bit_4	bit_3	$formula$
$away_1$	0	0	$\neg p_1 \wedge \neg p_2$	$away_2$	0	0	$\neg p_3 \wedge \neg p_4$
$wait_1$	1	0	$\neg p_1 \wedge p_2$	$wait_2$	1	0	$\neg p_3 \wedge p_4$
$tunnel_1$	0	1	$p_1 \wedge \neg p_2$	$tunnel_2$	0	1	$p_3 \wedge \neg p_4$

$Controller$			
$location$	bit_6	bit_5	$formula$
$green$	0	0	$\neg p_5 \wedge \neg p_6$
red	0	1	$p_5 \wedge \neg p_6$
$faulty$	1	0	$\neg p_5 \wedge p_6$

Given the above, it is easy to see that each state of the network of automata modelling the FTC system can be represented by a valuation of a symbolic state $w = (p_1, \dots, p_6)$. Then, a propositional formula $I_{s^0}(w)$, which encodes the initial global state of the considered system, is the conjunction of three formulae that encode all the local initial states, i.e.

$$I_{s^0}(w) = (\neg p_1 \wedge \neg p_2) \wedge (\neg p_3 \wedge \neg p_4) \wedge (\neg p_5 \wedge \neg p_6)$$

Furthermore, let $w = (p_1, \dots, p_6), w' = (p'_1, \dots, p'_6)$ be two different symbolic states. A propositional formula $\mathcal{R}(w, w')$, which encodes all the transitions of the considered system is defined as the disjunction of formula that encode single transitions:

$\mathcal{R}(w, w')$	$approach_1 \vee in_1 \vee out_1 \vee approach_2 \vee in_2 \vee out_2$
$approach_1$	$\neg p_1 \wedge \neg p_2 \wedge \neg p'_1 \wedge p'_2 \wedge (p_3 \leftrightarrow p'_3) \wedge (p_4 \leftrightarrow p'_4) \wedge (p_5 \leftrightarrow p'_5)$ $\wedge (p_6 \leftrightarrow p'_6)$
in_1	$\neg p_1 \wedge p_2 \wedge p'_1 \wedge \neg p'_2 \wedge (p_3 \leftrightarrow p'_3) \wedge (p_4 \leftrightarrow p'_4) \wedge (\neg p_5 \wedge \neg p_6 \wedge$ $p'_5 \wedge \neg p'_6 \vee \neg p_5 \wedge \neg p_6 \wedge \neg p'_5 \wedge p'_6 \vee \neg p_5 \wedge p_6 \wedge \neg p'_5 \wedge p'_6)$
out_1	$p_1 \wedge \neg p_2 \wedge \neg p'_1 \wedge \neg p'_2 \wedge (p_3 \leftrightarrow p'_3) \wedge (p_4 \leftrightarrow p'_4) \wedge$ $(p_5 \wedge \neg p_6 \wedge \neg p'_5 \wedge \neg p'_6 \vee \neg p_5 \wedge p_6 \wedge \neg p'_5 \wedge \neg p'_6)$
$approach_2$	$\neg p_3 \wedge \neg p_4 \wedge \neg p'_3 \wedge p'_4 \wedge (p_1 \leftrightarrow p'_1) \wedge (p_2 \leftrightarrow p'_2) \wedge (p_5 \leftrightarrow p'_5)$ $\wedge (p_6 \leftrightarrow p'_6)$
in_2	$\neg p_3 \wedge p_4 \wedge p'_3 \wedge \neg p'_4 \wedge (p_1 \leftrightarrow p'_1) \wedge (p_2 \leftrightarrow p'_2) \wedge (\neg p_5 \wedge \neg p_6 \wedge$ $p'_5 \wedge \neg p'_6 \vee \neg p_5 \wedge \neg p_6 \wedge \neg p'_5 \wedge p'_6 \vee \neg p_5 \wedge p_6 \wedge \neg p'_5 \wedge p'_6)$
out_2	$p_3 \wedge \neg p_4 \wedge \neg p'_3 \wedge \neg p'_4 \wedge (p_1 \leftrightarrow p'_1) \wedge (p_2 \leftrightarrow p'_2) \wedge$ $(p_5 \wedge \neg p_6 \wedge \neg p'_5 \wedge \neg p'_6 \vee \neg p_5 \wedge p_6 \wedge \neg p'_5 \wedge \neg p'_6)$

			number of			BMC		MiniSat 2	
φ	k	$f_k(\varphi)$	trains	variables	clauses	sec	MB	sec	MB
φ_1	4	2	1000	4251246	12747733	217.9	553.5	2081.0	902.0
φ_2	16	1	8	4349	12418	0.1	2.0	1882.3	32.0
φ_3	4	2	240	292798	876949	7.7	39.6	1851.0	676.0

Table 1: Experimental results

4. Conclusions

In this paper we gave a SAT-based symbolic approach to bounded model checking of concurrent systems modelled by network of finite automata. We focused on the properties expressed in RTECTL. The method has been implemented, and tested on the standard benchmark – a faulty train controller system. The benchmark has been carefully selected in such a way as to reveal the advantages and disadvantages of both approaches.

Aknowledgements. This study was partly supported by National Science Center under the grant No. UMO-2011/01/B/ST6/05317.

References

[1] R. Bryant. Binary Decision Diagrams and beyond: Enabling technologies for formal verification. In: *Proceedings of the International Conference on Computer-Aided Design (ICCAD'95)*, pp. 236–243, 1995.

[2] E. Clarke, A. Biere, R. Raimi, Y. Zhu. Bounded model checking using satisfiability solving. *Formal Methods in System Design*, **19**(1), 7–34, 2001.

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

[4] D. Dams, O. Grumberg, R. Gerth. Abstract interpretation of reactive systems: Abstractions preserving ACTL*, ECTL* and CTL*. In: *Proceedings of the IFIP Working Conference on Programming Concepts, Methods and Calculi*. Elsevier, 1994.

[5] E.A. Emerson, A.K Mok., A.P. Sistla, J. Srinivasan. Quantitative temporal reasoning. In: *Proceedings of the 2nd International Workshop on Computer Aided Verification*, pp. 136–145, Springer, London, 1991.

[6] E.A. Emerson and A.P. Sistla. Symmetry and model checking. *Formal Methods in System Design*, **9**, 105–131, 1995.

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

[8] K.L. McMillan. Applying SAT methods in unbounded symbolic model checking. In: *Proc. of the 14th Int. Conf. on Computer Aided Verification*, vol. 2404 of *LNCS*, pp. 250–264, Springer, London, 2002.

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

[10] P. Wolper, P. Godefroid. Partial order methods for temporal verification. In: *Proceedings of the 4th International Conference on Concurrency Theory*, vol. 715 of *LNCS*, pp. 233–246, Springer, London, 1993.

[11] B. Woźna-Szcześniak. Bounded model checking for the existential part of Real-Time CTL and knowledge. In *Pre-Proceedings of CEE-SET'09*, pp. 178–191, AGH Krakow, Poland, 2009.