Scientific Issues
Jan Długosz University
in Częstochowa
Mathematics XXI (2016)
189–206
DOI http://dx.doi.org/10.16926/m.2016.21.15

A COMPARISON OF SMT-SOLVERS FOR TIMED WEIGHTED INTERPRETED SYSTEMS

AGNIESZKA M. ZBRZEZNY

Abstract

We compare four SMT-solvers for the same SMT-based bounded model checking algorithm for multi-agent systems modelled by timed weighted interpreted systems and for properties expressed in the existential fragment of epistemic weighted linear-time temporal logic (WELTLK). To this end, we use the timed weighted generic pipeline paradigm (TWGPP) and the timed weighted train controller system (TWTCS). We consider several properties of the problems that can be expressed in WELTLK, and we present the performance evaluation of the mentioned bounded model checking method using four different SMT-solvers: Z3, Yices, CVC4 and Mathsat, by means of the running time and the memory used.

1. Introduction

Model checking [6] is an automatic verification technique for concurrent systems. To be able to check automatically whether the system satisfies a given property, one must first create a model of the system, and then describe in a formal language both the created model and the property. Bounded model checking (BMC) for multi-agent systems (MAS) is a symbolic model checking method. It uses a reduction of the problem of truth of, among others, an epistemic formula [8] in a model of MAS to the problem of satisfiability of formulae. The reduction is typically to SAT, but more recently to SMT.

Weighted formalisms are well known in the model checking research area [4, 9] to reason about resource requirements. Another quantitative dimension is the temporal duration of actions. In this area were considered: the

• *Agnieszka M. Zbrzezny* — Jan Długosz University in Częstochowa.
Partly supported by National Science Centre under the grant No. 2014/15/N/ST6/05079.

timed automata [1] the timed interpreted systems [10], and the weighted timed automata [4].

We choose the bounded model checking method for TWIS and WELTLK because it is one of most complicated SMT-BMC methods we developed. In our previous methods we used only Z3 SMT-solver, which is one of the best SMT-solvers, but we would like to check other SMT-solvers.

In this paper, we make the following contributions. Firstly, we recall the SMT-based BMC method for WELTLK and for TWISs. Secondly, we report on the initial experimental evaluation of our SMT-based BMC method and analyse the results.

2. Preliminaries

2.1. **SMT-solvers.** We chose only four SMT-solvers which support SMT-LIBv2 and logic QF_LIA (quantifier-free linear integer arithmetic) and were submitted to The 11th International Satisfiability Modulo Theories Competition `http://smtcomp.sourceforge.net/2016/`.

2.1.1. *Z3.* Z3 is a state-of-the art theorem prover from Microsoft Research. It can be used to check the satisfiability of logical formulas over one or more theories. Z3 offers a compelling match for software analysis and verification tools, since several common software constructs map directly into supported theories. Z3 is a low level tool. It is best used as a component in the context of other tools that require solving logical formulas. Consequently, Z3 exposes a number of API facilities to make it convenient for tools to map into Z3, but there are no stand-alone editors or user-centric facilities for interacting with Z3. The language syntax used in the front ends favor simplicity in contrast to linguistic convenience [3] (`https://github.com/Z3Prover/z3/wiki`).

2.1.2. *Yices 2.* The winner of SMT-COMP 2016 in the application track for QF_LIA. Yices 2 is an SMT solver that decides the satisfiability of formulas containing uninterpreted function symbols with equality, real and integer arithmetic, bitvectors, scalar types, and tuples. Yices 2 supports both linear and nonlinear arithmetic. Yices 2 can process input written in the SMT-LIB notation (both versions 2.0 and 1.2 are supported). Alternatively, Yices 2 have its own specification language, which includes tuples and scalar types [7] (`http://yices.csl.sri.com/`).

2.1.3. *MathSAT.* MathSAT 5 is the successor of MathSAT 4, supporting a wide range of theories (including e.g. equality and uninterpreted functions, linear arithmetic, bit-vectors, and arrays) and functionalities (including e.g. computation of Craig interpolants, extraction of unsatisfiable cores, generation of models and proofs, and the ability of working incrementally) [5] (`http://mathsat.fbk.eu/`). MathSAT 5 is a joint project of Fondazione Bruno Kessler and DISI-University of Trento.

2.1.4. *CVC4.* The winner of SMT-COMP 2016 in the main track for QF_LIA. CVC4 is an efficient open-source automatic theorem prover for satisfiability modulo theories (SMT) problems. It can be used to prove the validity (or, dually, the satisfiability) of first-order formulas in a large number of built-in logical theories and their combination. CVC4 is the fourth in the Cooperating Validity Checker family of tools (CVC, CVC Lite, CVC3) but does not directly incorporate code from any previous version. A joint project of NYU and U Iowa, CVC4 aims to support the features of CVC3 and SMT-LIBv2 while optimizing the design of the core system architecture and decision procedures to take advantage of recent engineering and algorithmic advances [2] (`http://cvc4.cs.nyu.edu/`).

2.2. **Timed Weighted Interpreted Systems.** Timed Weighted Interpreted Systems were proposed in [12] to extend interpreted systems (ISs) in order to make possible reasoning about real-time aspects of MASs and to extend ISs to make the reasoning possible about not only temporal and epistemic properties, but also agents's quantitative properties. In the formalism of interpreted systems, each agent is characterised by a set of local states and by a set of local actions that are performed following a local protocol. Given a set of initial states, the system evolves in compliance with an evolution function that determines how the local state of an agent changes as a function of its local state and of the other agents actions. The evolution of all the agents local states describes a set of runs and a set of reachable states. These can be used to interpret formulae involving temporal operators, epistemic operators to reason about what agents know.

Let $\mathbb{N}$ be a set of natural numbers, and $\mathbb{N}_+ = \mathbb{N} \setminus \{0\}$. We assume a finite set $\mathcal{X}$ of variables, called *clocks.* Each clock is a variable ranging over a set of non-negative natural numbers. For $x \in \mathcal{X}$, $\bowtie \in \{<, \leq, =, >, \geq\}$, $c \in \mathbb{N}$ we define a set of clock constraints over $\mathcal{X}$, denoted by $\mathcal{C}(\mathcal{X})$, The constraints are conjunctions of comparisons of a clock with a time constant c from the set of natural numbers $\mathbb{N}$, generated by the following grammar:

$$\mathfrak{cc} := \mathbf{true} \mid x \bowtie c \mid \mathfrak{cc} \wedge \mathfrak{cc}.$$

A clock valuation v of $\mathcal{X}$ is a total function from $\mathcal{X}$ into the set of natural numbers. The set of all the clock valuations is denoted by $\mathbb{N}^{\mathcal{X}}$. For $\mathcal{X}' \subseteq \mathcal{X}$, the valuation which assigns the value 0 to all clocks is defined as: $\forall_{x \in \mathcal{X}'} v'(x) = 0$ and $\forall_{x \in \mathcal{X} \setminus \mathcal{X}'} v'(x) = v(x)$. For $v \in \mathbb{N}^{\mathcal{X}}$, $succ(v)$ is the clock valuation of $\mathcal{X}$ that assigns the value $v(x) + 1$ to each clock x. A clock valuation v satisfies a clock constraint $\mathfrak{cc}$, written as $v \models \mathfrak{cc}$, iff $\mathfrak{cc}$ evaluates to true using the clock values given by v.

Let $\mathcal{A} = \{1, \ldots, n\}$ denote a non-empty and finite set of agents, and $\mathcal{E}$ be a special agent that is used to model the environment in which the agents operate, and $\mathcal{AP} = \bigcup_{\mathbf{i} \in \mathcal{A} \cup \{\mathcal{E}\}} \mathcal{AP}_{\mathbf{i}}$ be a set of atomic formulae, such that $\mathcal{AP}_{\mathbf{i}_1} \bigcap \mathcal{AP}_{\mathbf{i}_2} = \emptyset$ for all $\mathbf{i}_1, \mathbf{i}_2 \in \mathcal{A} \cup \{\mathcal{E}\}$.

2.3. **TWISs.** A *timed weighted interpreted system* is a tuple $(\{L_{\mathbf{i}}, Act_{\mathbf{i}}, \mathcal{X}_{\mathbf{i}}, P_{\mathbf{i}}, \mathcal{V}_{\mathbf{i}}, \mathcal{I}_{\mathbf{i}}, d_{\mathbf{i}}\}_{\mathbf{i} \in \mathcal{A} \cup \{\mathcal{E}\}}, \{t_{\mathbf{i}}\}_{\mathbf{i} \in \mathcal{A}}, \{t_{\mathcal{E}}\}, \iota)$, where $L_{\mathbf{i}}$ is a non-empty set of *local states* of the agent $\mathbf{i}$, $\iota \subseteq S$ is a non-empty set of initial states, $Act_{\mathbf{i}}$ is a non-empty set of *possible actions* of the agent $\mathbf{i}$, $Act = Act_1 \times \ldots \times Act_n \times Act_{\mathcal{E}}$ is the set of *joint actions*, $\mathcal{X}_{\mathbf{i}}$ is a non-empty set of *clocks*, $P_{\mathbf{i}} : L_{\mathbf{i}} \to 2^{Act_{\mathbf{i}}}$ is a *protocol function*, $t_{\mathbf{i}} : L_{\mathbf{i}} \times L_{\mathcal{E}} \times \mathcal{C}(\mathcal{X}_{\mathbf{i}}) \times 2^{\mathcal{X}_{\mathbf{i}}} \times Act \to L_{\mathbf{i}}$ is a (partial) *evolution function* for agents, $t_{\mathcal{E}} : L_{\mathcal{E}} \times \mathcal{C}(\mathcal{X}_{\mathcal{E}}) \times 2^{\mathcal{X}_{\mathcal{E}}} \times Act \to L_{\mathcal{E}}$ is a (partial) *evolution function* for environment, $\mathcal{V}_{\mathbf{i}} : L_{\mathbf{i}} \to 2^{\mathcal{AP}_{\mathbf{i}}}$ is a *valuation function* assigning to each local state a set of propositional variables that are assumed to be true at that state, $\mathcal{I}_{\mathbf{i}}$: $L_{\mathbf{i}} \to \mathcal{C}(\mathcal{X}_{\mathbf{i}})$ is an *invariant function*, that specifies the amount of time the agent $\mathbf{i}$ may spend in a given local state, and $d_{\mathbf{i}} : Act_{\mathbf{i}} \to \mathbb{N}$ is a *weight function*.

We assume that if $\epsilon_{\mathbf{i}} \in P_{\mathbf{i}}(l_{\mathbf{i}})$, then $t_{\mathbf{i}}(l_{\mathbf{i}}, l_{\mathcal{E}}, \varphi_{\mathbf{i}}, \mathcal{X}, (a_1, \ldots, a_n, a_{\mathcal{E}})) = l_{\mathbf{i}}$ for $a_{\mathbf{i}} = \epsilon_{\mathbf{i}}$, any $\varphi_{\mathbf{i}} \in \mathcal{C}(\mathcal{X})$, and any $\mathcal{X} \in 2^{\mathcal{X}_{\mathbf{i}}}$. An element $< l_{\mathbf{i}}, l_{\mathcal{E}}, \varphi_{\mathbf{i}}, \mathcal{X}, a, l'_{\mathbf{i}} >$ represents a transition from the local state $l_{\mathbf{i}}$ to the local state $l'_{\mathbf{i}}$ of agent $\mathbf{i}$ labelled with action a. The invariant condition allows the TWIS to stay at the local state l as long only as the constraint $\mathcal{I}_{\mathbf{i}}(l_{\mathbf{i}})$ is satisfied. The guard φ has to be satisfied to enable the transition.

2.4. **Timed Weighted Model.** For a given TWIS we define a *timed weighted model* (or a *model*) as a tuple $\mathcal{M} = (Act, S, \iota, T, \mathcal{V}, d)$, where: $Act = Act_1 \times \ldots \times Act_n \times Act_{\mathcal{E}}$ is the set of all the joint actions, $S = \prod_{\mathbf{i} \in \mathcal{A} \cup \{\mathcal{E}\}} (L_{\mathbf{i}} \times \mathbb{N}^{\mathcal{X}_{\mathbf{i}}})$ is the set of all the *global states*, $\iota = \prod_{\mathbf{i} \in \mathcal{A} \cup \{\mathcal{E}\}} (\iota_i \times \{0\}^{\mathcal{X}_i})$ is the set of all the *initial* global states, $\mathcal{V} : S \to 2^{\mathcal{AP}}$ is the valuation function defined as $\mathcal{V}(s) = \bigcup_{\mathbf{i} \in \mathcal{A} \cup \{\mathcal{E}\}} \mathcal{V}_{\mathbf{i}}(l_{\mathbf{i}}(s))$, $T \subseteq S \times (Act \cup \mathbb{N}) \times S$ is a transition relation defined by action and time transitions. For $a \in Act$ and $\delta \in \mathbb{N}$: action transition is defined as $(s, a, s') \in T$ (or $s \xrightarrow{a} s'$) iff for all $\mathbf{i} \in \mathcal{A} \cup \mathcal{E}$, there exists a local transition $t_{\mathbf{i}}(l_{\mathbf{i}}(s), \varphi_{\mathbf{i}}, \mathcal{X}', a) = l_{\mathbf{i}}(s')$ such that $v_{\mathbf{i}}(s) \models \varphi_{\mathbf{i}} \wedge \mathcal{I}(l_{\mathbf{i}}(s))$ and $v'_{\mathbf{i}}(s') = v_{\mathbf{i}}(s)[\mathcal{X}' := 0]$ and $v'_{\mathbf{i}}(s') \models \mathcal{I}(l_{\mathbf{i}}(s'))$; time transition is defined as $(s, \delta, s') \in T$ iff for all $\mathbf{i} \in \mathcal{A} \cup \mathcal{E}$, $l_{\mathbf{i}}(s) = l_{\mathbf{i}}(s')$ and $v'_{\mathbf{i}}(s') = v_{\mathbf{i}}(s) + \delta$ and $v'_{\mathbf{i}}(s') \models \mathcal{I}(l_{\mathbf{i}}(s'))$. The "joint" weight function $d : Act \to \mathbb{N}$ is defined as follows: $d((a_1, \ldots, a_n, a_{\mathcal{E}})) = d_1(a_1) + \ldots + d_n(a_n) + d_{\mathcal{E}}(a_{\mathcal{E}})$.

Given a TWIS, one can define for any agent $\mathbf{i}$ the indistinguishability relation $\sim_{\mathbf{i}} \subseteq S \times S$ as follows: $s \sim_{\mathbf{i}} s'$ iff $l_{\mathbf{i}}(s') = l_{\mathbf{i}}(s)$ and $v_{\mathbf{i}}(s') \simeq v_{\mathbf{i}}(s)$. We assume the following definitions of epistemic relations: $\sim^E_\Gamma \overset{def}{=} \bigcup_{\mathbf{i} \in \Gamma} \sim_{\mathbf{i}}$, $\sim^C_\Gamma \overset{def}{=} (\sim^E_\Gamma)^+$ (the transitive closure of $\sim^E_\Gamma$), $\sim^D_\Gamma \overset{def}{=} \bigcap_{\mathbf{i} \in \Gamma} \sim_{\mathbf{i}}$, where $\Gamma \subseteq \mathcal{A}$.

A run in $\mathcal{M}$ is an infinite sequence $\rho = s_0 \overset{\delta_0,a_0}{\longrightarrow} s_1 \overset{\delta_1,a_1}{\longrightarrow} s_2 \overset{\delta_2,a_2}{\longrightarrow} \dots$ of global states such that the following conditions hold for all $i \in \mathbb{N}$: $s_i \in S, a_i \in Act, \delta_i \in \mathbb{N}_+$, and there exists $s_i' \in S$ such that $(s_i, \delta_i, s_i') \in T$ and $(s_i, a_i, s_{i+1}) \in T$. The definition of a run does not permit two consecutive joint actions to be performed one after the other, i.e., between each two joint actions some time must pass; such a run is called *strongly monotonic*.

2.5. **Abstract model.** The set of all the clock valuations is infinite which means that a model has an infinite set of states. We need to abstract the proposed model before we can apply the bounded model checking technique. Let $c_{\mathbf{i}}$ be the largest constant appearing in any enabling condition or state invariants of agent $\mathbf{i}$, and $v, v' \in \mathbb{N}^{|\mathcal{X}|}$ be two clock valuations. We say that $v \simeq_{\mathbf{i}} v'$ iff the following condition holds for each $x \in \mathcal{X}_{\mathbf{i}}$:

$v(x) > c_{\mathbf{i}}$ and $v'(x) > c_{\mathbf{i}}$ or $v(x) \leq c_{\mathbf{i}}$ and $v'(x) \leq c_{\mathbf{i}}$ and $v(x) = v'(x)$.

Next, we define the relation $\simeq$ as follows: $v \simeq v'$ iff $v \simeq_{\mathbf{i}} v'$, for every $\mathbf{i} \in \mathcal{A} \cup \{\mathcal{E}\}$. Obviously, $\simeq$ is an equivalence relation. It is easy to see that equivalent clock valuations satisfy the same clock constraints that occur in TWIS. Basing on this observation one can define the *abstract model* for TWIS. Namely, let $\mathbb{D}_{\mathbf{i}} = \{0, \dots, c_{\mathbf{i}} + 1\}$, and $\mathbb{D} = \bigcup_{\mathbf{i} \in \mathcal{A} \cup \{\mathcal{E}\}} \mathbb{D}_{\mathbf{i}}^{\mathcal{X}_{\mathbf{i}}}$. For any $v \in \mathbb{D}$ let us define the successor $succ(v)$ of v as follows: for each $x \in \mathcal{X}$,

$$succ(v)(x) = \begin{cases} v(x) + 1, & \text{if } x \in \mathcal{X}_{\mathbf{i}} \text{ and } v(x) \leq c_{\mathbf{i}}, \\ \\ v(x), & \text{if } x \in \mathcal{X}_{\mathbf{i}} \text{ and } v(x) > c_{\mathbf{i}}. \end{cases}$$

Now, one can define the *abstract model* as a tuple $\widehat{\mathcal{M}} = (Act, \widehat{S}, \widehat{\iota}, \widehat{T}, \widehat{\mathcal{V}}, d)$, where $\widehat{\iota} = \prod_{\mathbf{i} \in \mathcal{A} \cup \mathcal{E}} (\iota_{\mathbf{i}} \times \{0\}^{|\mathcal{X}_{\mathbf{i}}|})$ is the set of all the initial global states, $\widehat{S} = \prod_{\mathbf{i} \in \mathcal{A} \cup \mathcal{E}} (L_{\mathbf{i}} \times \mathbb{D}_{\mathbf{i}}^{|\mathcal{X}_{\mathbf{i}}|})$ is the set of all the abstract global states. $\widehat{\mathcal{V}} : \widehat{S} \rightarrow 2^{\mathcal{AP}}$ is the valuation function such that: $p \in \widehat{\mathcal{V}}(\widehat{s})$ iff $p \in \bigcup_{\mathbf{i} \in \mathcal{A} \cup \mathcal{E}} \widehat{\mathcal{V}}_{\mathbf{i}}(l_{\mathbf{i}}(\widehat{s}))$ for all $p \in \mathcal{AP}$; and $\widehat{T} \subseteq \widehat{S} \times (Act \cup \tau) \times \widehat{S}$. Let $a \in Act$. Then, action transition is defined as $(\widehat{s}, a, \widehat{s}') \in \widehat{T}$ iff $\forall_{\mathbf{i} \in \mathcal{A}} \exists_{\phi_{\mathbf{i}} \in \mathcal{C}(\mathcal{X}_{\mathbf{i}})} \exists_{\mathcal{X}_{\mathbf{i}}' \subseteq \mathcal{X}_{\mathbf{i}}} (t_{\mathbf{i}}(l_{\mathbf{i}}(\widehat{s}), \phi_{\mathbf{i}}, \mathcal{X}_{\mathbf{i}}', a) = l_{\mathbf{i}}(\widehat{s}')$ and $v_{\mathbf{i}} \models \phi_{\mathbf{i}} \wedge \mathcal{I}(l_{\mathbf{i}}(\widehat{s}))$ and $v_{\mathbf{i}}'(\widehat{s}') = v_{\mathbf{i}}(\widehat{s})[\mathcal{X}_{\mathbf{i}}' := 0]$ and $v_{\mathbf{i}}'(\widehat{s}') \models \mathcal{I}(l_{\mathbf{i}}(\widehat{s}')))$; time transition is defined as $(\widehat{s}, \tau, \widehat{s}') \in \widehat{T}$ iff $\forall_{\mathbf{i} \in \mathcal{A} \cup \mathcal{E}} (l_{\mathbf{i}}(\widehat{s}) = l_{\mathbf{i}}(\widehat{s}'))$ and $v_{\mathbf{i}}(\widehat{s}) \models \mathcal{I}(l_{\mathbf{i}}(\widehat{s}))$ and $succ(v_{\mathbf{i}}(\widehat{s})) \models \mathcal{I}(l_{\mathbf{i}}(\widehat{s})))$ and $\forall_{\mathbf{i} \in \mathcal{A}} (v_{\mathbf{i}}'(\widehat{s}') = succ(v_{\mathbf{i}}(\widehat{s}')))$ and $(v_{\mathcal{E}}'(\widehat{s}') = succ(v_{\mathcal{E}}(\widehat{s})))$. Given an abstract model one can define for any agent $\mathbf{i}$ the indistinguishability relation $\sim_{\mathbf{i}} \subseteq \widehat{S} \times \widehat{S}$ as follows: $\widehat{s} \sim_{\mathbf{i}} \widehat{s}'$ iff $l_{\mathbf{i}}(\widehat{s}') = l_{\mathbf{i}}(\widehat{s})$ and $v_{\mathbf{i}}(\widehat{s}') = v_{\mathbf{i}}(\widehat{s})$. An abstract path $\widehat{\pi}$ in an abstract model is a sequence $\widehat{s}_0 \xrightarrow{b_1} \widehat{s}_1 \xrightarrow{b_2} \widehat{s}_2 \xrightarrow{b_3} \dots$ of transitions such that for each $i > 1$, $b_i \in Act \cup \{\tau\}$ and $b_1 = \tau$ and for each two consecutive transitions at least one of them is a time transition. Next, $\widehat{\pi}[j..m]$ denotes the finite sequence

$\widehat{s}_j \xrightarrow{\delta_{j+1},a_{j+1}} \widehat{s}_{j+1} \xrightarrow{\delta_{j+2},a_{j+2}} \ldots \widehat{s}_m$ with $m-j$ transitions and $m-j+1$ states, and $D\widehat{\pi}[j..m]$ denotes the (cumulative) weight of $\widehat{\pi}[j..m]$ that is defined as $d(a_{j+1}) + \ldots + d(a_m)$ (hence 0 when $j = m$). The set of all the paths starting at $\widehat{s} \in \widehat{S}$ is denoted by $\widehat{\Pi}(\widehat{s})$, and the set of all the paths starting at an initial state is denoted by $\widehat{\Pi} = \bigcup_{\widehat{s}^0 \in \widehat{\iota}} \widehat{\Pi}(\widehat{s}^0)$.

2.5.1. *WELTLK..* Let I be an interval in $\mathbb{N}$ of the form: $[a, b)$ or $[a, \infty)$, for $a, b \in \mathbb{N}$ and $a \neq b$. WELTLK is the existential fragment of WLTLK [11], defined by the grammar: $\varphi ::= \mathbf{true} \mid \mathbf{false} \mid p \mid \neg p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \mathbf{X}_{\mathrm{I}}\varphi \mid \varphi\mathbf{U}_{\mathrm{I}}\varphi \mid \varphi\mathbf{R}_{\mathrm{I}}\varphi \mid \overline{\mathrm{K}}_{\mathbf{i}}\varphi \mid \overline{\mathrm{E}}_{\Gamma}\varphi \mid \overline{\mathrm{D}}_{\Gamma}\varphi \mid \overline{\mathrm{C}}_{\Gamma}\varphi$.
The semantics of WELTLK is the following. A WELTLK formula φ is true along the abstract path $\widehat{\pi}$ in the abstract model $\widehat{\mathcal{M}}$ (in symbols $\widehat{\mathcal{M}}, \widehat{\pi} \models \varphi$) iff $\widehat{\mathcal{M}}, \widehat{\pi}^0 \models \varphi$, where:

- $\widehat{\mathcal{M}}, \widehat{\pi}^m \models \mathbf{X}_{\mathrm{I}}\alpha$ iff $D\widehat{\pi}[m..m+1] \in \mathrm{I}$ and$\widehat{\mathcal{M}}, \widehat{\pi}^{m+1} \models \alpha$,
- $\widehat{\mathcal{M}}, \widehat{\pi}^m \models \alpha\mathbf{U}_{\mathrm{I}}\beta$ iff $(\exists i \geq m)(D\widehat{\pi}[m..i] \in \mathrm{I}$ and $\widehat{\mathcal{M}}, \widehat{\pi}^i \models \beta$ and $(\forall m \leq j < i)\widehat{\mathcal{M}}, \widehat{\pi}^j \models \alpha)$,
- $\widehat{\mathcal{M}}, \widehat{\pi}^m \models \alpha\mathbf{R}_{\mathrm{I}}\beta$ iff $(\forall i \geq m)(D\widehat{\pi}[m..i] \in \mathrm{I}$ implies $\widehat{\mathcal{M}}, \widehat{\pi}^i \models \beta)$ or $(\exists i \geq m)$ $(D\widehat{\pi}[m..i] \in \mathrm{I}$ and $\widehat{\mathcal{M}}, \widehat{\pi}^i \models \alpha$ and $(\forall m \leq j \leq i)\widehat{\mathcal{M}}, \widehat{\pi}^j \models \beta)$,
- $\widehat{\mathcal{M}}, \widehat{\pi}^m \models \overline{\mathrm{K}}_{\mathbf{i}}\alpha$ iff $(\exists \widehat{\pi}' \in \widehat{\Pi})(\exists i \geq 0)(\widehat{\pi}'(i) \sim_{\mathbf{i}} \widehat{\pi}(m)$ and $\widehat{\mathcal{M}}, \widehat{\pi}'^i \not\models \alpha)$.
- $\widehat{\mathcal{M}}, \widehat{\pi}^m \models \overline{Y}_{\Gamma}\alpha$ iff $(\exists \widehat{\pi}' \in \widehat{\Pi})(\exists i \geq 0)(\widehat{\pi}'(i) \sim_{\Gamma}^{Y} \widehat{\pi}(m)$ and $\widehat{\mathcal{M}}, \widehat{\pi}'^i \not\models \alpha)$, where $Y \in \{\mathrm{D}, \mathrm{E}, \mathrm{C}\}$.

Theorem 1. *Let $TWIS$ be a timed weighted interpreted system, $\mathcal{M}$ be a concrete model for $TWIS$, φ WELTLK formula, and $\widehat{\mathcal{M}}$ the abstract model for $TWIS_{\varphi}$. Then, $\mathcal{M} \models \mathbf{E}\varphi$ iff $\widehat{\mathcal{M}} \models \mathbf{E}\varphi$.*

3. Bounded Model Checking

The SMT-based Bounded Model Checking (BMC) is a popular model checking technique for the verification of concurrent systems, real-time systems and multi-agent systems.. We have given a model $\mathcal{M}$, an existential modal formula φ, and a non-negative bound k, the SMT-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 SMT-based bounded model checking algorithms generate a quantifier-free first order formula which is satisfiable if and only if the mentioned set of paths exist. The quantifier-free first order formula is usually obtained as a combination of an encoding of the unfolding of the transition relation of the given model, and an SMT- encoding of the property in question. If the generated quantifier-free first order formula is not satisfiable, then k is incremented until either the problem becomes intractable due to the complexity of solving the corresponding SMT instance, or k reaches the upper bound of the bounded model checking problem for the language under consideration.

The SMT-based BMC is based on so called bounded semantics, which are the base of translations of specifications to the SMT-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 SMT-based BMC method for WELTLK was introduced in [13]. In this paper we use implementation of the SMT-based BMC for WELTLK that was presented in [14].

4. Experimental Results

In this section we experimentally evaluate the performance of four SMT-solvers for SMT-based BMC encoding for WELTLK over the TWIS semantics. We compare our experimental results with each other. We have conducted the experiments using two benchmarks: the timed weighted generic pipeline paradigm (TWGPP) TWIS model [12] and weighted timed train controller system (TWTCS) TWIS model [14]. We would like to point out that both benchmarks are very useful and scalable examples.
TWGPP. The specifications we consider are as follows:

- $\varphi_1 = \mathbf{F}_{[0,\infty)}(\mathbf{G}_{[0,\infty)}(\neg ConsReady))$, *which expresses that there exist a computation that always Consumer is ready to ready to consuming the data.*
- $\varphi_2 = \mathrm{K}_P\mathbf{G}_{[Min,Min+1)}ConsFree$, which expresses that Producer knows that always the cost of receiving by Consumer the commodity is Min.
- $\varphi_3 = \mathrm{K}_P\mathbf{G}\big(ProdSend \rightarrow \mathrm{K}_C\mathrm{K}_P\mathbf{F}_{[0,Min-d_P(Produce))}ConsFree\big)$, *which states that Producer knows that always if she/he produces a commodity, then Consumer knows that Producer knows that Consumer has received the commodity and the cost is less than* $Min - d_P(Produce)$.
- $\varphi_4 = \mathrm{K}_C\mathbf{G}\big(ProdReady \rightarrow \mathbf{X}_{[d_P(Produce),d_P(Produce)+1)}ProdSend\big)$, *which expresses that Consumer knows that the cost of producing of a commodity by Producer is* $d_P(Produce)$.
- $\varphi_5 = \mathbf{G}\big(ProdSend \rightarrow \mathrm{K}_C\mathrm{K}_P(ConsStart \vee ConsReady)\big)$, *which express that always if Producer produces the data then Consumer knows that Producer knows that Consumer started his job or Consumer is ready to consuming the data.*

TWTCS. The specifications we consider are as follows for $w \in \{1, 10^6\}$:

- $\phi_1 = \mathbf{G}_{[28\cdot w,\infty)}\left(\bigwedge_{j=1}^{n-1}\bigwedge_{j=i+1}^{n}(tunnel_i \vee tunnel_j)\right)$, *which expresses that the system satisfies mutual exclusion property.*
- $\phi_2 = \mathbf{G}_{[0,15\cdot w)}\left(tunnel_1 \rightarrow \mathrm{K}_{T_1}\left(\mathbf{G}(\bigwedge_{j=2}^{n}\neg tunnel_j)\right)\right)$, *which expresses that always if the $Train_1$ enters its critical section, then it knows that always in the future no other train will enter its critical section.*

Note, that we describe specifications as universal formulae, for which we verify the corresponding counterexample formulae that are interpreted existentially and belong to WELTLK. Moreover, for every specification given, the corresponding WELTLK formula holds in the model of the benchmark.

4.1. **Performance evaluation.** We have performed our experimental results on a computer equipped with I7-5500U processor, 12 GB of RAM, and the operating system Ubuntu Linux with the kernel 4.4.0. Our SMT-based algorithm is implemented as standalone program written in the programming language `C++`. We used the state of the art SMT-solvers: Z3, Yices, Mathsat, and CVC4.

We will not analyse experimental results for CVC4 and MathSAT. The time and memory usage for this two SMT-solvers was really high. We did not expect this, because according to SMT-competition this two SMT-solvers are the best one. It was very unexpected.

4.1.1. *TWGPP.*. As one can see from the line charts (Fig. 1) Z3 won this comparison. The number of the considered k-paths is equal to 1. The length of the witness is $2(2n + 3)$, where n is the number of nodes. In every case Yices2 uses less memory than Z3.

Also in this case (Fig. 2) Z3 won this comparison. The number of the considered k-paths is equal to 2. The length of the witness is 6 for $n = 1$, 10 for $n = 2$, 12 for $n = 3$ and $n = 4$, 14 for $n = 5$, 16 for $n = 6$ and $n = 7$, and 18 for $n = 8$ and $n = 9$, where n is the number of nodes.

Yices2 won a comparison for the formulae φ_3 (Fig. 3), φ_4 (Fig. 4) and φ_5 (Fig. 5). The number of the considered k-paths for the formula φ_3 is equal to 3. The length of the witness is $4(n + 2)$, where n is the number of nodes. The number of the considered k-paths for the formula φ_4 is equal to 4. The length of the witness is 6. The number of the considered k-paths for the formula φ_5 is equal to 2. The length of the witness is 8.

4.1.2. *TWTCS.*. In this case Z3 performed better than Yices2. As one can see from the line charts (Fig. 6) Z3 won this comparison. The number of the considered k-paths for the formula ψ_1 is equal to 1. The length of the witness is 14. The number of the considered k-paths for the formula ψ_2 is equal to 2. The length of the witness is 14.

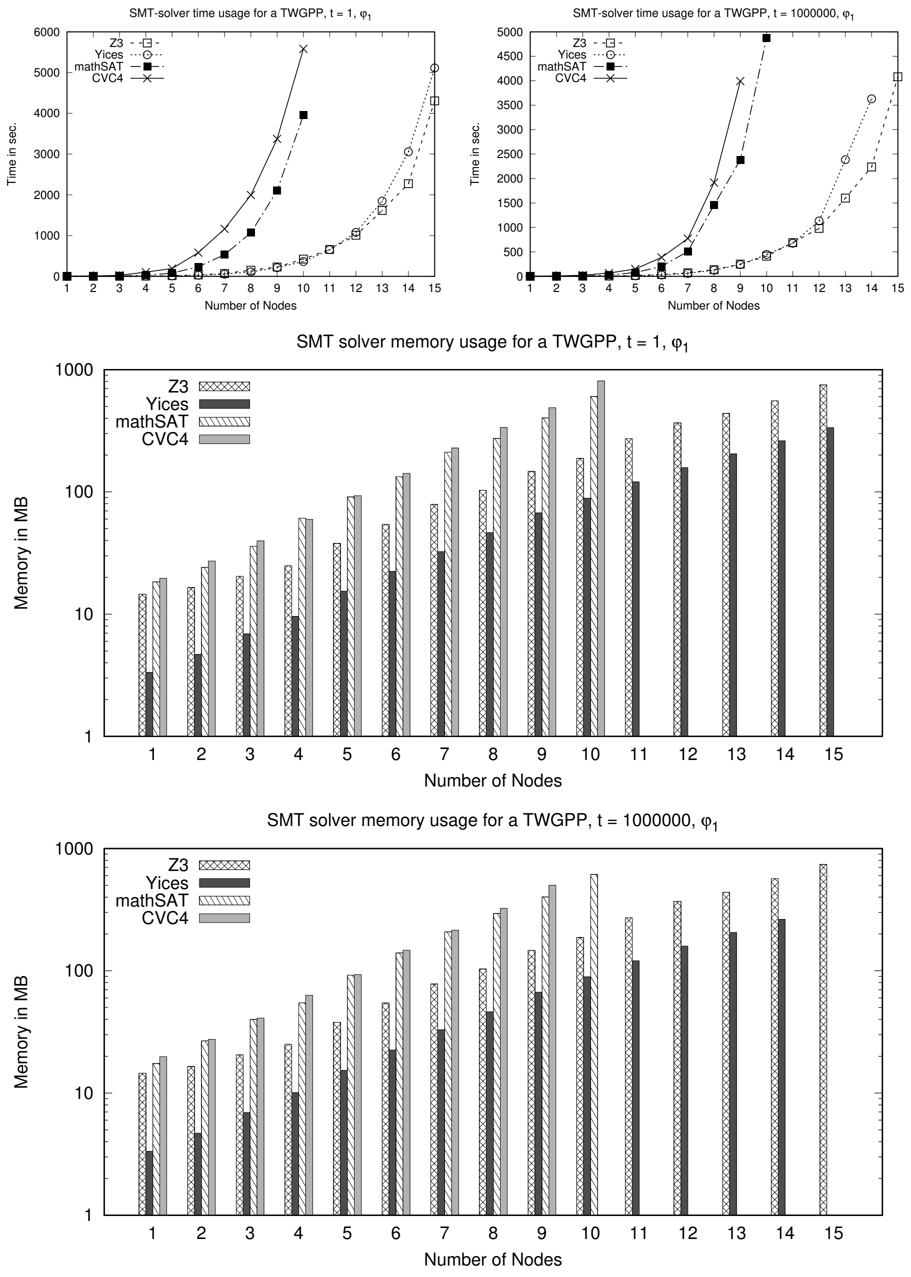

FIGURE 1. φ_1: SMT-based BMC: TWGPP with n nodes.

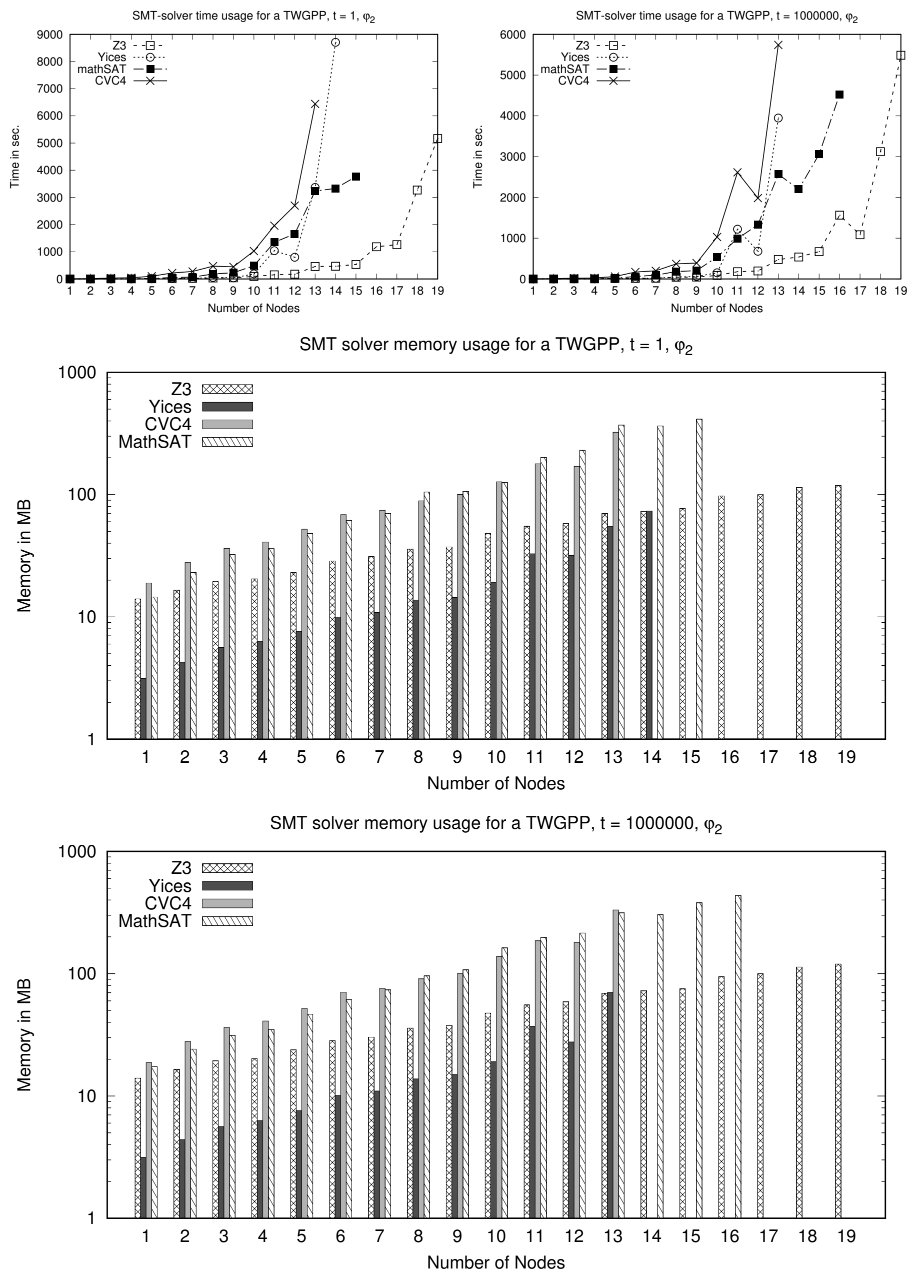

FIGURE 2. φ_2: SMT-based BMC: TWGPP with n nodes.

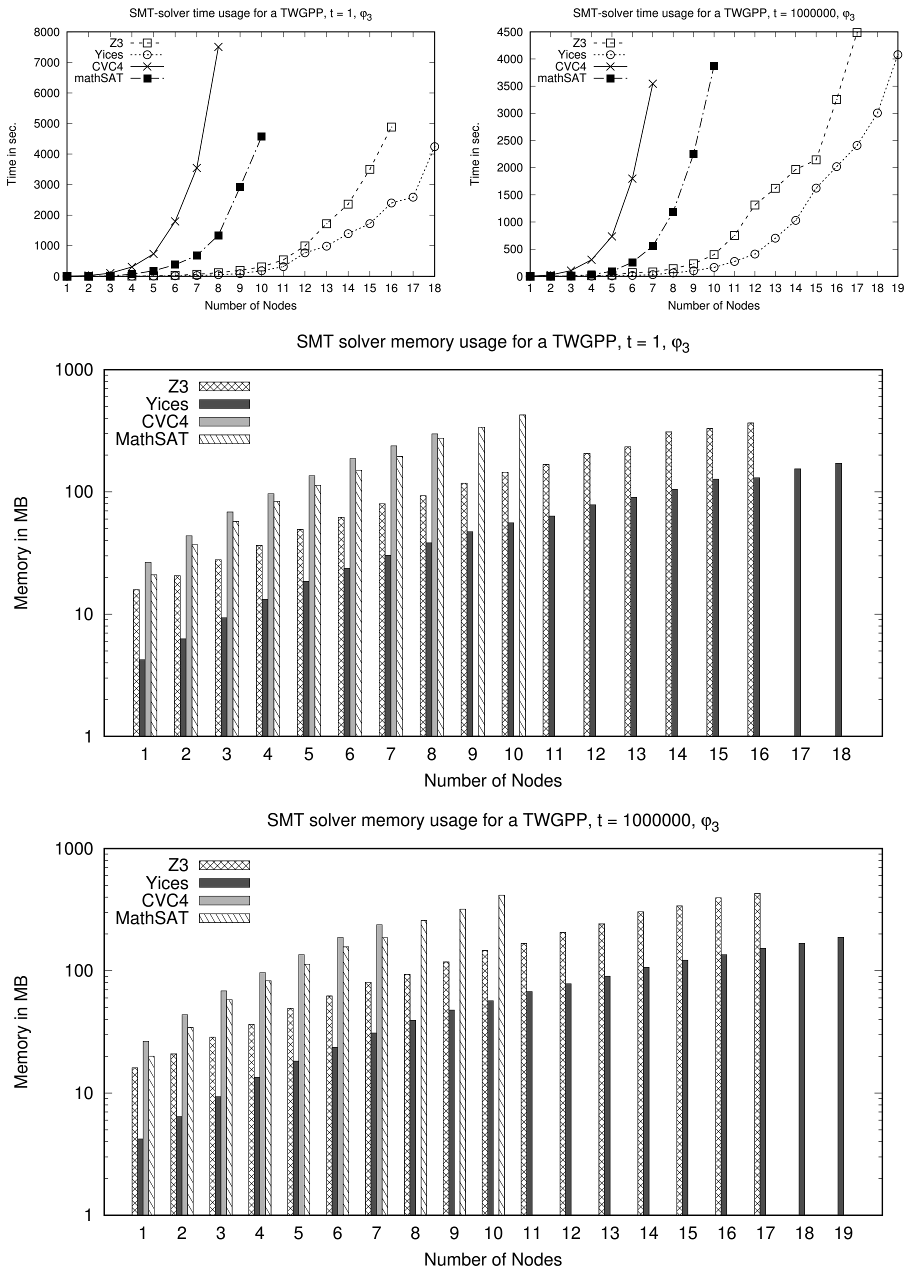

Figure 3. φ_3: SMT-based BMC: TWGPP with n nodes.

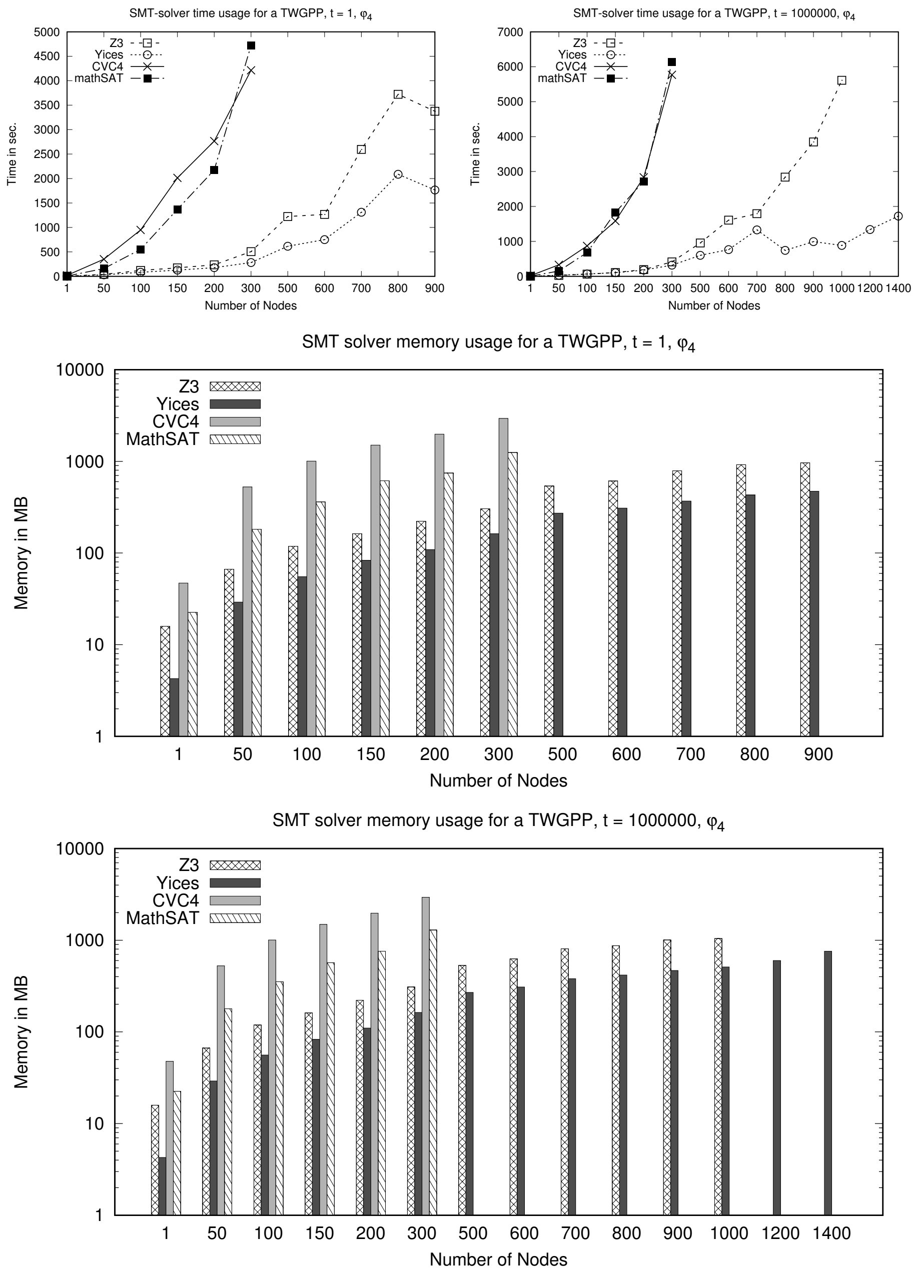

FIGURE 4. φ_4: SMT-based BMC: TWGPP with n nodes.

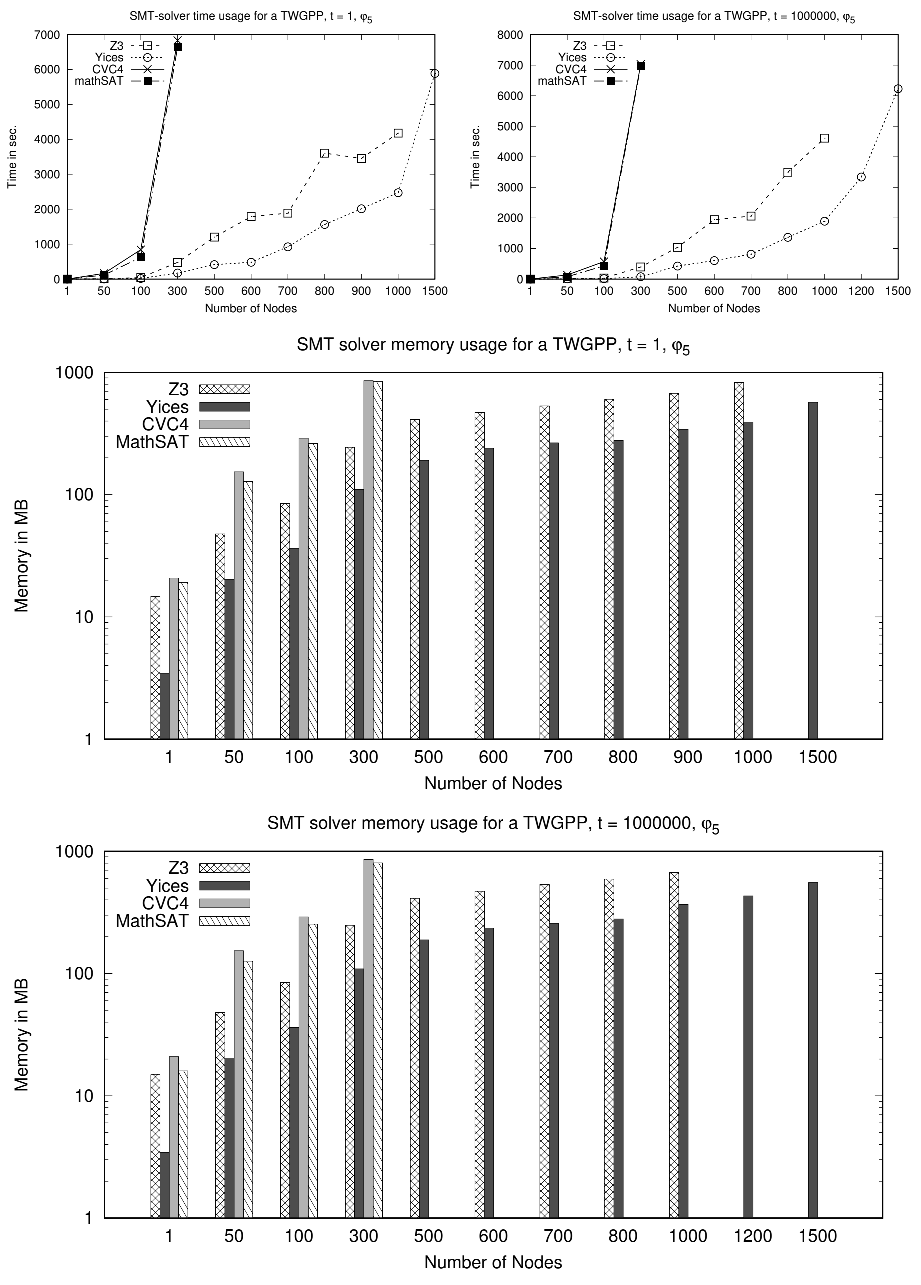

FIGURE 5. φ_5: SMT-based BMC: TWGPP with n nodes.

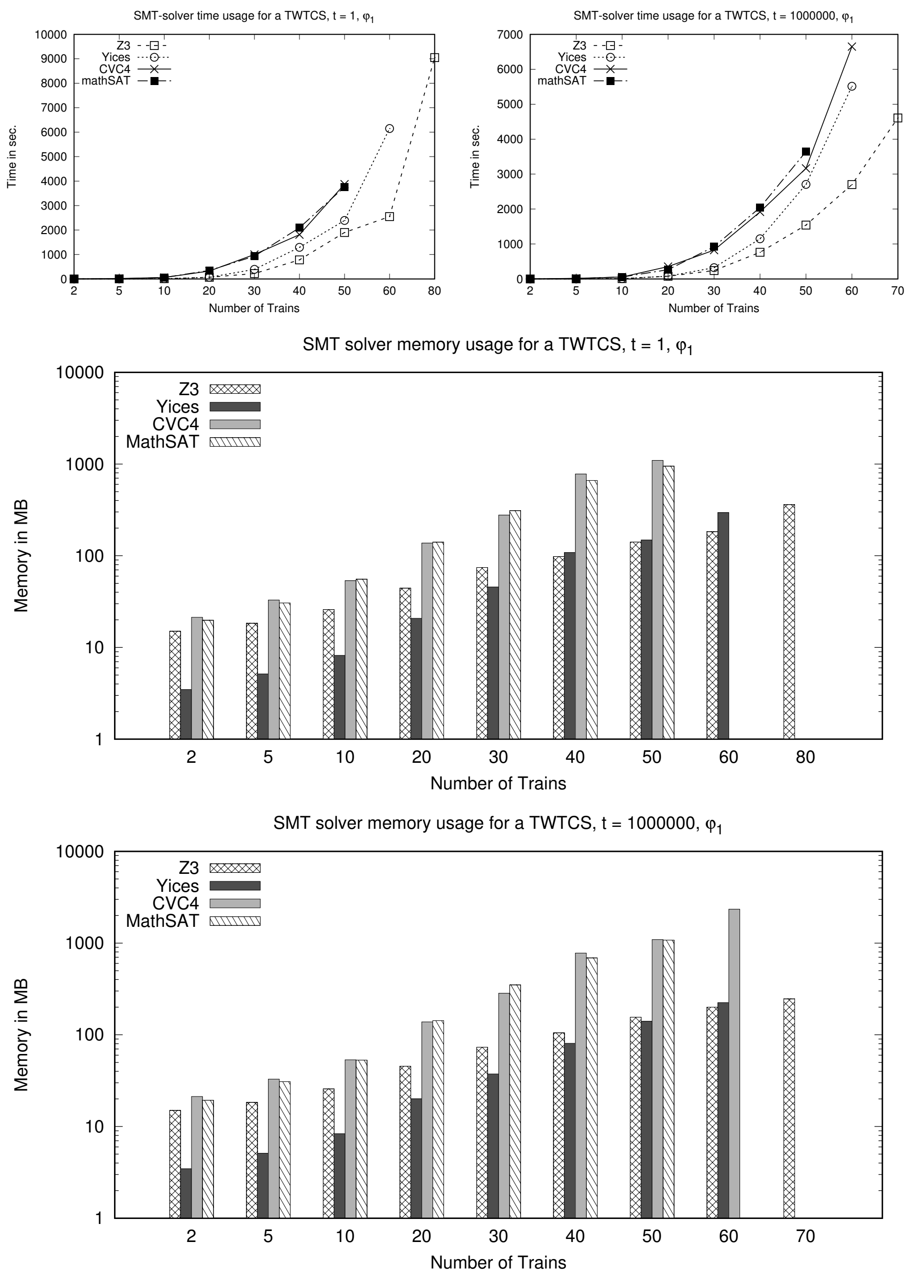

FIGURE 6. SMT-based BMC: TWTCS with n trains.

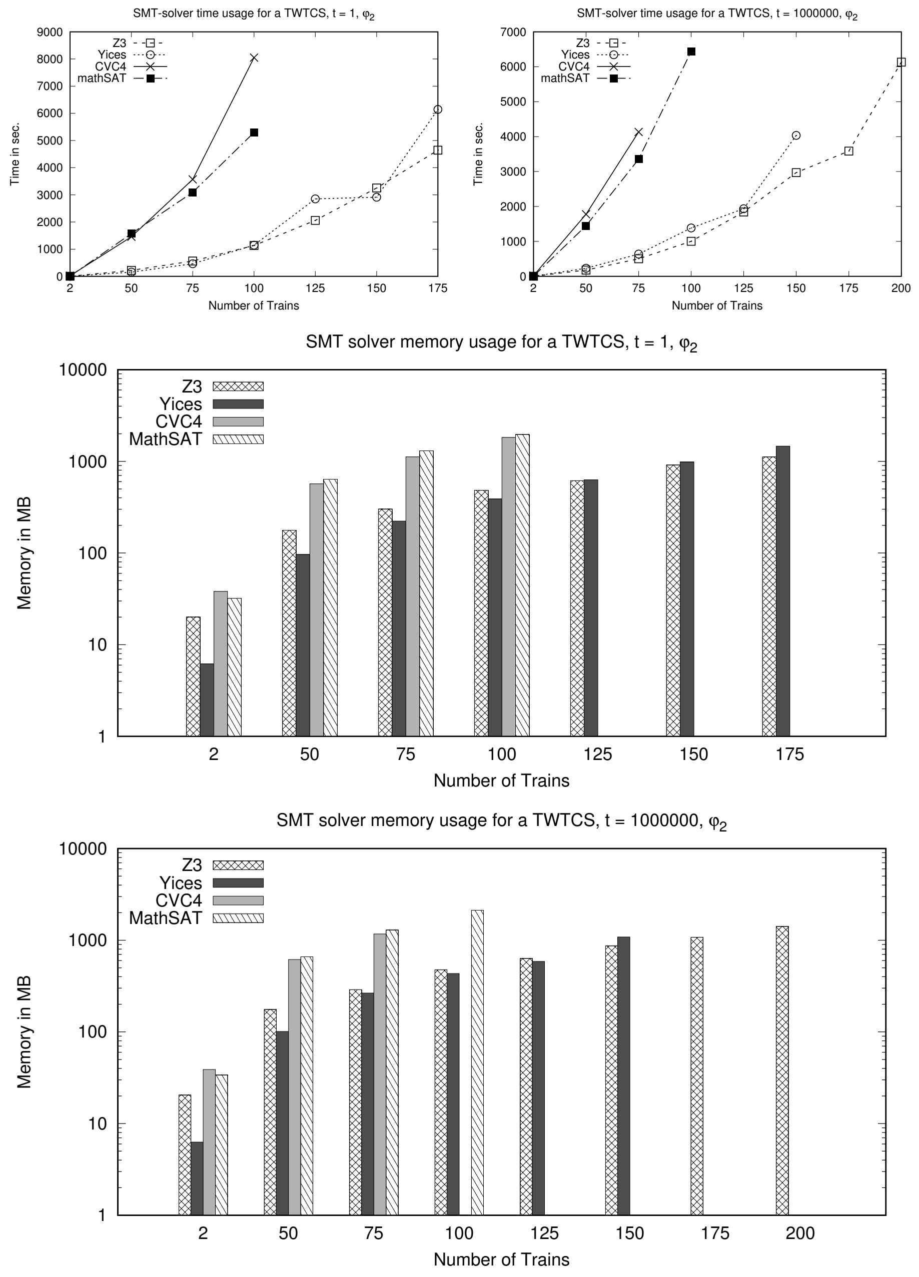

FIGURE 7. SMT-based BMC: TWTCS with n trains.

5. Conclusions

We have experimentally evaluated the SMT-based BMC approach for WELTLK interpreted over the timed weighted interpreted systems. The experimental results show that the Z3 SMT-solver in more cases is better than Yices2 SMT-solver when we compare time usage, but when we compare memory usage Yices2 is better. In general the Z3 approach appears to be superior for the both systems, while the Yices2 approach appears to be superior only for three formulae for the TWGPP system. This is a novel and interesting result.

In the tables 5 and 2 we showed the winners of our comparison.

SMT-solver time usage				
formula	Yices2		Z3	
	basic weights		basic weights	
	1	10^6	1	10^6
φ_1			x	x
φ_2			x	x
φ_3	x	x		
φ_4	x	x		
φ_5	x	x		
ϕ_1			x	x
ϕ_2			x	x

Table 1. The winners for all the benchmarks.

SMT-solver memory usage				
formula	Yices2		Z3	
	basic weights		basic weights	
	1	10^6	1	10^6
φ_1	x	x		
φ_2	x	x		
φ_3	x	x		
φ_4	x	x		
φ_5	x	x		
ϕ_1			x	x
ϕ_2			x	x

Table 2. The winners for all the benchmarks.

References

[1] Rajeev Alur and David L. Dill. The theory of timed automata. In *Proceedings of REX Workshop*, pages 45–73, 1991.

[2] Clark Barrett, Christopher L. Conway, Morgan Deters, Liana Hadarean, Dejan Jovanovic, Tim King, Andrew Reynolds, and Cesare Tinelli. CVC4. In *Computer Aided Verification - 23rd International Conference, CAV 2011, Snowbird, UT, USA, July 14-20, 2011. Proceedings*, pages 171–177, 2011.

[3] Nikolaj Bjørner, Kenneth L. McMillan, and Andrey Rybalchenko. Program verification as satisfiability modulo theories. In *10th International Workshop on Satisfiability Modulo Theories, SMT 2012, Manchester, UK, June 30 - July 1, 2012*, pages 3–11, 2012.

[4] Patricia Bouyer, Nicolas Markey, and Ocan Sankur. Robust weighted timed automata and games. In *Proceedings of FORMATS 2013*, pages 31–46, 2013.

[5] Alessandro Cimatti, Alberto Griggio, Bastiaan Joost Schaafsma, and Roberto Sebastiani. The mathsat5 SMT solver. In *Tools and Algorithms for the Construction and Analysis of Systems - 19th International Conference, TACAS 2013, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2013, Rome, Italy, March 16-24, 2013. Proceedings*, pages 93–107, 2013.

[6] E. Clarke, O. Grumberg, and D. Peled. *Model Checking.* MIT Press, 1999.

[7] Bruno Dutertre. Yices 2.2. In *Computer Aided Verification - 26th International Conference, CAV 2014, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 18-22, 2014. Proceedings*, pages 737–744, 2014.

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

[9] Kim G. Larsen and Radu Mardare. Complete proof systems for weighted modal logic. *Theor. Comput. Sci.*, 546:164–175, 2014.

[10] B. Woźna-Szcześniak and A. Zbrzezny. Checking EMTLK properties of timed interpreted systems via bounded model checking. *Studia Logica*, pages 1–38, 2015.

[11] B. Woźna-Szcześniak, A. M. Zbrzezny, and A. Zbrzezny. SAT-based bounded model checking for weighted interpreted systems and weighted linear temporal logic. In *Proceedings of PRIMA'2013*, volume 8291 of *LNAI*, pages 355–371. Springer-Verlag, 2013.

[12] Agnieszka M. Zbrzezny and Andrzej Zbrzezny. Checking WECTLK Properties of Timed Real-Weighted Interpreted Systems via SMT-Based Bounded Model Checking. In *Proceedings of EPIA 2015*, volume 9273 of *LNCS*, pages 638–650. Springer, 2015.

[13] Agnieszka M. Zbrzezny and Andrzej Zbrzezny. Checking WELTLK properties of weighted interpreted systems via smt-based bounded model checking. In *Proceedings of PRIMA 2015*, volume 9387 of *LNCS*, pages 660–669. Springer, 2015.

[14] Agnieszka M. Zbrzezny, Andrzej Zbrzezny, and Franco Raimondi. *Efficient Model Checking Timed and Weighted Interpreted Systems Using SMT and SAT Solvers*, pages 45–55. Springer International Publishing, 2016.

Received: November 2016

Agnieszka M. Zbrzezny
Jan Długosz University in Częstochowa
Institute of Mathematics and Computer Science
al. Armii Krajowej 13/15, 42-200 Częstochowa, Poland
E-mail address: `agnieszka.zbrzezny@ajd.czest.pl`