Checking WECTLK Properties of TRWISs via SMT-based Bounded Model Checking* †

Agnieszka M. Zbrzezny and Andrzej Zbrzezny

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

Abstract

In this paper, we present a Satisfiability Modulo Theory based (SMT-based) bounded model checking (BMC) method for Timed Real-Weighted Interpreted Systems and for the existential fragment of the Weighted Epistemic Computation Tree Logic. SMT-based bounded model checking consists in translating the existential model checking problem for a modal logic and for a model to the satisfiability problem of a quantifier-free first-order formula. We have implemented the SMT-BMC method and performed the BMC algorithm on Timed Weighted Generic Pipeline Paradigm benchmark. The preliminary experimental results demonstrate the feasibility of the method. To perform the experiments, we used the state of the art SMT-solver Z3.

1998 ACM Subject Classification D.2.4 Software/Program Verification

Keywords and phrases SMT, Timed Real-Weighted Interpreted Systems, Bounded Model Checking

Digital Object Identifier 10.4230/OASIcs.ICCSW.2015.78

1 Introduction

The formalism of *interpreted systems* (ISs) was introduced in [1] to model multi-agent systems (MASs) [6], which are intended for reasoning about the agents' epistemic and temporal properties. *Timed interpreted systems* (TIS) was proposed in [8] to extend interpreted systems in order to make possible reasoning about real-time aspects of MASs. The formalism of weighted interpreted systems (WISs) [9] extends ISs to make the reasoning possible about not only temporal and epistemic properties, but also agents's quantitative properties.

Multi-agent systems (MASs) are composed of many intelligent agents that interact with each other. The agents can share a common goal or they can pursue their own interests. Also, the agents may have a deadline or other timing constraints to achieve intended targets. As it was shown in [1], knowledge is a useful concept for analysing the information state and the behaviour of agents in multi-agent systems. In this paper, we consider the existential fragment of a weighted epistemic computation tree logic (WECTLK) interpreted over Timed Real-Weighted Interpreted Systems (TRWISs).

To the best of our knowledge, there is no work that considers SMT-based BMC methods to check multi-agent systems modelled by means of timed weighted interpreted systems. Thus, in this paper such a method is offered.

* Partly supported by National Science Centre under the grant No. 2014/15/N/ST6/05079.
† A longer version of this paper will appear in the proceedings of the 17th Portuguese Conference on Artificial Intelligence (EPIA'2015) in September 2015.

2015 Imperial College Computing Student Workshop (ICCSW 2015).
Editors: Claudia Schulz and Daniel Liew; pp. 78–86

OpenAccess Series in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

We do not compare our results with other model checkers for MASs, e.g. MCMAS [4] or MCK [2], simply because they do not support the WECTLK language and the timed weighted interpreted systems.

Firstly, we define and implement an SMT-based BMC method for WECTLK and for TRWISs. Secondly, we report on the initial experimental evaluation of our SMT-based BMC method. To this aim, we use a scalable benchmark: the *timed weighted generic pipeline paradigm* [7, 9].

2 Preliminaries

Let $\mathbb{N}$ be a set of natural numbers, $\mathbb{N}_+ = \mathbb{N} \setminus \{0\}$, $\mathbb{R}$ be the set of non-negative real numbers, and $\mathcal{X}$ be a finite set of non-negative natural variables, called *clocks* ranging over a set of non-negative natural numbers. A clock valuation is a function $v : \mathcal{X} \to \mathbb{N}$ that assigns to each clock $x \in \mathcal{X}$ a non-negative natural value $v(x)$. A set of all the clock valuations is denoted by $\mathbb{N}^{|\mathcal{X}|}$. The valuation $v' = v[\mathcal{X}' := 0]$, for $\mathcal{X}' \subseteq \mathcal{X}$ 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 $\delta \in \mathbb{N}$, $v + \delta$ denotes the valuation that assigns the value $v(x) + \delta$ to each clock x.

The grammar $\varphi := \mathbf{true} \mid x < c \mid x \leq c \mid x = c \mid x \geq c \mid x > c \mid \varphi \wedge \varphi$ generates the set $\mathcal{C}(\mathcal{X})$ of clock constraints over $\mathcal{X}$, where $x \in \mathcal{X}$ and $c \in \mathbb{N}$. A clock valuation v satisfies a clock constraint φ, written as $v \models \varphi$, iff φ evaluates to be true using the clock values given by v.

Let c_{max} be a constant and $v, v' \in \mathbb{N}^{|\mathcal{X}|}$ two clock valuation. We say that $v \simeq v'$ iff the following condition holds for each $x \in \mathcal{X}$: $v(x) > c_{max}$ and $v'(x) > c_{max}$ or $v(x) \leq c_{max}$ and $v'(x) \leq c_{max}$ and $v(x) = v'(x)$ The clock valuation v' such that for each clock $x \in \mathcal{X}$, $v'(x) = v(x) + 1$ if $v(x) \leq c_{max}$, and $v'(x) = c_{max} + 1$ otherwise, is called a time successor of v (written $succ(v)$).

TRWISs. Let $Ag = \{1, \ldots, n\}$ denotes 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 $\mathcal{PV} = \bigcup_{\mathbf{c} \in Ag \cup \{\mathcal{E}\}} \mathcal{PV}_{\mathbf{c}}$ be a set of propositional variables, such that $\mathcal{PV}_{\mathbf{c}_1} \bigcap \mathcal{PV}_{\mathbf{c}_2} = \emptyset$ for all $\mathbf{c}_1, \mathbf{c}_2 \in Ag \cup \{\mathcal{E}\}$. The *timed real-weighted interpreted system* (TRWIS) is a tuple $(\{L_{\mathbf{c}}, Act_{\mathbf{c}}, \mathcal{X}_{\mathbf{c}}, P_{\mathbf{c}}, t_{\mathbf{c}}, \mathcal{V}_{\mathbf{c}}, \mathcal{I}_{\mathbf{c}}, d_{\mathbf{c}}\}_{\mathbf{c} \in Ag \cup \{\mathcal{E}\}}, \iota)$, where $L_{\mathbf{c}}$ is a non-empty set of *local states* of the agent $\mathbf{c}$, $S = L_1 \times \ldots \times L_n \times L_{\mathcal{E}}$ is the set of all global states, $\iota \subseteq S$ is a non-empty set of initial states, $Act_{\mathbf{c}}$ is a non-empty set of *possible actions* of the agent $\mathbf{c}$, $Act = Act_1 \times \ldots \times Act_n \times Act_{\mathcal{E}}$ is the set of *joint actions*, $\mathcal{X}_{\mathbf{c}}$ is a non-empty set of *clocks*, $P_{\mathbf{c}} : L_{\mathbf{c}} \to 2^{Act_{\mathbf{c}}}$ is a *protocol function*, $t_{\mathbf{c}} : L_{\mathbf{c}} \times \mathcal{C}(\mathcal{X}_{\mathbf{c}}) \times 2^{\mathcal{X}_c} \times Act \to L_{\mathbf{c}}$ is a (partial) *evolution function*, $\mathcal{V}_{\mathbf{c}} : L_{\mathbf{c}} \to 2^{\mathcal{PV}}$ is a *valuation function* assigning to each local state a set of propositional variables that are assumed to be true at that state, $\mathcal{I}_{\mathbf{c}}$: $L_{\mathbf{c}} \to \mathcal{C}(\mathcal{X}_{\mathbf{c}})$ is an *invariant function*, that specifies an amount of time the agent $\mathbf{c}$ may spend in a given local state, and $d_{\mathbf{c}} : Act_{\mathbf{c}} \to \mathbb{R}$ is a *weight function*.

For a given TRWIS we define a *timed real-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,
- $S = (L_1 \times \mathbb{N}^{|\mathcal{X}_1|}) \times \ldots \times (L_n \times \mathbb{N}^{|\mathcal{X}_n|)}) \times (L_{\mathcal{E}} \times \mathbb{N}^{|\mathcal{X}_{\mathcal{E}}|})$ is the set of all the *global states*,
- $\iota = (\iota_1 \times \{0\}^{|\mathcal{X}_1|}) \times \ldots \times (\iota_n \times \{0\}^{|\mathcal{X}_n|}) \times (\iota_{\mathcal{E}} \times (\{0\}^{|\mathcal{X}_{\mathcal{E}}|})$ is the set of all the *initial* global states,
- $\mathcal{V} : S \to 2^{\mathcal{PV}}$ is the valuation function defined as $\mathcal{V}(s) = \bigcup_{\mathbf{c} \in Ag \cup \{\mathcal{E}\}} \mathcal{V}_{\mathbf{c}}(l_{\mathbf{c}}(s))$, $T \subseteq S \times (Act \cup \mathbb{N}) \times S$ is a transition relation defined by action and time transitions. For $a \in Act$ and $\delta \in \mathbb{N}$:

1. action transition: $(s,a,s') \in T$ (or $s \xrightarrow{a} s'$) iff for all $\mathbf{c} \in Ag \cup \mathcal{E}$, there exists a local transition $t_{\mathbf{c}}(l_{\mathbf{c}}(s), \varphi_{\mathbf{c}}, \mathcal{X}', a) = l_{\mathbf{c}}(s')$ such that $v_{\mathbf{c}}(s) \models \varphi_{\mathbf{c}} \wedge \mathcal{I}(l_{\mathbf{c}}(s))$ and $v'_{\mathbf{c}}(s') = v_{\mathbf{c}}(s)[\mathcal{X}' := 0]$ and $v'_{\mathbf{c}}(s') \models \mathcal{I}(l_{\mathbf{c}}(s'))$;
2. time transition $(s,\delta,s') \in T$ iff for all $\mathbf{c} \in Ag \cup \mathcal{E}$, $l_{\mathbf{c}}(s) = l_{\mathbf{c}}(s')$ and $v'_{\mathbf{c}}(s') = v_{\mathbf{c}}(s) + \delta$ and $v'_{\mathbf{c}}(s') \models \mathcal{I}(l_{\mathbf{c}}(s'))$.

- $d : Act \to \mathbb{R}$ is the "joint" weight function 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 TRWIS, one can define for any agent $\mathbf{c}$ the indistinguishability relation $\sim_{\mathbf{c}} \subseteq S \times S$ as follows: $s \sim_{\mathbf{c}} s'$ iff $l_{\mathbf{c}}(s') = l_{\mathbf{c}}(s)$ and $v_{\mathbf{c}}(s') \simeq v_{\mathbf{c}}(s)$

A run in $\mathcal{M}$ is an infinite sequence $\rho = s_0 \xrightarrow{\delta_0,a_0} s_1 \xrightarrow{\delta_1,a_1} s_2 \xrightarrow{\delta_2,a_2} \ldots$ 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 $(s_i, \delta, s'_i) \in T$ and $(s_i, a, s_{i+1}) \in T$. Notice that 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*.

Abstract model. Let $\mathbb{D}_{\mathbf{c}} = \{0, \ldots, c_{\mathbf{c}} + 1\}$ with $c_{\mathbf{c}}$ be the largest constant appearing in any enabling condition or state invariants of agent $\mathbf{c}$ and $\mathbb{D} = \bigcup_{\mathbf{c} \in Ag \cup \mathcal{E}} \mathbb{D}_{\mathbf{c}}^{|\mathcal{X}_{\mathbf{c}}|}$. A tuple $\widehat{\mathcal{M}} = (Act, \widehat{S}, \widehat{\iota}, \widehat{T}, \widehat{\mathcal{V}}, d)$, is an *abstract model*, where $\widehat{\iota} = \prod_{\mathbf{c} \in Ag \cup \mathcal{E}} \iota_{\mathbf{c}} \times \{0\}^{|\mathcal{X}_{\mathbf{c}}|}$ is the set of all the initial global states, $\widehat{S} = \prod_{\mathbf{c} \in Ag \cup \mathcal{E}} L_{\mathbf{c}} \times \mathbb{D}_{\mathbf{c}}^{|\mathcal{X}_{\mathbf{c}}|}$ is the set of all the abstract global states. $\widehat{\mathcal{V}} : \widehat{S} \to 2^{\mathcal{PV}}$ is the valuation function such that: $p \in \widehat{\mathcal{V}}(s)$ iff $p \in \bigcup_{\mathbf{c} \in Ag \cup \mathcal{E}} \widehat{\mathcal{V}}_{\mathbf{c}}(l_{\mathbf{c}}(s))$ for all $p \in \mathcal{PV}$; and $\widehat{T} \subseteq \widehat{S} \times (Act \cup \tau) \times \widehat{S}$. Let $a \in Act$. Then,

1. Action transition: $(s,a,s') \in \widehat{T}$ iff $\forall_{\mathbf{c} \in Ag} \exists_{\phi_{\mathbf{c}} \in \mathcal{C}(\mathcal{X}_{\mathbf{c}})} \exists_{\mathcal{X}'_{\mathbf{c}} \subseteq \mathcal{X}_{\mathbf{c}}} (t_{\mathbf{c}}(l_{\mathbf{c}}(s), \phi_{\mathbf{c}}, \mathcal{X}'_{\mathbf{c}}, a) = l_{\mathbf{c}}(s')$ and $v_{\mathbf{c}} \models \phi_{\mathbf{c}} \wedge \mathcal{I}(l_{\mathbf{c}}(s))$ and $v'_{\mathbf{c}}(s') = v_{\mathbf{c}}(s)[\mathcal{X}'_{\mathbf{c}} := 0]$ and $v'_{\mathbf{c}}(s') \models \mathcal{I}(l_{\mathbf{c}}(s')))$
2. Time transition: $(s,\tau,s') \in \widehat{T}$ iff $\forall_{\mathbf{c} \in Ag \cup \mathcal{E}} (l_{\mathbf{c}}(s) = l_{\mathbf{c}}(s'))$ and $v_{\mathbf{c}}(s) \models \mathcal{I}(l_{\mathbf{c}}(s))$ and $succ(v_{\mathbf{c}}(s)) \models \mathcal{I}(l_{\mathbf{c}}(s)))$ and $\forall_{\mathbf{c} \in Ag} (v'_{\mathbf{c}}(s') = succ(v_{\mathbf{c}}(s')))$ and $(v'_{\mathcal{E}}(s') = succ(v_{\mathcal{E}}(s)))$.

Given an abstract model one can define for any agent $\mathbf{c}$ the indistinguishability relation $\sim_{\mathbf{c}} \subseteq \widehat{S} \times \widehat{S}$ as follows: $s \sim_{\mathbf{c}} s'$ iff $l_{\mathbf{c}}(s') = l_{\mathbf{c}}(s)$ and $v_{\mathbf{c}}(s') = v_{\mathbf{c}}(s)$. A path π in an abstract model is a sequence $s_0 \xrightarrow{b_1} s_1 \xrightarrow{b_2} s_2 \xrightarrow{b_3} \ldots$ of transitions such that for each $i \leq 1$, $b_i \in Act \cup \{\tau\}$ and $b_1 = \tau$ and for each two consecutive transitions at least one of them is a time transition. Next, $\pi[j..m]$ denotes the finite sequence $s_j \xrightarrow{\delta_{j+1},a_{j+1}} s_{j+1} \xrightarrow{\delta_{j+2},a_{j+2}} \ldots s_m$ with $m - j$ transitions and $m - j + 1$ states, and $D\pi[j..m]$ denotes the (cumulative) weight of $\pi[j..m]$ 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 $s \in S$ is denoted by $\Pi(s)$, and the set of all the paths starting at an initial state is denoted by $\Pi = \bigcup_{s^0 \in \widehat{\iota}} \Pi(s^0)$.

WECTLK. The WECTLK has been defined in [7] as the existential fragment of the weighted CTLK with integer cost constraints on *all* temporal modalities. We extend WECTLK logic by adding non-negative real cost constraints. In the syntax of WECTLK we assume the following: $p \in \mathcal{PV}$ is an atomic proposition, $\mathbf{c} \in Ag$, $\Gamma \subseteq Ag$, I is an interval in $\mathbb{R}$ of the form: $[a, \infty)$ and $[a, b)$, for $a, b \in \mathbb{N}$ and $a \neq b$. Moreover, hereafter, $\mathbf{right}(I)$ denotes the right end of the interval I. The WECTLK formulae are defined by the following grammar:

$$\varphi ::= \mathbf{true} \,|\, \mathbf{false} \,|\, p \,|\, \neg p \,|\, \varphi \vee \varphi \,|\, \varphi \wedge \varphi \,|\, \mathbf{EX}_I \varphi \,|\, \mathbf{E}(\varphi \mathbf{U}_I \varphi) \,|\, \mathbf{EG}_I \varphi \,|\, \overline{\mathrm{K}}_{\mathbf{c}} \varphi.$$

A WECTLK formula φ is *true* in an abstract model $\widehat{\mathcal{M}}$ (in symbols $\widehat{\mathcal{M}} \models \varphi$) iff $\widehat{\mathcal{M}}, s^0 \models \varphi$ for some $s^0 \in \widehat{\iota}$ (i.e., φ is true at some initial state of the abstract model $\widehat{\mathcal{M}}$). For every $s \in \widehat{S}$ the relation $\models$ is defined inductively as follows:

- $\widehat{\mathcal{M}}, s \models \mathbf{true}$, $\widehat{\mathcal{M}}, s \not\models \mathbf{false}$, $\widehat{\mathcal{M}}, s \models p$ iff $p \in \widehat{\mathcal{V}}(s)$, $\widehat{\mathcal{M}}, s \models \neg p$ iff $p \notin \widehat{\mathcal{V}}(s)$,
- $\widehat{\mathcal{M}}, s \models \alpha \wedge \beta$ iff $\widehat{\mathcal{M}}, s \models \alpha$ and $\widehat{\mathcal{M}}, s \models \beta$, $\widehat{\mathcal{M}}, s \models \alpha \vee \beta$ iff $\widehat{\mathcal{M}}, s \models \alpha$ or $\widehat{\mathcal{M}}, s \models \beta$
- $\widehat{\mathcal{M}}, s \models \mathbf{EX}_I \alpha$ iff $(\exists \pi \in \Pi(s))(D\pi[0..1] \in I$ and $\widehat{\mathcal{M}}, \pi(1) \models \alpha)$,
- $\widehat{\mathcal{M}}, s \models \mathbf{EG}_I \alpha$ iff $(\exists \pi \in \Pi(s))(\forall i \geq 0)(D\pi[0..i] \in I$ implies $\widehat{\mathcal{M}}, \pi(i) \models \beta)$,
- $\widehat{\mathcal{M}}, s \models \mathbf{E}(\alpha \mathbf{U}_I \beta)$ iff $(\exists \pi \in \Pi(s))(\exists i \geq 0)(D\pi[0..i] \in I$ and $\widehat{\mathcal{M}}, \pi(i) \models \beta$ and $(\forall j < i)\widehat{\mathcal{M}}, \pi(j) \models \alpha)$,
- $\widehat{\mathcal{M}}, s \models \overline{\mathrm{K}}_{\mathbf{c}} \alpha$ iff $(\exists \pi \in \Pi)\ (\exists i \geq 0)(s \sim_{\mathbf{c}} \pi(i)$ and $\widehat{\mathcal{M}}, \pi(i) \models \alpha)$.

3 SMT-based Bounded Model Checking

In this section, we present an outline of the bounded semantics for WECTLK and define an SMT-based BMC method for WECTLK, which is based on the BMC encoding presented in [7]. As usual, we start by defining k-paths and $(k,l) - loops$. Next, we define a bounded semantics, which is used for the translation to SMT.

Bounded semantics. Let $\widehat{\mathcal{M}}$ be an abstract model, and $k \in \mathbb{N}$ a bound. A *k-path* π_k is a finite sequence $s_0 \xrightarrow{b_1} s_1 \xrightarrow{b_2} \ldots \xrightarrow{b_k} s_k$ of transitions such that for each $1 \leq i \leq k$, $b_i \in Act \cup \{\tau\}$ and $b_1 = \tau$ and for each two consecutive transitions at least one is a time transition. A k-path π_k is a *loop* if $l < k$ and $\pi(k) = \pi(l)$. Note that if a k-path π_k is a loop, then it represents the infinite path of the form uv^{ω}, where $u = (s_0 \xrightarrow{b_1} s_1 \xrightarrow{b_2} \ldots \xrightarrow{b_l} s_l)$ and $v = (s_{l+1} \xrightarrow{b_{l+2}} \ldots \xrightarrow{b_k} s_k)$. $\Pi_k(s)$ denotes the set of all the k-paths of $\widehat{\mathcal{M}}$ that start at s, and $\Pi_k = \bigcup_{s^0 \in \widehat{\iota}} \Pi_k(s^0)$.

The bounded satisfiability relation $\models_k$ which indicates k-truth of a WECTLK formula in the abstract model $\widehat{\mathcal{M}}$ at some state s of $\widehat{\mathcal{M}}$ is also defined in [7]. A WECTLK formula φ is *k-true* in the abstract model $\widehat{\mathcal{M}}$ (in symbols $\widehat{\mathcal{M}} \models_k \varphi$) iff φ is k-true at some initial state of the abstract model $\widehat{\mathcal{M}}$.

The *model checking problem* asks whether $\widehat{\mathcal{M}} \models \varphi$, but the *bounded model checking problem* asks whether there exists $k \in \mathbb{N}$ such that $\widehat{\mathcal{M}} \models_k \varphi$. The following theorem states that for a given abstract model and a WECTLK formula there exists a bound k such that the model checking problem ($\widehat{\mathcal{M}} \models \varphi$) can be reduced to the bounded model checking problem ($\widehat{\mathcal{M}} \models_k \varphi$).

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

Proof. The theorem can be proved by induction on the length of the formula φ (for details one can see [7]). ◀

Translation to SMT. Let $\widehat{\mathcal{M}}$ be an abstract model, φ a WECTLK formula, and $k \geq 0$ a bound. The presented SMT encoding of the BMC problem for WECTLK and for TRWIS is based on the SAT encoding of the same problem [10, 9], and it relies on defining the quantifier-free first-order formula: $[\widehat{\mathcal{M}}, \varphi]_k := [\widehat{\mathcal{M}}^{\varphi,\iota}]_k \wedge [\varphi]_{\widehat{\mathcal{M}},k}$ that is satisfiable if and only if $\widehat{\mathcal{M}} \models_k \varphi$ holds.

Let $\mathbf{c} \in Ag \cup \{\mathcal{E}\}$. The definition of the formula $[\widehat{\mathcal{M}}, \varphi]_k$ assumes that

- each global state $s \in \widehat{S}$ is represented by a valuation of a *symbolic state* $\overline{\mathbf{w}} = ((w_1, v_1), \ldots, (w_n, v_n), (w_{\mathcal{E}}, v_{\mathcal{E}}))$ that consists of *symbolic local states* and each symbolic local state $w_{\mathbf{c}}$ is a pair $(w_{\mathbf{c}}, v_{\mathbf{c}})$ of individual variables ranging over the natural numbers, in which the first element represents a local state of the agent $\mathbf{c}$, and the second represents a clock valuation;

- each joint action $a \in Act$ is represented by a valuation of a *symbolic action* $\overline{a} = (a_1, \ldots, a_n, a_{\mathcal{E}})$ that consists of *symbolic local actions* and each symbolic local action $a_{\mathbf{c}}$ is an individual variable ranging over the natural numbers;
- each sequence of weights associated with the joint action is represented by a valuation of a *symbolic weights* $\overline{d} = (d_1, \ldots, d_{n+1})$ that consists of *symbolic local weights* and each symbolic local weight $d_{\mathbf{c}}$ is an individual variable ranging over the natural numbers.

The formula $[\widehat{\mathcal{M}}^{\varphi,\widehat{\iota}}]_k$ encodes a rooted tree of $k-$paths of the abstract model $\widehat{\mathcal{M}}$. The number of branches of the tree depends on the value of $f_k : \text{WECTLK} \to \mathbb{N}$ which is an auxiliary function defined in [7]. The formula $[\widehat{\mathcal{M}}^{\varphi,\widehat{\iota}}]_k$ is defined over $(k+1) \cdot f_k(\varphi)$ different symbolic states, $k \cdot f_k(\varphi)$ different symbolic actions, and $k \cdot f_k(\varphi)$ different symbolic weights. Moreover, it uses the following auxiliary quantifier-free first-order formulae:

- $I_s(\overline{\mathbf{w}})$ – it encodes the state s of the abstract model $\widehat{\mathcal{M}}$;
- $H_{\mathbf{c}}(w_{\mathbf{c}}, w'_{\mathbf{c}})$ – it encodes equality of two local states, such that $w_{\mathbf{c}} = w'_{\mathbf{c}}$ for $\mathbf{c} \in Ag \cup \mathcal{E}$;
- $\mathcal{T}_{\mathbf{c}}(w_{\mathbf{c}}, ((\overline{a}, \overline{d}), \overline{\delta}), w'_{\mathbf{c}})$ – it encodes the local evolution function of agent $\mathbf{c}$;
- $\mathcal{A}(\overline{a})$ – it encodes that each symbolic local action $a_{\mathbf{c}}$ of $\overline{a}$ has to be executed by each agent in which it appears;
- $\mathcal{T}(\overline{\mathbf{w}}, ((\overline{a}, \overline{d}), \overline{\delta}), \overline{\mathbf{w}}') := \mathcal{A}(\overline{a}) \wedge \bigwedge_{\mathbf{c} \in Ag \cup \{\mathcal{E}\}} \mathcal{T}_{\mathbf{c}}(w_{\mathbf{c}}, ((\overline{a}, \overline{d}), \overline{\delta}), w'_{\mathbf{c}})$;
- Let $\boldsymbol{\pi}_j$ denote the j-th *symbolic k-path*, i.e. the sequence of symbolic transitions: $\overline{\mathbf{w}}_{0,j} \xrightarrow{(\overline{a}_{1,j}, \overline{d}_{1,j}), \delta_{1,j}} \overline{\mathbf{w}}_{1,j} \xrightarrow{(\overline{a}_{2,j}, \overline{d}_{2,j}), \delta_{2,j}} \ldots \xrightarrow{(\overline{a}_{k,j}, \overline{d}_{k,j}), \delta_{k,j}} \overline{\mathbf{w}}_{k,j}$. Then, $\mathcal{D}^I_{a,b;c,d}(\boldsymbol{\pi}_n)$ for $a \leq b$ and $c \leq d$ is a formula that:
  - for $a < b$ and $c < d$ encodes that the weight represented by the sequences $\overline{d}_{a+1,n}, \ldots, \overline{d}_{b,n}$ and $\overline{d}_{c+1,n}, \ldots, \overline{d}_{d,n}$ belongs to the interval I,
  - for $a = b$ and $c < d$ encodes that the weight represented by the sequence $\overline{d}_{c+1,n}, \ldots, \overline{d}_{d,n}$ belongs to the interval I,
  - for $a < b$ and $c = d$ encodes that the weight represented by the sequence $\overline{d}_{a+1,n}, \ldots, \overline{d}_{b,n}$ belongs to the interval I,
  - for $a = b$ and $c = d$, the formula $\mathcal{D}^I_{a,b;c,d}(\boldsymbol{\pi}_n)$ is true iff $0 \in I$.

Thus, given the above, one can define the formula $[\widehat{\mathcal{M}}^{\varphi,\widehat{\iota}}]_k$ as follows:

$$[\widehat{\mathcal{M}}^{\varphi,\widehat{\iota}}]_k := \bigvee_{s \in \widehat{\iota}} I_s(\overline{\mathbf{w}}_{0,0}) \wedge \bigvee_{j=1}^{f_k(\varphi)} \overline{\mathbf{w}}_{0,0} = \overline{\mathbf{w}}_{0,j} \wedge \bigwedge_{j=1}^{f_k(\varphi)} \bigwedge_{i=0}^{k-1} \mathcal{T}(\overline{\mathbf{w}}_{i,j}, ((\overline{a}_{i,j}, \overline{d}_{i,j}), \overline{\delta}_{i,j}), \overline{\mathbf{w}}_{i+1,j})$$

where $\overline{\mathbf{w}}_{i,j}$, $\overline{a}_{i,j}$, and $\overline{d}_{i,j}$ are, respectively, symbolic states, symbolic actions, and symbolic weights for $0 \leq i \leq k$ and $1 \leq j \leq f_k(\varphi)$. Hereafter, by $\boldsymbol{\pi}_j$ we denote the j-th symbolic k-path of the above unfolding, i.e., the sequence of transitions: $\overline{\mathbf{w}}_{0,j} \xrightarrow{(\overline{a}_{1,j}, \overline{d}_{1,j}), \overline{\delta}_{1,j}} \overline{\mathbf{w}}_{1,j} \xrightarrow{(\overline{a}_{2,j}, \overline{d}_{2,j}), \overline{\delta}_{2,j}} \ldots \xrightarrow{(\overline{a}_{k,j}, \overline{d}_{k,j}), \overline{\delta}_{k,j}} \overline{\mathbf{w}}_{k,j}$.

The formula $[\varphi]_{\widehat{\mathcal{M}},k}$ encodes the bounded semantics of a WECTLK formula φ, and it is defined on the same sets of individual variables as the formula $[\widehat{\mathcal{M}}^{\varphi,\widehat{\iota}}]_k$. Moreover, it uses the auxiliary quantifier-free first-order formulae defined in [7].

Furthermore, following [7], our formula $[\varphi]_{\widehat{\mathcal{M}},k}$ uses the following auxiliary functions g_l, g_r, g_μ, $h_{\mathbf{U}}$, $h_{\mathbf{G}}$ that were introduced in [10], and which allow to divide the set $A \subseteq F_k(\varphi) = \{j \in \mathbb{N} \mid 1 \leq j \leq f_k(\varphi)\}$ into subsets needed for translating the subformulae of φ. Let $0 \leq n \leq f_k(\varphi)$, $m \leqslant k$, and $n' = min(A)$. The rest of translation is defined in the same way as in [7].

- $[\mathbf{true}]_k^{[m,n,A]} := \mathbf{true}$, $[\mathbf{false}]_k^{[m,n,A]} := \mathbf{false}$,
- $[p]_k^{[m,n,A]} := p(\overline{\mathbf{w}}_{m,n})$, $[\neg p]_k^{[m,n,A]} := \neg p(\overline{\mathbf{w}}_{m,n})$,
- $[\alpha \wedge \beta]_k^{[m,n,A]} := [\alpha]_k^{[m,n,g_l(A,f_k(\alpha))]} \wedge [\beta]_k^{[m,n,g_r(A,f_k(\beta))]}$,

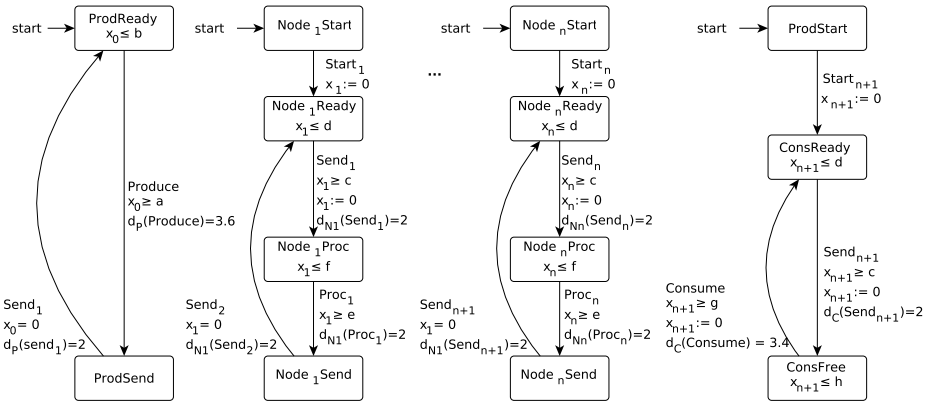

Figure 1 The TWGPP system.

- $[\alpha \vee \beta]_k^{[m,n,A]} := [\alpha]_k^{[m,n,g_l(A,f_k(\alpha))]} \vee [\beta]_k^{[m,n,g_l(A,f_k(\beta))]}$,
- $[\mathbf{EX}_I\alpha]_k^{[m,n,A]} := \overline{\mathbf{w}}_{m,n} = \overline{\mathbf{w}}_{0,n'} \wedge (\overline{d}_{1,n'} \in I) \wedge [\alpha]_k^{[1,n',g_\mu(A)]}$, if $k > 0$; **false**, otherwise,
- $[\mathbf{E}(\alpha \mathbf{U}_I \beta)]_k^{[m,n,A]} := \overline{\mathbf{w}}_{m,n} = \overline{\mathbf{w}}_{0,n'} \wedge \bigvee_{i=0}^{k}([\beta]_k^{[i,n',h_{\mathbf{U}}(A,k,f_k(\beta))(j)]} \wedge (\sum_{j=1}^{i} \overline{d}_{j,n} \in I \wedge \bigwedge_{j=0}^{i-1}[\alpha]_k^{[j,n',h_{\mathbf{U}}(A,k,f_k(\beta))]})$,
- $[\mathbf{E}(\mathbf{G}_I\alpha)]_k^{[m,n,A]} := \overline{\mathbf{w}}_{m,n} = \overline{\mathbf{w}}_{0,n'} \wedge \big((\sum_{j=1}^{k} \overline{d}_{j,n} \geq \mathbf{right}(I) \wedge \bigwedge_{i=0}^{k}(\sum_{j=1}^{i} \overline{d}_{j,n} \notin I \vee [\alpha]_k^{[i,n',h_{\mathbf{G}}(A,k)(j)]})) \vee (\sum_{j=1}^{k} \overline{d}_{j,n} < \mathbf{right}(I) \wedge \bigwedge_{i=0}^{k}(\sum_{j=1}^{i} \overline{d}_{j,n} \notin I \vee [\alpha]_k^{[i,n',h_{\mathbf{G}}(A,k)(j)]}) \wedge \bigvee_{l=0}^{k-1}(\overline{\mathbf{w}}_{k,n'} = \overline{\mathbf{w}}_{l,n'} \wedge \bigwedge_{i=l}^{k-1}(\neg \mathcal{D}^I_{0,k;l,i+1}(\boldsymbol{\pi}_{n'}) \vee [\alpha]_k^{[i,n',h_{\mathbf{G}}(A,k)(j)]}))))\big)$,
- $[\overline{\mathrm{K}}_{\mathbf{c}}\alpha]_k^{[m,n,A]} := (\bigvee_{s \in \widehat{\iota}} I_s(\overline{\mathbf{w}}_{0,n'})) \wedge \bigvee_{j=0}^{k}([\alpha]_k^{[j,n',g_\mu(A)]} \wedge H_{\mathbf{c}}(\overline{\mathbf{w}}_{m,n}, \overline{\mathbf{w}}_{j,n'}))$,

The theorem below states the correctness and the completeness of the presented translation. It can be proved in a standard way, using induction on the complexity of the given WECTLK formula.

▶ **Theorem 2.** *Let $\widehat{\mathcal{M}}$ be an abstract model, and φ a WECTLK formula. For every $k \in \mathbb{N}$, $\widehat{\mathcal{M}} \models_k \varphi$ if, and only if, the quantifier-free first-order formula $[\widehat{\mathcal{M}}, \varphi]_k$ is satisfiable.*

4 Experimental Results

In this section, we experimentally evaluate the performance of our SMT-based BMC encoding for WECTLK over the TRWIS semantics.

The benchmark, we consider is the *timed weighted generic pipeline paradigm* (TWGPP) TRWIS abstract model [9]. The abstract model of TWGPP involves $n + 2$ agents: Producer producing data within the certain time interval ($[a, b]$) or being inactive, Consumer receiving data within the certain time interval ($[c, d]$) or being inactive within the certain time interval ($[g, h]$), a chain of n intermediate Nodes which can be ready for receiving data within the certain time interval ($[c, d]$), processing data within the certain time interval ($[e, f]$) or sending data. The weights are used to adjust the cost properties of Producer, Consumer, and of the intermediate Nodes.

Each agent of the scenario can be modelled by considering its local states, the local actions, the local protocol, the local evolution function, the local weight function, the local clocks, the clock constraints, the invariants, and the local valuation function. Fig. 1 shows

the local states, the possible actions, and the protocol, the clock constraints, invariants and weights for each agent. Null actions are omitted in the figure.

Given Fig. 1, the local evolution functions of TWGPP are straightforward to infer. Moreover, we assume the following set of propositional variables: $\mathcal{PV} = \{ProdReady, ProdSend, ConsReady, ConsFree\}$ with the following definitions of local valuation functions: $\widehat{\mathcal{V}}_P(ProdReady\text{-}0) = \{ProdReady\}$, $\widehat{\mathcal{V}}_P(ProdSend\text{-}1) = \{ProdSend\}$; $\widehat{\mathcal{V}}_C(ConsReady\text{-}0) = \{ConsReady\}$,
$\widehat{\mathcal{V}}_C(ConsFree\text{-}1) = \{ConsFree\}$.

Let $Act = Act_P \times \prod_{i=1}^{n} Act_{N_i} \times Act_C$, with $Act_P = \{Produce, Send_1\}$, $Act_C = \{Start_{n+1}, Consume, Send_{n+1}\}$, and $Act_{N_i} = \{Start_i, Send_i, Send_{i+1}, Proc_i\}$ defines the set of joint actions for the scenario. For $\widetilde{a} \in Act$ let $act_P(\widetilde{a})$ denotes an action of Producer, $act_C(\widetilde{a})$ denotes an action of Consumer, and $act_{N_i}(\widetilde{a})$ denotes an action of Node i. We assume the following local evolution functions: $t_P(ProdReady, x_0 \geq a, \emptyset, \widetilde{a}) = ProdSend$, if $act_P(\widetilde{a}) = Produce$; $t_P(ProdSend, true, \{x_0\}, \widetilde{a}) = ProdReady$, if $act_P(\widetilde{a}) = Send_1$ and $act_{N_i}(\widetilde{a}) = Send_1$; $t_C(ConsStart, true, \{x_{n+1}\}, \widetilde{a}) = ConsReady$, if $act_C(\widetilde{a}) = Start_{n+1}$; $t_C(ConsReady, x_{n+1} \geq c, \{x_{n+1}\}, \widetilde{a}) = ConsFree$, if $act_C(\widetilde{a}) = Send_{n+1}$ and $act_{N_n}(\widetilde{a}) = Send_{n+1}$; $t_C(ConsFree, x_{n+1} \geq g, \{x_{n+1}\}, \widetilde{a}) = ConsReady$, if $act_C(\widetilde{a}) = Consume$.

Finally, we assume the following two local weight functions for each agent:

- $d_P(Produce) = 4$, $d_P(send_1) = 2$, $d_C(Consume) = 4$, $d_C(send_{n+1}) = 2$, $d_{N_i}(send_i) = d_{Ni}(send_{i+1}) = d_{N_i}(Proc_i) = 2$,
- $d_P(Produce) = 4000$, $d_P(send_1) = 2000$, $d_C(Consume) = 4000$, $d_C(send_{n+1}) = 2000$, $d_{N_i}(send_i) = d_{Ni}(send_{i+1}) = d_{N_i}(Proc_i) = 2000$.

The set of all the global states $\widehat{S}$ for the scenario is defined as the product $(L_P \times \mathbb{N}) \times \prod_{i=1}^{n}(L_i \times \mathbb{N}) \times (L_C \times \mathbb{N})$. The set of the initial states is defined as $\widehat{\iota} = \{s^0\}$, where $s^0 = ((ProdReady\text{-}0, 0), (Node_1Ready\text{-}0, 0), \ldots, (Node_nReady\text{-}0, 0), (ConsReady\text{-}0, 0))$.

The system is scaled according to the number of its Nodes (agents), i.e., the problem parameter n is the number of Nodes. For any natural number $n \geq 0$, let $D(n) = \{1, 3, \ldots, n-1, n+1\}$ for an even n, and $D(n) = \{2, 4, \ldots, n-1, n+1\}$ for an odd n. Moreover, let $r(j) = d_P(Produce) + 2 \cdot \sum_{i=1}^{j} d_{N_i}(Send_i) + \sum_{i=1}^{j-1} \cdot d_{N_i}(proc_i)$. Then, we define $Right$ as follows: $Right = \sum_{j \in D(n)} r(j)$.

We consider the following formulae as specifications:

- $\varphi_1 = \overline{\mathrm{K}}_P(\mathbf{EF}_{[0,Right)}(ConsFree \wedge \mathbf{EG}(ProdSend \vee ConsFree)))$ - *it states that it is not true that Producer knows that there exists a path on which Consumer receives a data and the cost of receiving the data is less than Right and from that point there exists a path on which always either the Producer has sent a data or the Consumer has received a data.*
- $\varphi_2 = \overline{\mathrm{K}}_P(\mathbf{EF}_{[0,Right)}(ConsFree \wedge \overline{\mathrm{K}}_C\overline{\mathrm{K}}_P(\mathbf{EG}(ProdSend \vee ConsFree))))$- *it states that it is not true that Producer knows that there exists a path on which Consumer receives a data and the cost of receiving the data is less than Right and at that point it is not true that Consumer knows that it is not true that Producer knows that there exists a path on which always either the Producer has sent a data or Consumer has received a data.*

The number of the considered k-paths is equal to 3 for φ_1, and 5 for φ_2, respectively. The length of the witness is $2 \cdot n + 4$ if $n \in \{1, 2\}$ and, $2 \cdot n + 2$ if $n > 2$ for the formula φ_1, $2 \cdot n + 2$ for the formula φ_2, respectively.

Performance evaluation. We performed our experimental results on a computer equipped with I7-3770 processor, 32 GB of RAM, and the operating system Arch Linux with the kernel 3.19.2. The CPU time limit was set to 3600 seconds. Our SMT-based BMC algorithm was

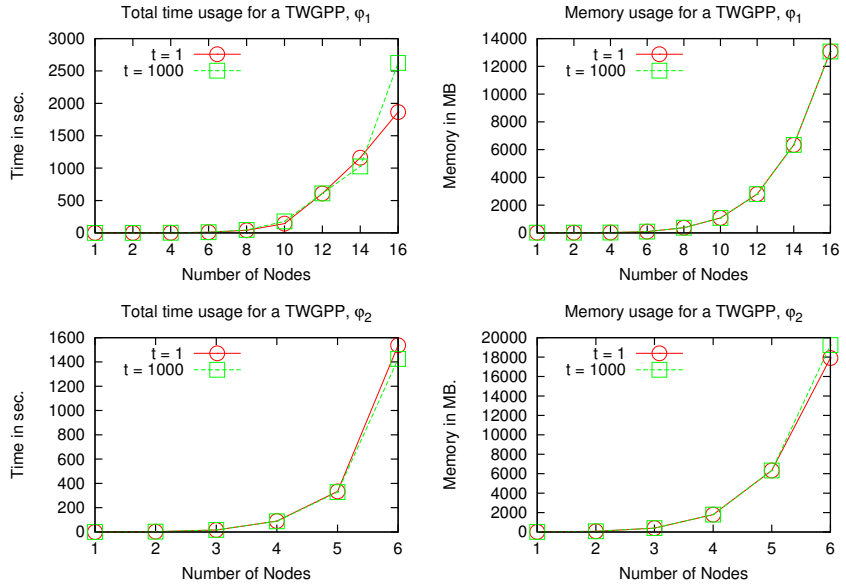

Figure 2 Formulae φ_1 and φ_2: Scaling up both the number of nodes and weights.

implemented as a standalone program written in the programming language `C++`. We used the state of the art SMT-solver Z3 [5].

For both properties φ_1 and φ_2 we scaled up both the number of nodes and the weights parameters. The results are summarised on charts in Fig. 2. One can observe that our SMT-based BMC is not sensitive to scaling up the weights, but it is sensitive to scaling up the size of benchmark.

For both the formulae, we obtained encouraging results. Namely, for φ_1 and for TWGPP with 16 nodes and the basic weights (`bw` for short) our method uses 13074.2 MB and 1864.4 seconds (13072.0 MB and 2624.5 seconds for `bw` multiplied by 1000); Next, for φ_2 and TWGPP with 6 nodes our method uses 17904.5 MB and 1536.9 second (19240.9 MB and 1424.4 seconds for `bw` multiplied by 1000).

5 Conclusions

We have proposed SMT-based BMC verification method for model checking WECTLK properties interpreted over the timed real-weighted interpreted systems. The preliminary experimental results show that the method is worth interest. In the future we are going to provide a comparison of our new method with the SAT- and BDD-based BMC methods. The module will be added to the model checker VerICS ([3]). All the benchmarks together with an instruction how to reproduce our results can be found at the webpage `http://www.ajd.czest.pl/~imi/agnieszkazbrzezny/modelchecking/`.

References

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

2 P. Gammie and R. van der Meyden. MCK: Model checking the logic of knowledge. In *Proceedings of CAV'04*, volume 3114 of *LNCS*, pages 479–483. Springer-Verlag, 2004.

3 M. Kacprzak, W. Nabialek, A. Niewiadomski, W. Penczek, A. Pólrola, M. Szreter, B. Woźna, and A. Zbrzezny. VerICS 2007 – a model checker for knowledge and real-time. *Fundamenta Informaticae*, 85(1-4):313–328, 2008.

4 A. Lomuscio, H. Qu, and F. Raimondi. MCMAS: A model checker for the verification of multi-agent systems. In *Proceedings of CAV'09*, volume 5643 of *LNCS*, pages 682–688. Springer-Verlag, 2009.

5 L. De Moura and N. Bjørner. Z3: an efficient SMT solver. In *Proceedings of TACAS'08*, volume 4963 of *LNCS*, pages 337–340. Springer-Verlag, 2008.

6 M. Wooldridge. *An introduction to multi-agent systems – Second Edition.* John Wiley & Sons, 2009.

7 B. Woźna-Szcześniak. SAT-based bounded model checking for weighted deontic interpreted systems. In *Proceedings of EPIA'13*, volume 8154 of *LNAI*, pages 444–455. Springer-Verlag, 2013.

8 B. Woźna-Szcześniak. Checking EMTLK properties of timed interpreted systems via bounded model checking. In *Proceedings of AAMAS'14*, pages 1477–1478. IFAAMAS/ACM, 2014.

9 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'13*, volume 8291 of *LNAI*, pages 355–371. Springer-Verlag, 2013.

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