ECAI 2023
K. Gal et al. (Eds.)
© 2023 The Authors.
This article is published online with Open Access by IOS Press and distributed under the terms of the Creative Commons Attribution Non-Commercial License 4.0 (CC BY-NC 4.0).
doi:10.3233/FAIA230394

SMT-Based Satisfiability Checking of Strategic Metric Temporal Logic

Magdalena Kacprzak[a], **Artur Niewiadomski**[b;*], **Wojciech Penczek**[c] **and Andrzej Zbrzezny**[d]

[a]Faculty of Computer Science, Bialystok University of Technology, Bialystok, Poland
[b]Siedlce University, Faculty of Exact and Natural Sciences, Siedlce, Poland
[c]Institute of Computer Science, Polish Academy of Sciences, Warsaw, Poland
[d]Faculty of Science and Technology, Jan Dlugosz University in Czestochowa, Czestochowa, Poland
ORCiD ID: Magdalena Kacprzak https://orcid.org/0000-0002-9464-7686,
Artur Niewiadomski https://orcid.org/0000-0002-9652-5092,
Wojciech Penczek https://orcid.org/0000-0001-6477-4863,
Andrzej Zbrzezny https://orcid.org/0000-0003-2771-9683

Abstract. The paper presents a novel SMT-based method for testing the satisfiability of formulae that express strategic properties of timed multi-agent systems represented by networks of timed automata. Strategic Metric Temporal Logic (SMTL) is introduced, which extends Metric Temporal Logic (MTL) with strategy operators. SMTL is interpreted over maximal continuous time runs of timed automata. We define a procedure that synthesises a model for a given SMTL formula if such a model exists. The method exploits Satisfiability Modulo Theories (SMT) techniques and Parametric Bounded Model Checking algorithms. The presented approach enables bounded satisfiability checking, where the model is partially given and needs to be completed in line with the given specification. Our method has been implemented, and its application is demonstrated through an example of the well-known dining philosophers problem extended with clocks and strategies. The experimental results are quite encouraging.

1 Introduction

Artificial Intelligence (AI) has made significant strides in driving cars, playing games, composing music, and creating art [16]. However, will software engineering also become fully automated? While AI can certainly provide support [33], building a correct model for a system is still a difficult and complex task that requires human expertise. Although fully automatic model building may be computationally expensive or impossible, a practical compromise is to combine the engineer's work with the capabilities of a computer. For example, when designing a safe road intersection, the vehicles and their possible activities can be modelled, and the challenge is to synthesise a traffic controller or traffic lights automatically. This paper focuses on the problem of satisfiability (SAT) checking for a strategic logic. Our solution involves synthesising a model for a given formula, but the model does not need to be built from scratch. By leveraging existing data on selected attributes of the system, it is only necessary to complete a (finite) candidate model to satisfy the desired property.

The primary objective of this paper is to introduce a new method for implementing bounded satisfiability (BSAT) for Metric Temporal Logic (MTL) [28, 7] and its extension with strategic operators, called Strategic Metric Temporal Logic (SMTL). Our BSAT procedure for SMTL combines Bounded Model Checking (BMC) with Satisfiability Modulo Theories (SMT) solvers. The basic idea behind our method is to synthesise the model as the product of agents represented by a "fully" parametric Network of Timed Automata (NTA), given an SMTL formula expressing the desired property. In addition to standard parameters used in guards and invariants, we can also specify constraints on parameters representing the number of agents and the numbers of their locations, transitions, and actions. Optionally, if we have additional knowledge about the system's structure, we can replace some parameters with fixed values. Next, we encode the SMTL formula and the runs of the parametric timed automata network, unfolded to a given depth k, as an SMT problem instance, which is then checked for satisfiability by an SMT-solver. If the answer is SAT, we obtain all the parameter values from a model returned by the SMT-solver. Otherwise, we increase the unfolding depth to $k + 1$ and possibly adjust the values of some parameters such as the number of agents, locations, and transitions.

We also present some preliminary experimental results generated by our prototype tool SMT4SMTL, applied to a parametric timed version of the Dining Philosophers benchmark. Among these results, we demonstrate an automatic synthesis of a lackey agent. Other potential applications of SMTL include the specification of properties related to the existence of strategies for reaching specific states within a given time period (time-bounded (un)reachability, (non-)response properties). Our tool helps to build or enhance systems to meet these properties, with applications in timed games, earliest deadline first (EDF) schedulability, and unmanned vehicle protocols.

1.1 Contribution and Outline

The main contributions of the paper are as follows:
- A definition of Strategic Metric Temporal Logic (SMTL);
- A method for solving BSAT for (S)MTL;
- A parametric encoding of networks of timed automata;

* Emails: artur.niewiadomski@uph.edu.pl (corresponding author), m.kacprzak@pb.edu.pl, penczek@ipipan.waw.pl, a.zbrzezny@ujd.edu.pl

• Parametric Bounded Model Checking (PBMC) for (S)MTL;
• The tool SMT4SMTL using PBMC and SMT-solvers.
The rest of the paper is organised as follows. The related work is discussed in Section 2. Next in Section 3, timed automata and their models are defined. The logic SMTL is introduced in Section 4. In Section 5 the SMT encoding of the SAT problem for SMTL is discussed. Section 6 presents our implementation and experimental results. Finally the summary is provided in Section 7.

2 Related Work

While there are only a few implemented approaches to the satisfiability problem for (timed) strategic logics (discussed below), there have been a lot of approaches to the model checking problem of such logics [24, 25, 29, 21, 18, 38, 20, 26]. The reason of this unbalanced situation is twofold. Firstly, the model checking problem (MC) is typically of lower complexity than the satisfiability problem (SAT) for the same logic. For example: for ATL the complexity of MC is PTIME-complete [6], while the complexity of SAT is EXPTIME-complete [44], for ATL*, the complexity of MC is PSPACE-complete [19], while the complexity of SAT is 2EXPTIME-complete [42, 14], for SL (strategy logic), the complexity of MC is non-elementarily decidable [31], while SAT is undecidable [31], and for TCTL the complexity of MC is PSPACE-complete [1, 39], while SAT is undecidable [1]. However, for MTL interpreted over real-time, without any restrictions, both MC and SAT are undecidable [12, 7]. Secondly, model checking is considered to be more practice oriented as it is applied directly to the system verification while satisfiability checking is exploited for the system synthesis, which is frequently less feasible.

In [27] a lazy SMT-solver for Boolean monotonic theories (SMMT) was applied to model synthesis for Computation Tree Logic (CTL [9, 49]). This method was then extended in [22, 34] to deciding the satisfiability of Alternating-time Temporal Logic (ATL [5, 6]), and in [23] to deciding Simple-Goal Strategy Logic (SL[SG] [10]). Another approach to checking satisfiability of ATL* was developed in [14], using a tableau-based decision procedure.

In this paper, we present a different approach to the satisfiability checking and synthesis. First of all, we consider the logic MTL over real-time, and its extension with strategic operators (SMTL), regardless of their undecidability in general. Secondly, we combine the SMT-based parametric model checking with bounded model checking, both approaches known for their efficiency. Thirdly, while synthesising a model based on NTA for SMTL we do not need to start from scratch, but we can start with a model partially built by a (human) designer. Our method builds upon the MTL formula translation and encoding of the transition relation of [48], but we introduce parameters that require additional constraints to ensure the meaningfulness of the resulting NTA.

SMTL resembles STCTL [8], where subformulae of TCTL [1] are replaced with subformulae of MTL, so SMTL and STCTL are of incomparable expressiveness like in case of TCTL and MTL.

With the exception of our work, the only tool for deciding the satisfiability of variants of real-time MTL is reported in [11], where a translation of MITL and MTL$_{(0,\infty)}$ into CLTLoc and further a reduction to SMT for QF-EUF and QF-LRA is described. Similarly to our approach the authors consider BSAT and apply SMT-solvers, but they do not consider a strategic extension of MTL and do not deal with synthesis of NTA satisfying SMTL specifications. The paper [11] is focused on finding clock values and yes/no answer for the SAT problem, while our paper applies the parametric BMC to construct a NTA, whose paths satisfy the tested formula.

In order to regain decidability of MTL, certain semantic and syntactic restrictions are introduced [37]. Semantic restrictions include adopting an integer-time model [7, 17, 15] or a bounded-variation dense-time model [45] for which SAT is decidable. Syntactic restrictions concern: punctuality or non-singularity [4], boundedness and safety [13, 35, 36]. Then, the SAT problem becomes decidable and is EXPSPACE-complete.

3 Timed Automata and Their Models

In this section we define Timed Automata [2] with asynchronous, strongly monotonic semantics and continuous time.

3.1 Timed Automata

Hereafter, $\mathbb{N} = \{0, 1, 2, \dots\}$ denotes the set of natural numbers, $\mathbb{R}_+$ denotes the set of positive real numbers, and $\mathbb{R}_\geq$ denotes the set of non-negative real numbers. Let $\mathcal{X} = \{x_1, \dots, x_n\}$ be a finite set of variables, called clocks. A *clock valuation* is a function $v : \mathcal{X} \to \mathbb{R}_\geq$, assigning to each clock x a non-negative value $v(x)$. The set of all the valuations is denoted by $(\mathbb{R}_\geq)^{\mathcal{X}}$. For a subset X of $\mathcal{X}$ by $v[X := 0]$ we mean the valuation v' such that $\forall x \in X$, $v'(x) = 0$ and $\forall x \in \mathcal{X} \backslash X$, $v'(x) = v(x)$. For $\delta \in \mathbb{R}_+$, $v + \delta$ denotes the valuation v'' such that $\forall x \in \mathcal{X}$, $v''(x) = v(x) + \delta$.

The set $\mathcal{C}(\mathcal{X})$ of *clock constraints* over the set of clocks $\mathcal{X}$ is defined inductively as follows: $\mathfrak{cc} := x \sim c \mid \mathfrak{cc} \wedge \mathfrak{cc}$, where $x \in \mathcal{X}$, $\sim \in \{\leq, <, =, >, \geq\}$, and $c \in \mathbb{R}_\geq$.
A clock valuation v *satisfies* the clock constraint $\mathfrak{cc} \in \mathcal{C}(\mathcal{X})$:

$$\begin{array}{lll} v \models x \sim c & \text{iff} & v(x) \sim c, \\ v \models \mathfrak{cc} \wedge \mathfrak{cc}' & \text{iff} & v \models \mathfrak{cc} \text{ and } v \models \mathfrak{cc}'. \end{array}$$

For each $\mathfrak{cc} \in \mathcal{C}(\mathcal{X})$ by $v(\mathfrak{cc})$ we denote the set of all the clock valuations satisfying $\mathfrak{cc}$, i.e., $v(\mathfrak{cc}) = \{v \in (\mathbb{R}_\geq)^{\mathcal{X}} \mid v \models \mathfrak{cc}\}$. Similarly, $v(\mathcal{C}(\mathcal{X})) = \{v(\mathfrak{cc}) \subseteq (\mathbb{R}_\geq)^{\mathcal{X}} \mid \mathfrak{cc} \in \mathcal{C}(\mathcal{X})\}$. Now, we are ready to define Timed Automata.

Definition 1. *A timed automaton (*TA *for short) is a 8-tuple*

$$\mathcal{A} = (Act, Loc, \ell^0, \mathcal{X}, T, Inv, \mathcal{AP}, V), \textit{ where}$$

- *Act is a finite set of actions,*
- *Loc is a finite set of locations,*
- *$\ell^0 \in Loc$ is an initial location,*
- *$\mathcal{X}$ is a finite set of clocks,*
- *$T \subseteq Loc \times Act \times \mathcal{C}(\mathcal{X}) \times 2^{\mathcal{X}} \times Loc$ is a transition relation,*
- *$Inv : Loc \to \mathcal{C}(\mathcal{X})$ is a state invariant function,*
- *$\mathcal{AP}$ is a finite set of atomic propositions, and*
- *$V\colon Loc \to 2^{\mathcal{AP}}$ is a valuation function assigning to each location a set of atomic propositions true in this location.*

Each $t \in T$, denoted by $\ell \xrightarrow{\alpha, \mathfrak{cc}, X} \ell'$, represents a transition from ℓ to ℓ' on the action α. $X \subseteq \mathcal{X}$ is the set of the clocks to be reset upon this transition, and $\mathfrak{cc} \in \mathcal{C}(\mathcal{X})$ is the enabling condition for t.

Given a transition t, we write $source(t)$, $target(t)$, $guard(t)$ for ℓ, ℓ', and $\mathfrak{cc}$, respectively. The clocks of a timed automaton allow to express the timing properties. An enabling condition constrains the execution of a transition without forcing it to be taken. An invariant condition allows an automaton to stay at some location only as long as the constraint is satisfied.

3.2 Product of a Network of Timed Automata

A network of timed automata (NTA) can be composed into a global (*product*) timed automaton [39] in the following way. The transitions of the timed automata that do not correspond to a shared action are interleaved, whereas the transitions labelled with a shared action, are synchronised. Let $n \in \mathbb{N}$, $Id = \{1, \ldots, n\}$ be a non-empty and finite set of indices, $\mathcal{F} = \{\mathcal{A}_i \mid i \in Id\}$ be a network of timed automata $\mathcal{A}_i = (Act_i, Loc_i, \ell_i^0, \mathcal{X}_i, T_i, Inv_i, \mathcal{AP}_i, V_i)$ such that $\mathcal{X}_i \cap \mathcal{X}_j = \emptyset$ and $\mathcal{AP}_i \cap \mathcal{AP}_j = \emptyset$ for each $i \neq j$. Moreover, let $Id(\alpha) = \{i \in Id \mid \alpha \in Act_i\}$.

Definition 2. *The* product *of the network $\mathcal{F} = \{\mathcal{A}_i \mid i \in Id\}$ of timed automata is the timed automaton*

$$\mathcal{A} = (Act, Loc, \ell^0, \mathcal{X}, T, Inv, \mathcal{AP}, V), \text{ where}$$

- $Act = \bigcup_{i \in Id} Act_i$,
- $Loc = \prod_{i \in Id} Loc_i$,
- $\ell^0 = (\ell_1^0, \ldots, \ell_n^0)$,
- $\mathcal{X} = \bigcup_{i \in Id} \mathcal{X}_i$,
- $Inv(\ell_1, \ldots, \ell_n) = \bigwedge_{i=1}^n Inv_i(\ell_i)$,
- $\mathcal{AP} = \bigcup_{i \in Id} \mathcal{AP}_i$,
- $V(\ell_1, \ldots, \ell_n) = \bigcup_{i \in Id} V_i(\ell_i)$

and the transition relation T is defined as follows:
$((\ell_1, \ldots, \ell_n), \alpha, \bigwedge_{i \in I} \mathfrak{cc}_i, \bigcup_{i \in I} X_i, (\ell'_1, \ldots, \ell'_n)) \in T$ *iff* $(\forall i \in Id(\alpha))(\ell_i, \alpha, \mathfrak{cc}_i, X_i, \ell'_i) \in T_i$ *and* $(\forall i \in Id \setminus Id(\alpha))(\ell'_i = \ell_i)$.

3.3 Concrete Model

We define the semantics of a timed automaton $\mathcal{A}$.

Definition 3. *Let $\mathcal{A} = (Act, Loc, \ell^0, \mathcal{X}, T, Inv, \mathcal{AP}, V)$ be a* TA, *and v^0 be a clock valuation such that $\forall x \in \mathcal{X}$, $v^0(x) = 0$. The* concrete model *for $\mathcal{A}$ is a 4-tuple $\mathcal{M}_\mathcal{A} = (\mathcal{Q}, q^\iota, \longrightarrow, \mathcal{V})$, where*

- *$\mathcal{Q} = Loc \times (\mathbb{R}_{\geq})^{\mathcal{X}}$ is the set of the concrete states,*
- *$q^\iota = (\ell^0, v^0)$ is the initial state,*
- *$\mathcal{V}\colon \mathcal{Q} \to 2^{\mathcal{AP}}$ is a valuation function defined as $\mathcal{V}((\ell, v)) = V(\ell)$ for each state $(\ell, v) \in \mathcal{Q}$,*
- *$\longrightarrow \subseteq \mathcal{Q} \times (Act \cup \mathbb{R}_+) \times \mathcal{Q}$ is a transition relation on $\mathcal{Q}$ defined by action and time transitions as follows. For $\alpha \in Act$ and $\delta \in \mathbb{R}_+$:*
    1. *Action transition: $(\ell, v) \xrightarrow{\alpha} (\ell', v')$ if there is a transition $\ell \xrightarrow{\alpha, \mathfrak{cc}, X} \ell' \in T$ such that $v \models \mathfrak{cc} \wedge Inv(\ell)$ and $v' = v[X := 0]$ and $v' \models Inv(\ell')$,*
    2. *Time transition: $(\ell, v) \xrightarrow{\delta} (\ell, v+\delta)$ iff $v \models Inv(\ell)$ and $v+\delta \models Inv(\ell)$.*

Let $q \in \mathcal{Q}$ and $k \in \mathbb{N}$. A $(q-)$path ρ of $\mathcal{A}$ is a maximal[1] sequence of concrete states and action and time labels: $q_0 \xrightarrow{\delta_0} q_0 + \delta_0 \xrightarrow{\alpha_0} q_1 \xrightarrow{\delta_1} q_1 + \delta_1 \xrightarrow{\alpha_1} q_2 \xrightarrow{\delta_2} \ldots$, where $q_0 = q$, $\alpha_i \in Act$, and $\delta_i \in \mathbb{R}_+$ for each $i \geq 0$. We may sometimes omit the labels for brevity. The paths of length k we call k-paths. For a path $\rho = q_0, q_1, q_2, \ldots$, let $\rho(m)$ denote the state q_m and for a state $q = ((\ell_1, \ldots, \ell_n), v)$ let q^i denote ℓ_i. Moreover, let $\Pi_{\mathcal{M}_\mathcal{A}}(q)$ be the set of all q-paths in the model $\mathcal{M}_\mathcal{A}$, and each q^ι-path is called *initial*.

[1] It is not a strict prefix of another path.

4 Strategic Metric Temporal Logic

Metric Temporal Logic (MTL) [28, 12] extends Linear Temporal Logic (LTL) [40] by subscribing the temporal operators with time intervals. This allows for specifying time-dependent properties of systems. Strategic Metric Temporal Logic (SMTL) extends MTL by strategy operators $\langle\langle A \rangle\rangle\exists$ and $\langle\langle A \rangle\rangle\forall$ (preceding MTL formulae) for specifying existential and universal (resp.) strategic properties.

4.1 Syntax

Let $p \in \mathcal{AP}$ be an atomic proposition and $\mathcal{J}$ the set of all the intervals in $\mathbb{R}_+$ of the form $[a, a]$, (a, b), $[a, b)$, $(a, b]$, $[a, b]$, (a, ∞), or $[a, \infty)$, where $a, b \in \mathbb{N}$ and $a < b$, and let $\mathrm{I} \in \mathcal{J}$. The syntax of SMTL (MTL) is defined by the formulae ϕ (ψ, resp.) as follows:
$\phi := p \mid \neg\phi \mid \phi \wedge \phi \mid \phi \vee \phi \mid \langle\langle A \rangle\rangle\exists\psi \mid \langle\langle A \rangle\rangle\forall\psi$,
$\psi := p \mid \neg p \mid \psi \wedge \psi \mid \psi \vee \psi \mid \psi \mathbf{U}_\mathrm{I} \psi \mid \mathbf{G}_\mathrm{I}\psi$,
where $A \subseteq Id$.
The operators $\mathbf{U}_\mathrm{I}$ and $\mathbf{G}_\mathrm{I}$ are called *bounded until* and *bounded always*, respectively, and they are read as "until in the interval I" and "always in the interval I". The operator $\mathbf{F}_\mathrm{I}$ is defined in the standard way: $\mathbf{F}_\mathrm{I}\psi \overset{df}{=} \mathbf{true}\, \mathbf{U}_\mathrm{I}\, \psi$, where $\mathbf{true} := p \vee \neg p$, for some $p \in \mathcal{AP}$. Intuitively, $\langle\langle A \rangle\rangle\exists\psi$ means that the agents of A have a strategy s.t. it is possible to ensure ψ, while $\langle\langle A \rangle\rangle\forall\psi$ means that the agents of A have a strategy to inevitably ensure ψ.
The fragment of SMTL is called *existential* if it does not contain the subformulas $\langle\langle A \rangle\rangle\forall\psi$ and the negation is applied to the propositions only.

4.2 Continuous Time Asynchronous MAS

To give the semantics of SMTL we combine the formal definitions of *timed systems* [3] with *interpreted systems* [30] and define asynchronous *continuous-time multi-agent systems* (CMAS) [8].

Let $\mathcal{A}$ be a TA and $\mathcal{M}_\mathcal{A}$ the concrete model of $\mathcal{A}$. For a path $\rho = q_0, q_1, \ldots$, let $\Gamma_\rho(j) = \{i < j \mid \exists \delta_i \in \mathbb{R}_+ : \quad q_i \xrightarrow{\delta_i} q_{i+1}\}$, i.e., $\Gamma_\rho(j)$ is the set of the indices of the time transitions up to the state q_j. Now, we define a function $\zeta_\rho\colon \mathbb{N} \to \mathbb{R}_{\geq}$ such that, for all $j \geq 0$, $\zeta_\rho(j) = \sum_{i \in \Gamma_\rho(j)} \delta_i$, i.e., $\zeta_\rho(j)$ is the sum of all the time delays along the path ρ till the position j. For all $j \geq 0$, the function $\zeta_\rho(j)$ returns the value of the global time (called "*duration*" in [12]).

Definition 4. *A* continuous-time multi-agent system *(*CMAS*) consists of n agents (components) $\mathcal{A}gt = \{1, 2, \ldots, n\}$ each associated with a tuple $AG_i = (\mathcal{A}_i, P_i)$, where $\mathcal{A}_i = \big(Act_i, Loc_i, \ell_i^0, \mathcal{X}_i, T_i, Inv_i, \mathcal{AP}_i, V_i\big)$ is a* TA, *$P_i : Loc_i \to 2^{Act_i} \setminus \{\emptyset\}$ is an* i-local protocol *s.t. $(\forall \ell_i, \ell'_i \in Loc_i, \alpha_i \in Act_i, \mathfrak{cc}_i \in \mathcal{C}(\mathcal{X}_i), X_i \subseteq \mathcal{X}_i)$ if $(\ell_i, \alpha_i, \mathfrak{cc}_i, X_i, \ell'_i) \in T_i$, then $\alpha_i \in P_i(\ell_i)$.*

Next, we define the model of CMAS.

Definition 5 (Model of CMAS)**.** *The* model *of* CMAS *is a tuple $\mathcal{M} = (\mathcal{A}gt, \mathcal{A}, \{P_i\}_{i=1}^n)$, where $\mathcal{A}$ is the product of timed automata $\mathcal{A}_i$ for $i \in \mathcal{A}gt = \{1, \ldots, n\}$.*

SMTL is interpreted over the concrete model of a CMAS model.

Definition 6 (CTS)**.** *The* concrete model *of a* CMAS *model $\mathcal{M} = (\mathcal{A}gt, \mathcal{A}, \{P_i\}_{i=1}^n)$ is given by $(\mathcal{A}gt, \mathcal{M}_\mathcal{A}, \{P_i\}_{i=1}^n)$, where $\mathcal{M}_\mathcal{A}$ is the concrete model of $\mathcal{A}$.*

4.3 *Strategies*

We focus on imperfect information and imperfect recall (memoryless) strategies [43], denoted with ir. Intuitively, the agents take decisions based only on their own locations. Formally, an *(ir-)strategy* for $i \in \mathcal{A}gt$ is a function $\sigma_i \colon Loc_i \to Act_i$ s.t. $\sigma_i(\ell_i) \in P_i(\ell_i)$ for each $\ell_i \in Loc_i$. A notion of strategy is generalised to an agent coalition A, whose *joint strategy* σ_{A} is a tuple of strategies, one for each $i \in A$. The outcome of a strategy σ_{A} represents the possible path of the system, where A adopts this strategy, i.e., the coalition agents execute actions in line with their strategy, while the other agents freely choose from actions permitted by their protocols.

Definition 7 (Outcome)**.** *The* outcome *of a joint strategy* σ_{A} *in a state q of the concrete model* $\mathcal{M}_{\mathcal{A}}$*, is the set of q-paths* $out(q, \sigma_{\mathrm{A}})$*, such that* $\rho = q_0, \delta_0, q_0', \alpha_0, \dots \in out(q, \sigma_{\mathrm{A}})$ *iff* $q_0 = q$ *and for each* $i \in Id(\alpha_m) \cap \mathrm{A}$ *we have* $\alpha_m = \sigma_i(q_m'^{i})$*, for each* $m \geq 0$*.*

4.4 *Semantics of* SMTL

Definition 8. *The satisfaction relation* $\models$ *of an* SMTL *formula in the concrete model* $\mathcal{M}_{\mathcal{A}}$ *at state q, is defined inductively:*

- $q \models p$ *iff* $p \in \mathcal{V}(q)$,
- $q \models \neg\phi$ *iff* $q \not\models \phi$,
- $q \models \phi \wedge \varphi$ *iff* $q \models \phi$ *and* $q \models \varphi$,
- $q \models \phi \vee \varphi$ *iff* $q \models \phi$ *or* $q \models \varphi$,
- $q \models \langle\langle A\rangle\rangle\exists\psi$ *iff there is a joint strategy* σ_A *s.t. for some* $\rho \in out(q, \sigma_A)$ *we have* $(\rho, 0) \models \psi$,
- $q \models \langle\langle A\rangle\rangle\forall\psi$ *iff there is a joint strategy* σ_A *s.t. for all* $\rho \in out(q, \sigma_A)$ *we have* $(\rho, 0) \models \psi$*, where for each* $m \geq 0$*:*
- $(\rho, m) \models p$ *iff* $p \in \mathcal{V}(\rho(m))$,
- $(\rho, m) \models \neg p$ *iff* $p \notin \mathcal{V}(\rho(m))$,
- $(\rho, m) \models \psi \wedge \varphi$ *iff* $(\rho, m) \models \psi$ *and* $(\rho, m) \models \varphi$,
- $(\rho, m) \models \psi \vee \varphi$ *iff* $(\rho, m) \models \psi$ *or* $(\rho, m) \models \varphi$,
- $(\rho, m) \models \psi \mathbf{U}_{\mathrm{I}}\varphi$ *iff* $(\exists j \geq m)((\zeta_\rho(j) - \zeta_\rho(m)) \in \mathrm{I}$ *and* $(\rho, j) \models \varphi$ *and* $(\forall m \leqslant j' < j)(\rho, j') \models \psi)$,
- $(\rho, m) \models \mathbf{G}_{\mathrm{I}}\psi$ *iff* $(\forall j \geq m)((\zeta_\rho(j) - \zeta_\rho(m)) \in \mathrm{I} \Rightarrow (\rho, j) \models \psi)$.

An SMTL formula φ is *valid* in $\mathcal{M}_{\mathcal{A}}$, denoted by $\mathcal{M}_{\mathcal{A}} \models \varphi$, iff $\mathcal{M}_{\mathcal{A}}, q^\iota \models \varphi$, where q^ι is the initial state of $\mathcal{M}_{\mathcal{A}}$. An SMTL formula φ is satisfiable if there is a model $\mathcal{M}_{\mathcal{A}}$ s.t. $\mathcal{M}_{\mathcal{A}} \models \varphi$. Deciding whether φ is satisfiable is called the SMTL *satisfiability problem.* In our setting validity and satisfiability for MTL is defined slightly differently. An MTL formula ψ is *valid* in $\mathcal{M}_{\mathcal{A}}$ iff $\mathcal{M}_{\mathcal{A}}, q^\iota \models \langle\langle\emptyset\rangle\rangle\forall\psi$, i.e., ψ holds on all the initial paths of $\mathcal{M}_{\mathcal{A}}$. An MTL formula ψ is *satisfiable* if there is a model $\mathcal{M}_{\mathcal{A}}$ s.t. $\mathcal{M}_{\mathcal{A}}, q^\iota \models \langle\langle\emptyset\rangle\rangle\exists\psi$, i.e., ψ holds on some initial path of $\mathcal{M}_{\mathcal{A}}$. When certain bounds are put on a model, such as the number of transitions, agents, locations, or actions, then we deal with the *bounded* (S)MTL satisfiability problem.

5 Satisfiability checking for MTL and SMTL

We start with a SAT procedure for MTL and then extend it to SMTL.

5.1 *Satisfiability checking for* MTL

Let ψ be a formula of MTL. Our aim is to test ψ for satisfiability. To this end, we show how to synthesise a model for $\langle\langle\emptyset\rangle\rangle\exists\psi$, if it exists. The idea is to parametrically encode all the possible models under given restrictions and ask an SMT-solver to find a valuation of the parameters to get a model for $\langle\langle\emptyset\rangle\rangle\exists\psi$. Next, we show three SMT formulae: Spc_μ, Pth^k and $Tr^k(\psi)$, where Spc_μ constrains the parameters, Pth^k encodes the unfolding of a transition relation to a given depth $k = 0, 1, 2, \dots$ in a model for a network of (parametrically described) timed automata extended with protocols (CMAS, see Def. 4), and $Tr^k(\psi)$ encodes ψ on the symbolic k-path. We show that if the formula $H_\mu^k(\psi) := Spc_\mu \wedge Pth^k \wedge Tr^k(\psi)$ is satisfiable, then an SMT-solver returns values of the parameters and thus a model for $\langle\langle\emptyset\rangle\rangle\exists\psi$ is synthesised. If the SMT-solver does not find a satisfying valuation for $H_\mu^k(\psi)$, then k is increased and the search is repeated for $H_\mu^{k+1}(\psi)$. Optionally, the values of other fixed constraints can be increased.

Let $\mathcal{A}$ be the product automaton in the model for CMAS (see Def. 5). To define a parametric encoding of the transition relation for $\mathcal{A}$, we assume that there are given: n – the number of components (timed automata), components are numbered from 1 to n; l_j – the number of locations in j-th component, for $j = 1, \dots, n$; m – the number of local transitions in CMAS (by a local transition we mean any transition in a given component); and r – the number of different actions in CMAS, where $r \leq m$. We also assume that there is at most one clock in each component and that clocks only occur in the components numbered from 1 to nc, where $nc \leq n$. Although not every CMAS can be expressed with one clock in each component, most timed systems considered meet this condition. We have therefore made this assumption to improve efficiency. However, relaxing this condition is possible within our approach. Moreover, we assume that every location has a unique atomic proposition assigned, which is true in that location.

Let $\mu = [n, l_1, \dots l_n, nc, m, r]$. We call the elements of μ the meta-parameters.

5.2 *Parameters*

The following parameters, for $0 \leq i < m$, are used in the encoding of the transition relation: the component variables Y_i ranging from 1 to n; the location variables L_i and K_i for the transition source and target, resp., ranging from 0 to l_j (depending on the value of Y_i); the action variables A_i ranging from 1 to r; the variables C_i occurring in transition guards, ranging over $\mathbb{N}$; the Boolean variables R_i determining the occurrence of a reset on a transition. The 6-tuple $(Y_i, L_i, K_i, A_i, C_i, R_i)$ specifies the transition in the component Y_i, from location L_i to location K_i, at action A_i.

We also use parametric Boolean variables that define the occurrence of actions in components: B_a^j, for $1 \leq a \leq r$ and $1 \leq j \leq n$, defined as follows: $B_a^j =$ **true** if and only if action a occurs in component j. Moreover, we use parametric variables, which specify constants that occur in the invariants: D_j^l, for $1 \leq j \leq n$, $0 \leq l < l_j$, where $D_j^l = -1$ means that there is no invariant in the location.

5.3 *Constraints imposed on the parameters*

To encode a model the parameters are constrained as follows:

- The components are numbered from 1 to n:
$$\bigwedge_{i=0}^{m-1} (1 \leq Y_i \leq n)$$
- Each component is exploited:
$$\bigwedge_{j=1}^{n} \bigvee_{i=0}^{m-1} (Y_i = j)$$
- Each action is exploited:
$$\bigwedge_{a=1}^{r} \bigvee_{i=0}^{m-1} (A_i = a)$$
- The $j-$th component locations are numbered from 0 to $l_j - 1$:

$$\bigwedge_{i=0}^{m-1} \bigwedge_{j=1}^{n} \big(Y_i \neq j \vee (0 \leq L_i < l_j \wedge 0 \leq K_i < l_j)\big)$$

- Each action occurs in some component:
$$\bigwedge_{a=1}^{r} \bigvee_{j=1}^{n} B_a^j$$
- In each component there is an action:
$$\bigwedge_{j=1}^{n} \bigvee_{a=1}^{r} B_a^j$$
- In each component, an available action must be used:
$$\bigwedge_{a=1}^{r} \bigwedge_{j=1}^{n} \big(B_a^j \implies \bigvee_{i=0}^{m-1}(Y_i = j \wedge A_i = a)\big)$$
- In each component, an unavailable action must not be used:
$$\bigwedge_{a=1}^{r} \bigwedge_{j=1}^{n} \big(\neg B_a^j \implies \bigwedge_{i=0}^{m-1}(Y_i = j \implies A_i \neq a)\big)$$
- In each component, from each location outgoes some transition:
$$\bigwedge_{j=1}^{n} \bigwedge_{k=0}^{l_j-1} \bigvee_{i=0}^{m-1} \big(Y_i = j \wedge L_i = k\big)$$
- In each component, each location is a target of some transition:
$$\bigwedge_{j=1}^{n} \bigwedge_{k=0}^{l_j-1} \bigvee_{i=0}^{m-1} \big(Y_i = j \wedge K_i = k\big)$$
- There is only one transition between two locations labelled by a given action:

$$\bigwedge_{i=0}^{m-2} \bigwedge_{j>i}^{m-1} \Big(\big(Y_i = Y_j \wedge A_i = A_j\big) \implies \big(L_i \neq L_j \vee K_i \neq K_j\big)\Big)$$

The above constraints are for eliminating degenerate NTAs and they do not significantly constrain the class of NTAs under consideration. The conjunction of the above SMT formulae is denoted by Spc_μ.

5.4 Symbolic states and symbolic paths

We assume that the states of the model for $\mathcal{A}$ are encoded by *symbolic states* $\mathtt{w} = ((\mathtt{l}_1, \ldots, \mathtt{l}_n), (\mathtt{x}_1, \ldots, \mathtt{x}_{nc}))$, formed by *symbolic local states*, $(\mathtt{l}_1, \ldots, \mathtt{l}_n)$ and *symbolic clock valuations* $(\mathtt{x}_1, \ldots, \mathtt{x}_{nc})$. Each symbolic local state $\mathtt{l}_j$ ($1 \leq j \leq n$) is an individual variable ranging over the non-negative integers, and each symbolic clock valuation $\mathtt{x}_j$ ($1 \leq j \leq nc$) is an individual variable ranging over the non-negative real numbers. Furthermore, we assume that actions are encoded by *symbolic actions* $\mathtt{a}$ and time delays are encoded by *symbolic time delays* $\mathtt{d}$. Each symbolic action is an individual variable ranging over positive integers and each symbolic delay is an individual variable ranging over positive real numbers. In addition, each symbolic state's element is an individual variable called *symbolic global time*, which encodes the global time on a given symbolic path.

5.4.1 Encoding of the action transitions

The execution of an action a in a network of automata means that it is executed in every component in which it occurs (as long as it is enabled in that component). In components where the action a does not occur, the current location does not change. The encoding of the above idea is made possible by the Boolean variables B_a^j. The symbolic global state before execution of the action $\mathtt{a}$ is denoted by $\mathtt{w} = ((\mathtt{l}_1, \mathtt{x}_1), \ldots, (\mathtt{l}_n, \mathtt{x}_n))$, and the symbolic global state after execution of the action by $\mathtt{w}' = ((\mathtt{l}'_1, \mathtt{x}'_1), \ldots, (\mathtt{l}'_n, \mathtt{x}'_n))$. For a symbolic action $\mathtt{a}$, the formula encoding the transition in component j at action a, denoted by $\mathtt{AT}_a^j(\mathtt{w}, \mathtt{a}, \mathtt{w}')$, is defined as follows:

$$\mathtt{AT}_a^j(\mathtt{w}, \mathtt{a}, \mathtt{w}') := \bigvee_{i=0}^{m-1} \Big(Y_i = j \wedge A_i = a \wedge \mathtt{a} = a \wedge \mathtt{l}_j = L_i \wedge \mathtt{l}'_j = K_i \wedge Grd_i^j(\mathtt{w}) \wedge Rst_i^j(\mathtt{w}, \mathtt{w}') \wedge Inv_j(\mathtt{w}')\Big),$$

where, for $1 \leq j \leq nc$,

$$Grd_i^j(\mathtt{w}) := \mathtt{x}_j \geq C_i,$$
$$Rst_i^j(\mathtt{w}, \mathtt{w}') := (\neg R_i \wedge \mathtt{x}'_j = \mathtt{x}_j) \vee (R_i \wedge \mathtt{x}'_j = 0)$$
$$Inv_j(\mathtt{w}) := \bigvee_{l=0}^{l_j-1} \Big(\mathtt{l}_j = l \wedge \big(D_j^l = -1 \vee (D_j^l \geq 0 \wedge \mathtt{x}_j \leq D_j^l)\big)\Big)$$

$Grd_i^j(w)$ defines the enabling condition of the local action transition with number i in component j, and $Rst_i^j(w, w')$ determines whether the clock of component j is reset at the local transition with number i, and $Inv_j(w')$ defines the invariant in the target location in the local transition with number i. For j, such that $nc < j \leq n$, the formulae *Grd*, *Rst*, and *Inv* are defined as the formula **true**. Now, the formula $\mathtt{AT}_a(\mathtt{w}, \mathtt{a}, \mathtt{w}')$, encoding the transition in the network of timed automata at action a, is defined as follows:

$$\mathtt{AT}_a(\mathtt{w}, \mathtt{a}, \mathtt{w}') := \bigwedge_{j=1}^{n} \Big(\big(B_a^j \wedge \mathtt{AT}_a^j(\mathtt{w}, \mathtt{a}, \mathtt{w}')\big) \vee \big(\neg B_a^j \wedge \mathtt{l}_j = \mathtt{l}'_j \wedge \mathtt{x}_j = \mathtt{x}'_j\big)\Big)$$

Finally, $\mathtt{AT}(\mathtt{w}, \mathtt{a}, \mathtt{w}')$ encodes the action transition relation:

$$\mathtt{AT}(\mathtt{w}, \mathtt{a}, \mathtt{w}') := \bigvee_{a=1}^{r} \mathtt{AT}_a(\mathtt{w}, \mathtt{a}, \mathtt{w}')$$

5.4.2 Encoding of the time transitions

The encoding of the time transitions is partly based on that of [46]. The formula $\mathtt{TT}(\mathtt{w}, \mathtt{d}, \mathtt{w}')$ that encodes the time transition relation is defined as follows:

$$\mathtt{TT}(\mathtt{w}, \mathtt{d}, \mathtt{w}') := \mathtt{d} > 0 \wedge \bigwedge_{j=1}^{n} \big(\mathtt{l}_j = \mathtt{l}'_j\big) \wedge \bigwedge_{j=1}^{nc} \big(\mathtt{x}'_i = \mathtt{x}_i + \mathtt{d}\big) \wedge Inv(\mathtt{w}')$$

The above conjunction ensures that each time transition does not change the current locations, each clock is incremented by a real number greater than zero, and all the invariants associated with current locations remain satisfied.

5.4.3 Encoding of the symbolic paths

Let $\pi_k = (\mathtt{w}^0, \ldots, \mathtt{w}^k)$ be a sequence of symbolic states, $(\mathtt{a}_1, \ldots, \mathtt{a}_k)$ a sequence of symbolic actions, and $(\mathtt{d}_1, \ldots, \mathtt{d}_k)$ a sequence of symbolic time delays. The sequence π_k is called a *symbolic k-path*. In order to encode the unfolding of the transition relation for the model of $\mathcal{A}$ up to depth k we define the formula $\mathtt{T}(\mathtt{w}^j, (\mathtt{d}_{j+1}, \mathtt{a}_{j+1}), \mathtt{w}^{j+1})$ as follows: For each $0 \leq j < k$:

- if j is even, then $\mathtt{T}(\mathtt{w}^j, (\mathtt{d}_{j+1}, \mathtt{a}_{j+1}), \mathtt{w}^{j+1}) := \mathtt{a}_{j+1} = 0 \wedge \mathtt{TT}(\mathtt{w}^j, \mathtt{d}_{j+1}, \mathtt{w}^{j+1})$,
- if j is odd, then $\mathtt{T}(\mathtt{w}^j, (\mathtt{d}_{j+1}, \mathtt{a}_{j+1}), \mathtt{w}^{j+1}) := \mathtt{d}_{j+1} = 0 \wedge \mathtt{AT}(\mathtt{w}^j, \mathtt{a}_{j+1}, \mathtt{w}^{j+1})$.

The conditions imposed on the formula $\mathtt{T}(\mathtt{w}^j, (\mathtt{d}_{j+1}, \mathtt{a}_{j+1}), \mathtt{w}^{j+1})$ define all k-paths that start with a time transition and where any time transition is followed by an action transition and vice versa. The condition $a_{j+1} = 0$ ensures that a transition from $\mathtt{w}^j$ to $\mathtt{w}^{j+1}$ is a time transition, while the condition $d_{j+1} = 0$ ensures that a transition from $\mathtt{w}^j$ to $\mathtt{w}^{j+1}$ is an action transition.

Next, let $\mathtt{I}(\mathtt{w}^0) := (\bigwedge_{j=1}^{n} \mathtt{l}_j = \ell_j^0) \wedge (\bigwedge_{j=1}^{nc} \mathtt{x}_j = 0)$ be the formula encoding the initial state of the model of $\mathcal{A}$. The encoding of all initial $k-$paths in the model is defined by the formula Pth^k as follows:

$$Pth^k := \mathtt{I}(\mathtt{w}^0) \wedge \bigwedge_{j=0}^{k-1} \mathtt{T}(\mathtt{w}^j, (\mathtt{d}_{j+1}, \mathtt{a}_{j+1}), \mathtt{w}^{j+1})$$

5.4.4 Encoding of an MTL formula ψ

The encoding of an MTL formula ψ involves translating this formula into SMT. The translation of ψ into SMT is carried out in two steps. First, ψ is translated into the LTL formula $\mathrm{tr}(\psi)$, in which each interval I is replaced by a new propositional variable q_I. Then the formula $\mathrm{tr}(\psi)$ is translated to a quantifier-free first-order formula [47]. More formally, the translation from MTL to LTL of literals as well as of logical connectives is straightforward: $\mathrm{tr}(p) = p$, $\mathrm{tr}(\neg p) = \neg p$, $\mathrm{tr}(\psi_1 \wedge \psi_2) = \mathrm{tr}(\psi_1) \wedge \mathrm{tr}(\psi_2)$, $\mathrm{tr}(\psi_1 \vee \psi_2) = \mathrm{tr}(\psi_1) \vee \mathrm{tr}(\psi_2)$. The translation of $\psi_1 \mathbf{U}_{\mathrm{I}} \psi_2$ ensures that ψ_2 holds somewhere in the interval I and ψ_1 holds always before ψ_2: $\mathrm{tr}(\psi_1 \mathbf{U}_{\mathrm{I}} \psi_2) = \mathrm{tr}(\psi_1) \mathbf{U}(q_{\mathrm{I}} \wedge \mathrm{tr}(\psi_2))$, Similarly, the translation of $\mathbf{G}_{\mathrm{I}}\psi$ ensures that ψ always holds in the interval I: $\mathrm{tr}(\mathbf{G}_{\mathrm{I}}\psi) = \mathbf{G}(\neg q_{\mathrm{I}} \vee \mathrm{tr}(\psi))$.

The next step is the translation of $\mathrm{tr}(\psi)$ into the quantifier-free first order formula. This translation is defined inductively. It is standard for formulae without propositional variables q_{I}. But, each variable q_{I} is translated to the formula that ensures that the difference of the symbolic global time at a given depth and in a given starting point on a given symbolic k-path π belongs to the interval I.

The details of the translation for MTL formulae for the discrete semantics can be found in [48]. The translation for MTL formulae for the continuous semantics is similar as the SMT formulae resulting from the translation have an identical structure for both semantics. However, there are two differences in the resulting formulae. Firstly, in the continuous semantics, the type of the variables for storing global time in the path states, is non-negative reals, while in the discrete semantics, the type is non-negative integers. Secondly, in the continuous semantics the encoding of the state equivalence relation (used for identifying k-loops)[2] refers to dependencies between both the integer and fractional parts of the clock values, while in the discrete semantics only to the integer parts of these values.

Now. let $Tr^k(\psi)$ be the translation of the formula ψ into SMT on the symbolic k-path. Finally, let the SMT formula $H^k_\mu(\psi) := Spc_\mu \wedge Pth^k \wedge Tr^k(\psi)$.

Lemma 1. *If $H^k_\mu(\psi)$ is satisfiable, then ψ is satisfiable.*

Proof. If Spc_μ is satisfiable, then the parameters get values s.t. all constraints imposed on them are satisfied, so we get a concrete model $\mathcal{M}_{\mathcal{A}}$. If Pth^k is satisfiable, then there is at least one initial k-path in $\mathcal{M}_{\mathcal{A}}$. It follows from the satisfiability of $Tr^k(\psi)$ that ψ is true on some initial k-path of $\mathcal{M}_{\mathcal{A}}$. Since each k-path can be extended to a maximal path, we have $\mathcal{M}_{\mathcal{A}} \models \langle\langle\emptyset\rangle\rangle\exists\psi$, so ψ is satisfiable. □

Under some conditions we can also show the completeness of our encoding.

Lemma 2. *If there exists a* CMAS *C with metaparameters specified by μ and satisfying the constraints Spc_μ, such that ψ is true at some path of length k in C, then $H^k_\mu(\psi)$ is satisfiable.*

Proof. Let C be a CMAS with metaparameters specified by μ and satisfying the constraints Spc_μ, such that ψ is true at some path of length k in C. Since the formula $Spc_\mu \wedge Pth^k \wedge Tr^k(\psi)$ encodes the above, it is clearly satisfiable. As $H^k_\mu(\psi) = Tr^k(\psi) \wedge Spc_\mu \wedge Pth^k$, we have that $H^k_\mu(\psi)$ is satisfiable. □

5.5 *Satisfiability checking for* SMTL

Now, let ϕ be a formula of the existential fragment of SMTL. As before we deal with CMAS's, where each component contains at most one clock.

First we show how to encode formulas $\phi = \langle\langle\mathrm{A}\rangle\rangle\exists\psi$, where ψ is an MTL formula. Then, we extend the encoding to their Boolean combinations.

[2] For the definition of k-loops the reader is referred to [48]

Intuitively, the encoding of the operator $\langle\langle\mathrm{A}\rangle\rangle\exists$ restricts the symbolic k-path for every agent $i \in \mathrm{A}$ such that at each two global states sharing the same location of agent i the same action is executed by agent i. This is expressed by the SMT formula:

$$S(i,j,j') := \big((\textstyle\bigvee_{a=1}^{r}(\mathtt{a}_j = a \wedge B^i_a)) \wedge$$
$$(\mathtt{l}^j_i = \mathtt{l}^{j'}_i) \wedge (\textstyle\bigvee_{a=1}^{r}(\mathtt{a}_{j'} = a \wedge B^i_a))\big) \implies (\mathtt{a}_j = \mathtt{a}_{j'})$$

where $0 \leq j, j' \leq k$ denote points at the symbolic k-path.

Then, the restriction imposed on the symbolic k-path by the operator $\langle\langle\mathrm{A}\rangle\rangle\exists$ is as follows:

$$Stg^k_\phi := \textstyle\bigwedge_{i\in\mathrm{A}} \bigwedge_{j=0}^{k-1} \bigwedge_{j'=j+1}^{k} S(i,j,j')$$

Finally, the SMT formula $H^k_\mu(\phi) := Spc_\mu \wedge Pth^k \wedge Tr^k(\varphi) \wedge Stg^k_\phi$.

Lemma 3. *If $H^k_\mu(\phi)$ is satisfiable, then ϕ is satisfiable.*

Proof. Follows from Lemma 1 and the fact that Stg^k_ϕ restricts the symbolic k-path s.t. the agents of A follow some joint strategy. □

Now, we show how to encode any formula ϕ, which may contain more than one strategic operator. According to Sec. 4.1 $\phi = \phi_1 \odot_1 \phi_2 \odot_2 \ldots \odot_{s-1} \phi_s$, where $\odot_i \in \{\wedge, \vee\}$, $s \in \mathbb{N}$ is the number of the strategic operators in ϕ, and $\phi_i = \langle\langle\mathrm{A}_i\rangle\rangle\exists\psi_i$ for some $\psi_i \in$ MTL. Since each strategic operator potentially introduces a new strategy, each subformula ϕ_i needs to be interpreted over a separate path. Thus the translation into SMT requires as many as s symbolic k-paths. So, the encoding of each subformula ϕ_i is equal to $Spc_\mu \wedge H^{k,i}(\phi_i)$, where $H^{k,i}_\mu(\phi_i) := Pth^{k,i} \wedge Tr^{k,i}(\phi_i) \wedge Stg^{k,i}_{\phi_i}$, for $i \in P = \{1, \cdots, s\}$. Then, ϕ is encoded by $H^{k,P}_\mu(\phi) := Spc_\mu \wedge (H^{k,1}(\phi_1) \odot_1 H^{k,2}(\phi_2) \ldots \odot_{s-1} H^{k,s}(\phi_s))$.

Lemma 4. *If $H^{k,P}_\mu(\phi)$ is satisfiable, then ϕ is satisfiable.*

Proof. Follows from Lemma 3 by induction on the structure of ϕ. □

Similarly to the MTL case we can easily show the completeness of our encoding for SMTL.

Lemma 5. *If there exists a* CMAS *C with metaparameters specified by μ and satisfying the constraints Spc_μ, such that ϕ is true at s paths of length k in C, then $H^{k,P}_\mu(\phi)$ is satisfiable.*

Proof. Let C be a CMAS with metaparameters specified by μ and satisfying the constraints Spc_μ, such that ϕ is true at s paths of length k in C. Thus, the formula Spc_μ is satisfiable and there are paths $P = \{1, \cdots, s\}$ of C, where for each $i \in P$ the subformula ϕ_i is true at the depth k of the i-th path, what is encoded by the formula $H^{k,i}_\mu(\phi_i) := Pth^{k,i} \wedge Tr^{k,i}(\phi_i) \wedge Stg^{k,i}_{\phi_i}$, which is clearly satisfiable. Next, by induction on the structure of ϕ, we obtain that $H^{k,1}(\phi_1) \odot_1 H^{k,2}(\phi_2) \ldots \odot_{s-1} H^{k,s}(\phi_s)$ is also satisfiable. Finally, we have that $Spc_\mu \wedge (H^{k,1}(\phi_1) \odot_1 H^{k,2}(\phi_2) \ldots \odot_{s-1} H^{k,s}(\phi_s)) = H^{k,P}_\mu(\phi)$ is satisfiable. □

Our procedure is as a semi-decision one as if a given existential SMTL formula is satisfiable in a network of n timed automata (with m transitions), our procedure will eventually find a witness for its satisfiability.

Let $nt(\psi)$ be the highest degree of nesting of temporal operators in an MTL formula ψ. Formally, $nt(\psi) = 0$ for ψ being a propositional formula, $nt(\psi' \mathbf{U}_{\mathrm{I}} \psi'') = \max\{nt(\psi'), nt(\psi'')\} + 1$, and $nt(\mathbf{G}_{\mathrm{I}}\psi') = nt(\psi') + 1$. For $\phi = \phi_1 \odot_1 \phi_2 \odot_2 \ldots \odot_{s-1} \phi_s$ of SMTL, where $\phi_i = \langle\langle\mathrm{A}_i\rangle\rangle\exists\psi_i$ for some $\psi_i \in$ MTL, by $nt(\phi)$ we mean the maximum over $nt(\psi_i)$ for $i \in P = \{1, \cdots, s\}$. Moreover, let nl be the number of the locations in all components.

Theorem 1. *Let ϕ be an existential* SMTL *formula and each component of* CMAS *contain at most one clock. The complexity of $H^{k,P}_{\mu}(\phi)$ is in $O(m^2 \cdot n + s \cdot k \cdot (n \cdot nl \cdot m + k^{2 \cdot nt(\phi)-1} + k \cdot n \cdot m))$.*

Proof. Notice that $H^{k,P}_{\mu}(\phi)$ is the conjunction of Spc_{μ} with s subformulae $H^{k,i}_{\mu}(\phi_i)$ connected by boolean operators $\wedge$ and $\vee$. Each subformula $H^{k,i}_{\mu}(\phi_i)$ is the conjunction of three formulae $Pth^{k,i}$, $Tr^{k,i}(\phi_i)$, and $Stg^{k,i}(\phi_i)$. So we will examine the complexity of these formulae.

The formula Spc_{μ} is the conjunction of eleven formulae. The longest of them is of length $O(\max(n \cdot r \cdot m, n \cdot nl \cdot m))$. Since $r \leq m$, we can assume $r = m$ in the worst case, and we have $O(\max(m^2 \cdot n, n \cdot nl \cdot m)) = O(m^2 \cdot n)$ as $nl \leq m$.

The formula $Pth^{k,i}$ is the conjunction of two formulae. The longest of them is of length $O(k \cdot n \cdot nl \cdot m)$, where k is the depth of the unfolding of the transition relation for the model. Therefore, the complexity of $Pth^{k,i}$ is in $O(k \cdot n \cdot nl \cdot m)$.

The formula $Tr^{k,i}_{\psi_i}$ is the translation of the i-th MTL subformula following the i-th strategic operator in the formula ϕ_i. To estimate the length of the formula $Tr^{k,i}_{\psi_i}$, one has to analyse the translation of MTL formulae to SMT. This translation is analogous to the translation to SAT from [40]. First, the MTL formula is translated to LTL, enriched with new propositional variables responsible for the interpretation of intervals in temporal operators (Definition 9 of [40]). This translation is linear with respect to the length of the MTL formula to be translated. The resulting LTL formula is then translated to SMT analogously to Definition 12 of [40].

We will show by induction on $nt(\psi)$ that the length of Tr^{k}_{ψ} is in $O(k^{2 \cdot nt(\psi)})$.

Let us first consider the case where $nt(\psi) = 1$. The formula resulting from the translation of the MTL formula $\psi = \psi' \mathbf{U}_{\mathrm{I}} \psi''$ is the disjunction of two formulae. The analysis of these formulae under the assumption that ψ' and ψ'' are propositional formulae leads to the conclusion that the length of Tr^{k}_{ψ} is in $O(k^2 \cdot |Tr^{k}_{\psi'}| + k^2 \cdot |Tr^{k}_{\psi''}|) = O(k^2) = O(k^{2 \cdot nt(\psi)})$. The formula resulting from the translation of the MTL formula $\psi = \mathbf{G}_{\mathrm{I}} \psi'$ is the conjunction of three formulae. The analysis of these formulae under the assumption that ψ' is a propositional formula leads to the conclusion that the length of Tr^{k}_{ψ} is in $O(k^2 \cdot |Tr^{k}_{\psi'}|) = O(k^2) = O(k^{2 \cdot nt(\psi)})$.

In the case that $nt(\psi) > 1$, we first consider the formula $\psi = \psi' \mathbf{U}_{\mathrm{I}} \psi''$ and assume (by induction hypothesis) that $|Tr^{k}_{\psi'}|$ and $|Tr^{k}_{\psi''}|$ lie in $O(k^{2 \cdot (nt(\psi)-1)})$. The analysis of the translation of ψ leads to the conclusion that $|Tr^{k}_{\psi}|$ is in $O(k^2 \cdot g(\psi, k))$, where $g(\psi, k) \in O(k^{2 \cdot (nt(\psi)-1)})$. Thus, $|Tr^{k}_{\psi}|$ is in $O(k^2 \cdot k^{2 \cdot (nt(\psi)-1)}) = O(k^{2 \cdot nt(\psi)})$.

Let us now consider the formula $\psi = \mathbf{G}_{\mathrm{I}} \psi'$ and assume (by induction hypothesis) that $|Tr^{k}_{\psi'}|$ is in $O(k^{2 \cdot (nt(\psi)-1)})$. The analysis of the translation of ψ leads to the conclusion that $|Tr^{k}_{\psi}|$ is in $O(k^2 \cdot g(\psi, k))$, where $g(\psi, k) \in O(k^{2 \cdot (nt(\psi)-1)})$. Thus, $|Tr^{k}_{\psi}|$ is in $O(k^2 \cdot k^{2 \cdot (nt(\psi)-1)}) = O(k^{2 \cdot nt(\psi)})$.

From this we can conclude that the length of the translation of the MTL formula ψ such that $nt(\psi) > 0$ lies in $O(k^{2 \cdot nt(\psi)})$ and is thus polynomial with respect to k and exponential with respect to $nt(\psi)$.

The complexity of $Stg^{k,i}_{\phi_i}$ follows directly from its structure, and it is strictly correlated with the formula length. The auxiliary formula $S(i, j, j')$ (see Sec. 5.5 Satisfiability checking for SMTL), encoding the strategic constraints on a symbolic $k-$path for a single agent is of length $O(r)$. The translation of the strategic operator $\langle\langle \mathrm{A}_i \rangle\rangle \exists$ to an SMT instance requires the iteration trough the set of agents in the i-th coalition and the length of the symbolic path.

Thus, in the case of a single strategic operator $\langle\langle \mathrm{A}_i \rangle\rangle \exists$ the complexity of $Stg^{k,i}_{\phi_i}$ is in $O(k^2 \cdot |\mathrm{A}_i| \cdot r)$. As in the worst case the set A_i includes all the agents, we can replace it by n, and, as before, replace r by m, and we have the complexity of $Stg^{k,i}_{\phi_i}$ is in $O(k^2 \cdot n \cdot m)$.

In the general case, when the formula ϕ contains more than one strategic operator, and needs to be encoded over a set of symbolic paths P, where $|P| = s$, we have the complexity $O(k^2 \cdot n \cdot m \cdot s)$.

As mentioned above, we deal with a Boolean combination of s subformulae. Thus, we conclude that the complexity of the formula $H^{k,P}_{\phi}$ is in $O(m^2 \cdot n + s \cdot (k \cdot n \cdot nl \cdot m + k^{2 \cdot nt(\phi)} \cdot s + k^2 \cdot n \cdot m)$ $= O(m^2 \cdot n + s \cdot k \cdot (n \cdot nl \cdot m + k^{2 \cdot nt(\phi)-1} + k \cdot n \cdot m))$. □

We showed above the complexity of the encoding of ϕ to an SMT instance using quantifier free mixed linear and real arithmetic (QF_LIRA) theory. The translation is exponential in $nt(\phi)$ and polynomial in m and n. The complexity of the satisfiability checking of the resulting QF_LIRA formula is exponential in its length.

However, the complexity of the SMTL satisfiability problem depends on the adopted restrictions and varies from *EXPSPACE*-complete to undecidable in the worst case (see Sec. 2).

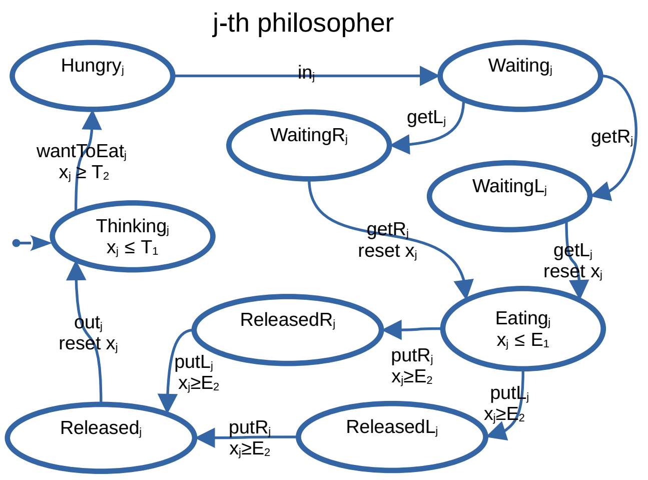

Figure 1: The TDPP system - the j-th philosopher automaton.

6 Experimental Results and Applications

Here, we present experimental results. First, we introduce a scalable example and SMTL formulae checked for satisfiability. Next, we discuss the methodology of the experiments and their results, obtained with our tool SMT4SMTL[3] which has been implemented in C++. It makes use of bash and Python scripts, and communicates with Z3[4] solver via files and command line. Our benchmark is the well-known dining philosophers problem [41] and its extension with clocks [48], slightly modified to give some agents a choice in selecting actions. The system consists of $n = 2p + 1$ agents given by timed automata, where the agents from 1 to p model philosophers (see Fig. 1), the subsequent p agents stand for the consecutive forks, and the last agent is the lackey who coordinates the philosophers' access to the dining room (see Fig. 2). We introduce four values determining the time constraints on the system behaviour: T_1, T_2, E_1,

[3] Download the tool from http://smt4smtl.epizy.com
[4] We observed that the Z3 solver was much more efficient than Yices2 here.

and E_2. Each philosopher j has to think at least T_2, and at most T_1 time units which is enforced by the guard $x_j \geq T_2$ and the invariant $x_j \leq T_1$, as well philosopher j is supposed to eat for at most E_1 ($x_j \leq E_1$), and at least E_2 time units ($x_j \geq E_2$). We tested the

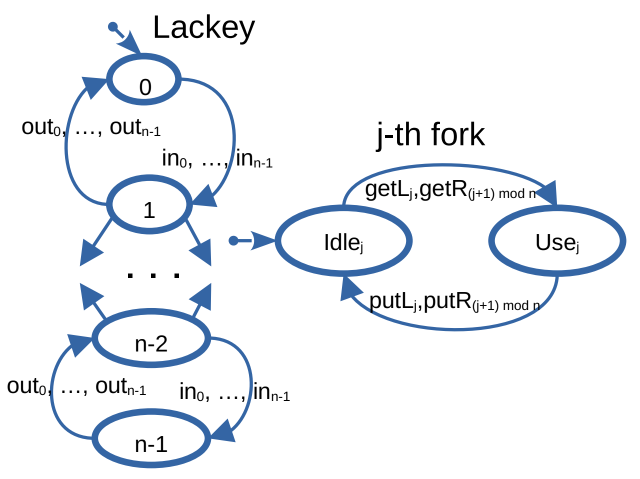

Figure 2: The TDPP system - the j-th fork and the lackey automata.

following properties:

- $\alpha = \langle\langle Lck \rangle\rangle \exists \big(F_{[1,E_2)}(\bigwedge_{j \in odd_p} Eating_j) \wedge (\bigwedge_{j \in odd_p}(F_{[0,\infty)} Hungry_j \wedge F_{[0,\infty)} Waiting_j \wedge F_{[0,\infty)} Released_j))\big)$ - the lackey has a strategy s.t. it is possible that all odd philosophers but the last one meet at the table at some point of time between 1 and E_2 and eat, and additionally we require that all of the crucial philosophers' locations must be reachable, where $odd_p = \{j \mid 1 \leq j < p \wedge j \bmod 2 = 1\}$.
- $\beta = \bigwedge_{j=1}^{p} \langle\langle Lck \rangle\rangle \exists F_{[1,E_2)}(Eating_j \wedge F_{[0,\infty)}(Thinking_j \wedge F_{[0,\infty)} Eating_j))$ - For every philosopher, the Lackey has a strategy that in the given time interval the philosopher can eat, then think, and then eat again.

The aforementioned formulae were tested 1 given the necessary values of $n = 2p + 1$, $m = 2p^2 + 13p$, $r = 7p$, and $nc = p$, what results from the structure of the TDPP system, with four variants of the constraints imposed on the parameter values:

- All - everything is a parameter, no constraints imposed. This is to show how our bounded satisfiability method works;
- Act - parameters Y_i, A_i, and B_a^j set, for $i = 0..m-1$, $a = 1..r$, and $j = 1..n$. In this setting we assign concrete actions and transitions to components, as well as actions to particular transitions, in order to decrease the search space for the SMT-solver;
- Lck - all agents but the last one (the lackey) are fully specified, for the lackey only the values of parameters B_a^n and Y_i are set, for $a = 1..r$, and i ranges through the lackey's transition set. The expected result was to synthesise a lackey agent;
- Nop - no parameters, the system is fully specified, the time constraints are $T1 = 2$, $T2 = 1$, $E1 = 1$, $E2 = 5$. This is a BMC case, given for reference and sanity check.

The preliminary experimental results are presented in Tab. 1 and Tab. 2. The meaning of the columns, from left to right, is as follows: the tested formula, the parameter setting variant, the number of philosophers, the total number of agents, the length of the shortest path satisfying the formula, time (in sec.) and memory (in MB) consumed by the BMC module and SMT-solver Z3 [32], respectively.

The experiments were performed on a server equipped with an Intel Xeon Gold 6234 3.30GHz CPU and 192GB RAM running Ubuntu Linux 20.04.5 and Z3-solver v4.8.7. Comparing the results obtained for the formulae α and β one can observe that additional requirements of reachability increase the length of the paths in the former case. Moreover, the more constraints imposed, the longer the path. In the case of β, additional symbolic k-paths increase the time and memory consumption. An interesting observation results from comparing the SMT-solver performance while testing satisfiability of formula β in All and Act variants: assigning actions and transitions to components greatly improves the satisfiability checking.

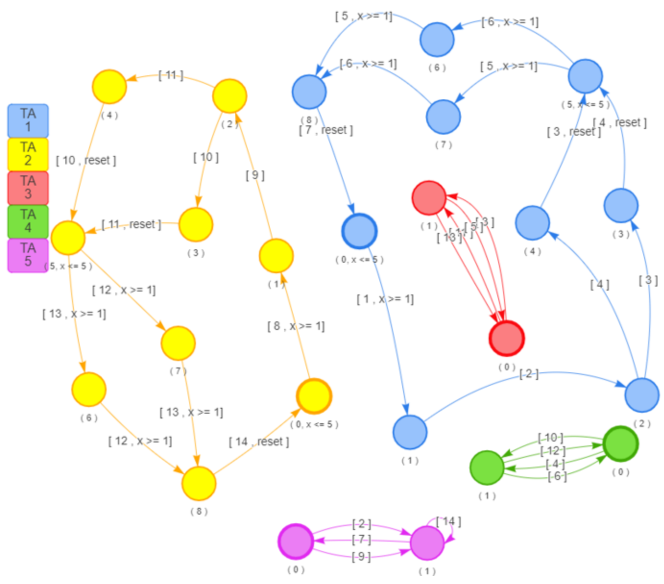

Figure 3: The NTA computed for β for 2 philosophers (Lck variant). The bottom (purple) automaton is the synthesised Lackey.

While the structure of the computed model is not relevant for the problem's satisfiability, it is essential for system synthesis. The preliminary results suggest that, in order to synthesise a realistic system, we need to impose as many constraints as possible, either through (strategic) formulae or by parameter constraints that partially define system components. Figure 3 illustrates this point, showing that the synthesised lackey is similar to the correct one in the part covered by the constraints and formula. Conversely, the NTA consistent with formula α, computed without any other constraints, is shown in Figure 4. This proves that formula α is satisfiable, but the obtained model differs from the one of Dining Philosophers problem. For instance, the first philosopher (the blue one) can move directly to the $Eating$ location (labelled $(5, x \leq 0)$) from the initial state ($Thinking$, labelled $(0, x \leq 1)$) without even obtaining a fork, and many locations are unreachable from the initial state. However, it is necessary to strike a balance between the solver's degree of freedom and the expected path length. The former negatively impacts the solver's performance, while the latter impairs parametric BMC efficiency. One of the most important advantages of our approach is that the reachable parts of the resulting models are often minimal, i.e., they consist only of the necessary transitions and locations that meet the given specification. Moreover, using the existential SMTL fragment we can seek, e.g., for security flaws in a system by synthesising a faulty/malicious agent which tries to reach a particular system state.

F	Par	p	n	k	BMCt	BMCm	Z3t	Z3m
α	All	2	5	4	0.2	6.0	1.6	44.5
		3	7	4	1.0	20.1	8.0	94.6
		4	9	4	3.1	45.3	25.8	169.2
		5	11	4	8.4	117.2	284.4	357.6
		6	13	4	21.8	168.0	528.7	608.8
		7	15	4	54.5	186.2	2077.1	923.6
	Act	2	5	4	0.2	6.0	0.8	25.4
		3	7	4	1.0	20.2	1.7	39.4
		4	9	8	12.7	47.2	13.7	97.0
		5	11	8	33.2	117.2	29.3	172.7
		6	13	12	183.9	168.9	123.1	383.7
		7	15	12	453.8	191.8	210.6	650.7
	Lck	2	5	12	2.2	6.5	3.4	34.4
		3	7	12	9.4	23.5	11.5	65.2
		4	9	24	118.1	47.9	125.9	229.7
		5	11	24	296.2	117.5	265.2	419.1
		6	13	36	1684.6	168.1	1495.6	1116.1
		7	15	36	4081.7	191.9	2413.5	1813.9
	Nop	2	5	12	2.1	6.5	3.3	33.8
		3	7	12	8.8	23.5	10.7	62.4
		4	9	24	111.0	47.3	119.6	217.0
		5	11	24	275.1	117.5	254.3	390.1
		6	13	36	1510.6	168.9	1289.9	1023.3
		7	15	36	3483.0	191.9	2112.0	1671.3

Table 1: Experimental results for formula α, 1 symbolic path

F	Par	p	n	k	BMCt	BMCm	Z3t	Z3m
β	All	2	5	4	0.4	6.3	2.9	55.3
		3	7	4	2.5	20.2	18.3	142.9
		4	9	4	9.5	46.3	207.8	375.6
		5	11	4	29.6	117.2	2086.5	937.7
		6	13	4	82.2	168.8	7215.7	1637.1
		7	15	4	216.4	186.3	59066.8	5365.8
	Act	2	5	4	0.4	5.9	1.9	48.4
		3	7	4	2.4	20.3	12.3	131.7
		4	9	4	9.5	47.6	69.3	298.6
		5	11	4	29.6	117.2	329.9	628.3
		6	13	4	82.2	168.8	1437.7	1238.1
		7	15	4	215.6	186.4	4585.0	2304.9
	Lck	2	5	22	19.1	73.8	37.0	89.0
		3	7	22	104.1	264.6	136.9	272.6
		4	9	22	389.8	516.8	444.1	775.5
		5	11	22	1163.5	983.2	1193.7	1763.2
		6	13	22	3150.6	1677.1	2589.2	3577.6
		7	15	22	8053.2	2629.4	5520.5	7049.1
	Nop	2	5	22	19.2	80.6	34.8	86.8
		3	7	22	103.6	270.5	137.2	261.5
		4	9	22	389.8	516.8	428.4	732.4
		5	11	22	1156.4	984.4	1092.0	1612.1
		6	13	22	3144.4	1677.1	2413.7	3364.7
		7	15	22	8053.0	2627.2	4770.2	6422.1

Table 2: Experimental results for formula β, p symbolic paths

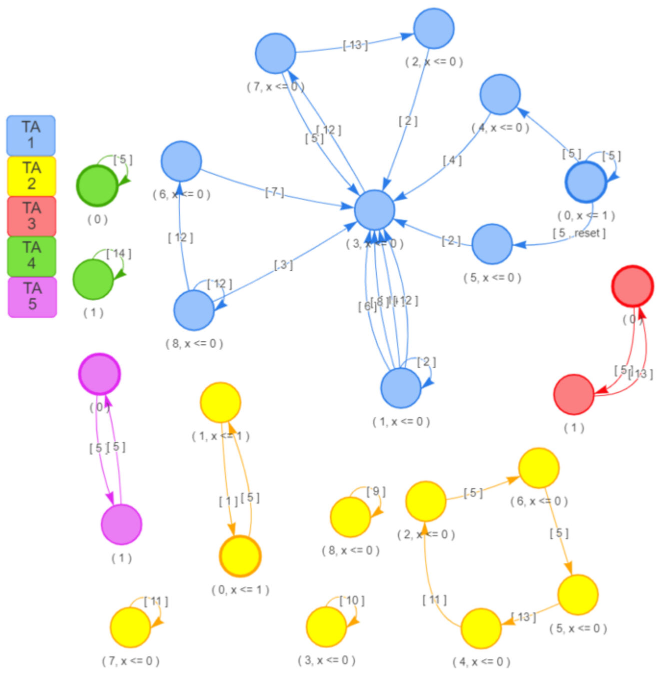

Figure 4: The NTA computed for α for 2 philosophers (All variant).

7 Conclusions

The paper presents a novel method for solving the BSAT problem and the synthesis problem for (S)MTL. The proposed method combines parametric BMC techniques with a translation to SMT and utilises the SMT-solver Z3. The method has been implemented and is supported by the SMT4SMTL tool. The effectiveness of the applied procedures, despite the high computational complexity of the problem, is a result of the adaptation of symbolic methods, efficient encoding of parametric models, and the use of parametric BMC for the existential fragment of SMTL.

Acknowledgments

The work of M. Kacprzak was supported by the Bialystok University of Technology, as part of the research grant WZ/WI-IIT/2/2022 of the Faculty of Computer Science and funded by Ministry of Science and Higher Education, Poland. W. Penczek acknowledges support from Luxembourg/Polish FNR/NCBiR project SpaceVote and CNRS/PAS project MOSART.

References

[1] R. Alur, C. Courcoubetis, and D. Dill, ‘Model checking in dense real-time’, *Information and Computation*, **104(1)**, 2–34, (1993).

[2] R. Alur and D. Dill, ‘The theory of timed automata’, in *Real-Time: Theory in Practice*, pp. 45–73. Springer Berlin Heidelberg, (1992).

[3] R. Alur and D. L. Dill, ‘Automata for modeling real-time systems’, in *Proceedings of the 17th International Colloquium on Automata, Languages and Programming, ICALP90*, volume 443 of *Lecture Notes in Computer Science*, pp. 322–335. Springer, (1990).

[4] R. Alur, T. Feder, and T. A. Henzinger, ‘The benefits of relaxing punctuality’, *J. ACM*, **43**(1), 116–146, (1996).

[5] R. Alur, T. A. Henzinger, and O. Kupferman, ‘Alternating-time Temporal Logic’, in *Proceedings of the 38th Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 100–109. IEEE Computer Society Press, (1997).

[6] R. Alur, T. A. Henzinger, and O. Kupferman, ‘Alternating-time Temporal Logic’, *Journal of the ACM*, **49**, 672–713, (2002).

[7] R. Alur and T.A. Henzinger, ‘Real-time logics: Complexity and expressiveness’, *Information and Computation*, **104**(1), 35–77, (1993).

[8] J. Arias, W. Jamroga, W. Penczek, L. Petrucci, and T. Sidoruk, 'Strategic (Timed) Computation Tree Logic', in *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2023*, pp. 382–390. ACM, (2023).

[9] C. Baier and J.-P. Katoen, *Principles of Model Checking*, MIT Press, 2008.

[10] F. Belardinelli, W. Jamroga, D. Kurpiewski, V. Malvone, and A. Murano, 'Strategy Logic with Simple Goals: Tractable reasoning about strategies', in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019*, pp. 88–94. ijcai.org, (2019).

[11] M. M. Bersani, M. Rossi, and S. P. Pietro, 'A tool for deciding the satisfiability of Continuous-time Metric Temporal Logic', *Acta Inf.*, **53**(2), 171–206, (2016).

[12] P. Bouyer, 'Model-checking timed temporal logics', *Electronic Notes in Theoretical Computer Science*, **231**, 323–341, (2009).

[13] P. Bouyer, N. Markey, J. Ouaknine, and J. Worrell, 'On expressiveness and complexity in real-time model checking', in *Automata, Languages and Programming, 35th International Colloquium, ICALP 2008*, volume 5126 of *Lecture Notes in Computer Science*, pp. 124–135. Springer, (2008).

[14] A. David, 'Deciding ATL* satisfiability by tableaux', in *International Conference on Automated Deduction*, pp. 214–228. Springer, (2015).

[15] C. A. Furia and P. Spoletini, 'Tomorrow and all our yesterdays: MTL satisfiability over the integers', in *Theoretical Aspects of Computing - ICTAC 2008*, volume 5160 of *Lecture Notes in Computer Science*, pp. 126–140. Springer, (2008).

[16] R. Gozalo-Brizuela and E. C. Garrido-Merchán, 'ChatGPT is not all you need. A state of the art review of large generative AI models', *CoRR*, **abs/2301.04655**, (2023).

[17] T. A. Henzinger, Z. Manna, and A. Pnueli, 'What good are digital clocks?', in *Automata, Languages and Programming*, pp. 545–558. Springer Berlin Heidelberg, (1992).

[18] X. Huang and R. van der Meyden, 'Symbolic Model Checking Epistemic Strategy Logic', in *Proceedings of the 28th AAAI Conference on Artificial Intelligence (AAAI14)*, pp. 1426–1432. AAAI Press, (2014).

[19] W. Jamroga, *Logical Methods for Specification and Verification of Multi-Agent Systems*, ICS PAS Publishing House, 2015.

[20] W. Jamroga, M. Knapik, D. Kurpiewski, and L. Mikulski, 'Approximate verification of strategic abilities under imperfect information', *Artif. Intell.*, **277**, (2019).

[21] W. Jamroga, B. Konikowska, and W. Penczek, 'Multi-Valued Verification of Strategic Ability', in *Proceedings of the 15th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2016)*, pp. 1180–1189. ACM, (2016).

[22] M. Kacprzak, A. Niewiadomski, and W. Penczek, 'SAT-based ATL satisfiability checking', in *Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning, KR 2020*, pp. 539–549, (2020).

[23] M. Kacprzak, A. Niewiadomski, and W. Penczek, 'Satisfiability checking of Strategy Logic with Simple Goals', in *Proceedings of the 18th International Conference on Principles of Knowledge Representation and Reasoning, KR 2021*, pp. 400–410, (2021).

[24] M. Kacprzak and W. Penczek, 'Unbounded Model Checking for Alternating-Time Temporal Logic', in *Proceedings of the 3rd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2004)*, pp. 646–653. IEEE Computer Society, (2004).

[25] M. Kacprzak and W. Penczek, 'Fully Symbolic Unbounded Model Checking for Alternating-time Temporal Logic', *Autonomous Agents and Multi-Agent Systems*, **11**(1), 69–89, (2005).

[26] M. Kanski, A. Niewiadomski, M. Kacprzak, W. Penczek, and W. Nabialek, 'SMT-Based Unbounded Model Checking for ATL', in *Verification and Evaluation of Computer and Communication Systems - 15th International Conference, VECoS 2021*, volume 13187 of *Lecture Notes in Computer Science*, pp. 43–58. Springer, (2021).

[27] T. Klenze, S. Bayless, and A.J. Hu, 'Fast, flexible, and minimal CTL synthesis via SMT', in *Computer Aided Verification*, pp. 136–156. Springer International Publishing, (2016).

[28] R. Koymans, 'Specifying real-time properties with Metric Temporal Logic', *Real-Time Syst.*, **2**(4), 255–299, (oct 1990).

[29] A. Lomuscio, H. Qu, and F. Raimondi, 'MCMAS: An Open-Source Model Checker for the Verification of Multi-Agent Systems', *International Journal on Software Tools for Technology Transfer*, **24**, 84–90, (2015).

[30] A. Lomuscio and M. Ryan, 'On the relation between interpreted systems and kripke models', in *Agents and Multi-Agent Systems Formalisms, Methodologies, and Applications*, volume 1441 of *Lecture Notes in Artificial Intelligence*, 46–59, Springer, (1997).

[31] F. Mogavero, A. Murano, G. Perelli, and M. Y. Vardi, 'Reasoning about strategies: on the satisfiability problem', *Log. Methods Comput. Sci.*, **13**(1), (2017).

[32] L. De Moura and N. Bjørner, 'Z3: an efficient SMT solver', in *Proceedings of 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'2008)*, volume 4963 of *LNCS*, pp. 337–340. Springer-Verlag, (2008).

[33] A. Nguyen-Duc, I. Sundbø, E. Nascimento, T. Conte, I. Ahmed, and P. Abrahamsson, 'A multiple case study of artificial intelligent system development in industry', in *EASE '20: Evaluation and Assessment in Software Engineering, Trondheim, Norway, April 15-17, 2020*, pp. 1–10. ACM, (2020).

[34] A. Niewiadomski, M. Kacprzak, D. Kurpiewski, M. Knapik, W. Penczek, and W. Jamroga, 'MsATL: A tool for SAT-based ATL satisfiability checking', in *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems, AAMAS '20*, pp. 2111–2113. International Foundation for Autonomous Agents and Multiagent Systems, (2020).

[35] J. Ouaknine and J. Worrell, 'Safety Metric Temporal Logic is fully decidable', in *Tools and Algorithms for the Construction and Analysis of Systems*, pp. 411–425. Springer Berlin Heidelberg, (2006).

[36] J. Ouaknine and J. Worrell, 'On the decidability and complexity of Metric Temporal Logic over finite words', *Logical Methods in Computer Science*, **Volume 3, Issue 1**, (February 2007).

[37] J. Ouaknine and J. Worrell, 'Some recent results in Metric Temporal Logic', in *Formal Modeling and Analysis of Timed Systems*, pp. 1–13. Springer Berlin Heidelberg, (2008).

[38] W. Penczek, 'Improving Efficiency of Model Checking for Variants of Alternating-time Temporal Logic', in *Proceedings of the 27th International Workshop on Concurrency, Specification and Programming*, volume 2240 of *CEUR Workshop Proceedings*. CEUR-WS.org, (2018).

[39] W. Penczek and A. Pólrola, *Advances in Verification of Time Petri Nets and Timed Automata: A Temporal Logic Approach*, volume 20 of *Studies in Computational Intelligence*, Springer, 2006.

[40] A. Pnueli, 'The temporal logic of programs', in *Proceedings of the 18th Annual Symposium on Foundations of Computer Science*, SFCS '77, pp. 46–57. IEEE Computer Society, (1977).

[41] D. K. Probst and H. F. Li, 'Verifying timed behavior automata with nonbinary delay constraints', in *Computer Aided Verification*, pp. 123–136. Springer Berlin Heidelberg, (1993).

[42] S. Schewe, 'ATL* satisfiability is 2EXPTIME-complete', in *Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, Proceedings, Part II - Track B: Logic, Semantics, and Theory of Programming & Track C: Security and Cryptography Foundations*, pp. 373–385, (2008).

[43] P. Y. Schobbens, 'Alternating-time Logic with Imperfect Recall', in *1st International Workshop on Logic and Communication in Multi-Agent Systems (LCMAS 2003)*, volume 85 of *Electronic Notes in Theoretical Computer Science*, pp. 1–12. Elsevier, (2004).

[44] G. van Drimmelen, 'Satisfiability in Alternating-Time Temporal Logic', in *18th Annual IEEE Symposium of Logic in Computer Science, 2003. Proceedings.*, pp. 208–217. IEEE, (2003).

[45] T. Wilke, 'Specifying timed state sequences in powerful decidable logics and timed automata', in *Proceedings of the Third International Symposium Organized Jointly with the Working Group Provably Correct Systems on Formal Techniques in Real-Time and Fault-Tolerant Systems*, ProCoS, pp. 694–715. Springer-Verlag, (1994).

[46] B. Woźna-Szcześniak, A. M. Zbrzezny, and A. Zbrzezny, 'SMT-based searching for k-quasi-optimal runs in weighted timed automata', *Fundam. Informaticae*, **152**(4), 411–433, (2017).

[47] A. M. Zbrzezny and A. Zbrzezny, 'Simple SMT-based bounded model checking for timed interpreted systems', in *Rough Sets*, pp. 487–504. Springer International Publishing, (2017).

[48] A. M. Zbrzezny and A. Zbrzezny, 'Bounded model checking for Metric Temporal Logic properties of timed automata with digital clocks', *Sensors*, **22**(23), 9552, (2022).

[49] Andrzej Zbrzezny, 'Improving the Translation from ECTL to SAT', *Fundam. Inf.*, **85**(1-4), 513–531, (2008).