Bounded Model Checking for Knowledge and Real Time

Bożena Woźna and Alessio Lomuscio *
Department of Computer Science, UCL
Gower Street, London WC1E 6BT,UK
email: {B.Wozna,A.Lomuscio}@cs.ucl.ac.uk

Wojciech Penczek †
Institute of Computer Science, PAS
Ordona 21, 01-237 Warsaw, Poland
email: penczek@ipipan.waw.pl

ABSTRACT

We present TECTLK, a logic to specify knowledge and real time in multi-agent systems. We show that the model checking problem is decidable, and we present an algorithm for TECTLK bounded model checking based on a discretisation method. We exemplify the use of the technique by means of the "Railroad Crossing System", a popular example in the multi-agent systems literature.

Categories and Subject Descriptors

F.3.1 [**Specifying and Verifying and Reasoning about Programs**]: Specification techniques; D.2.4 [**Software/Program Verification**]: Model checking; I.2.11 [**Distributed Artificial Intelligence**]: Multiagent systems

General Terms

Verification

Keywords

Model checking, interpreted systems, epistemic logic, real time.

1. INTRODUCTION

Model checking [8] is an area of formal methods concerned with automatic verification of hardware and software systems. It consists of a number of techniques to determine whether a given logical formula representing a specification is satisfied in a particular formal model representing the executions of a system. Originally developed for verification of (untimed) reactive systems, model checking has recently become an active subject of research in the area of multi-agent systems [6, 9, 12, 22]. In particular, recent contributions have focused on extending model checking techniques and tools [12, 16, 19, 20, 23, 26], to adapt them to the needs of multi-agent systems (MAS) formalisms.

As it was shown in [9], knowledge is a useful concept for analyzing the information state and the behaviour of agents in multi-agent systems. In particular, it is useful to reason about and to verify the evolution over time of epistemic states [11]. The usual assumption in the area is to consider time to be discrete. It is often argued that a model of time closer to reality should assume a continuous flow of instants. In this paper we make an attempt to evaluate the consequences of this suggestion in the context of epistemic states of multi-agent systems. Specifically, we make two contributions: first we present a logic, TECTLK, to reason about real time and knowledge in MAS; second, we present a technique for automatically verifying properties of MAS expressed in this logic.

The rest of the paper is organized as follows. The next section defines Real Time Interpreted Systems, the semantics on which we will work with throughout the paper. In Section 3 the logic TECTLK is introduced. In Section 4 a Bounded Model Checking method for TECTLK is presented. Section 5 shows how this method can be applied to the "railroad crossing system", a typical multi-agent systems example of time dependent systems. We conclude in Section 6 discussing related work.

2. INTERPRETED SYSTEMS ON REAL TIME

In this section we briefly recall the concept of timed automata, which were introduced in [2], and define a *Real Time Interpreted System.*

2.1 Timed Automata

Let $\mathbb{R} = [0, \infty)$ be a set of non-negative real numbers, $\mathbb{R}_+ = (0, \infty)$ be a set of positive real numbers, $\mathbb{N} = \{0, 1, \ldots\}$ a set of natural numbers, $\mathcal{X}$ a finite set of real variables, called *clocks*, $x \in \mathcal{X}$, $c \in \mathbb{N}$, and $\sim \in \{\leq, <, =, >, \geq\}$. The *clock constraints* over $\mathcal{X}$ are defined by the following grammar:

$$\mathfrak{cc} := true \mid x \sim c \mid \mathfrak{cc} \wedge \mathfrak{cc}$$

Notice that in order to keep the presentation as simple as possible and to use the discretisation method of [27] we do not allow for differences of clocks in $\mathcal{C}(\mathcal{X})$.

---

[*] The authors acknowledge support from the EPSRC (grant GR/S49353) and the Nuffield Foundation (grant NAL/690/G).

[†] Also affiliated with Podlasie Academy of Siedlce. The author acknowledges support from the Polish grant, No. 3T11C01128.

AAMAS'05, July 25-29, 2005, Utrecht, Netherlands.

The set of all the clock constraints over $\mathcal{X}$ is denoted by $\mathcal{C}(\mathcal{X})$. A *clock valuation* on $\mathcal{X}$ is a tuple $v \in \mathbb{R}^{|\mathcal{X}|}$. The value of the clock x in v is denoted by $v(x)$. For a valuation v and $\delta \in \mathbb{R}$, $v + \delta$ denotes the valuation v' such that for all $x \in \mathcal{X}$, $v'(x) = v(x) + \delta$. Moreover, for a subset of clocks $X \subseteq \mathcal{X}$, $v[X := 0]$ denotes the valuation v' such that for all $x \in X$, $v'(x) = 0$ and for all $x \in \mathcal{X} \setminus X$, $v'(x) = v(x)$. The satisfaction relation $\models$ for a clock constraint $\mathfrak{cc} \in \mathcal{C}(\mathcal{X})$ and $v \in \mathbb{R}^{|\mathcal{X}|}$ is defined inductively as follows:

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

For a constraint $\mathfrak{cc} \in \mathcal{C}(\mathcal{X})$, by $[\![\mathfrak{cc}]\!]$ we denote the set of all the clock valuations satisfying $\mathfrak{cc}$, i.e., $[\![\mathfrak{cc}]\!] = \{v \in \mathbb{R}^{|\mathcal{X}|} \mid v \models \mathfrak{cc}\}$.

Definition 1 (Timed automaton). *A* timed automaton *is a tuple* $\mathcal{TA} = (\mathfrak{Z}, L, l^0, E, \mathcal{X}, \mathfrak{I})$*, where*

- $\mathfrak{Z}$ *is a finite set of actions,*
- L *is a finite set of locations,*
- $l^0 \in L$ *is an initial location,*
- $\mathcal{X}$ *is a finite set of clocks,*
- $E \subseteq L \times \mathfrak{Z} \times \mathcal{C}(\mathcal{X}) \times 2^{\mathcal{X}} \times L$ *is a transition relation.*
- $\mathfrak{I} : L \to \mathcal{C}(\mathcal{X})$ *is a function, called a* location invariant*, which assigns to each location $l \in L$ a clock constraint defining the conditions under which $\mathcal{TA}$ can stay in l.*

Each element e of E is denoted by $l \xrightarrow{a,\mathfrak{cc},X} l'$, where l is a source location, l' is a target location, a is an action, $\mathfrak{cc}$ is the enabling condition for e, and $X \subseteq \mathcal{X}$ is the set of clocks to be reset.

A *state* of $\mathcal{TA}$ is a pair (l, v), where $l \in L$ and $v \in \mathbb{R}^{|\mathcal{X}|}$ is a clock valuation. The *dense state space* of $\mathcal{TA}$ is a tuple $(Q, q^0, \to)$, where $Q = L \times \mathbb{R}^{|\mathcal{X}|}$ is the set of all the states, $q^0 = (l^0, v^0)$ is the initial state such that $v^0(x) = 0$ for all $x \in \mathcal{X}$ and $v^0 \in [\![\mathfrak{I}(l^0)]\!]$, and $\to \subseteq Q \times (\mathfrak{Z} \cup \mathbb{R}) \times Q$ is the transition relation, defined by action- and time-successors as follows:

- for $a \in \mathfrak{Z}$, $(l, v) \xrightarrow{a} (l', v')$ iff $(\exists \mathfrak{cc} \in \mathcal{C}(\mathcal{X}))(\exists X \subseteq \mathcal{X})$ such that $l \xrightarrow{a,\mathfrak{cc},X} l' \in E$, $v \in [\![\mathfrak{cc}]\!]$, $v' = v[X := 0]$ and $v' \in [\![\mathfrak{I}(l')]\!]$ (*action successor*),
- for $\delta \in \mathbb{R}$, $(l, v) \xrightarrow{\delta} (l, v + \delta)$ iff $v + \delta \in [\![\mathfrak{I}(l)]\!]$ (*time successor*).

For $(l, v) \in Q$, let $(l, v) + \delta$ denote $(l, v + \delta)$. A q_0*-run* ρ of $\mathcal{TA}$ is a sequence of states: $q_0 \xrightarrow{\delta_0} q_0 + \delta_0 \xrightarrow{a_0} q_1 \xrightarrow{\delta_1} q_1 + \delta_1 \xrightarrow{a_1} q_2 \xrightarrow{\delta_2} \ldots$, where $q_i \in Q$, $a_i \in \mathfrak{Z}$ and $\delta_i \in \mathbb{R}_+$ for each $i \in \mathbb{N}$. A run ρ is said to be *progressive* iff $\Sigma_{i \in \mathbb{N}} \delta_i$ is unbounded. $\mathcal{TA}$ is *progressive* iff all its runs are progressive. For easiness of presentation, we consider only progressive timed automata. Note that progressiveness can be checked as in [21].

2.2 Parallel Composition

In general, we will model a multi-agent system by taking several timed automata running in parallel and communicating with each other. These concurrent timed automata can be composed into a global timed automaton as follows: 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 synchronized.

There are many different definitions of a parallel composition. We use a *multi-way synchronization*, i.e., we require that each component that contains a communication transition (labelled by a shared action) has to perform this action.

Let $\mathcal{TA}_i = (\mathfrak{Z}_i, L_i, l_i^0, E_i, \mathcal{X}_i, \mathfrak{I}_i)$ be a timed automaton, for $i = 1, \ldots, m$. To define a parallel composition of m timed automata, we assume that $L_i \cap L_j = \emptyset$, for all $i, j \in \{1, \ldots, m\}$ and $i \neq j$. Moreover, by $\mathfrak{Z}(a) = \{1 \leq i \leq m \mid a \in \mathfrak{Z}_i\}$ we denote the set of indexes representing the timed automata containing an action a.

Definition 2 (Parallel composition). *A* parallel composition *of m timed automata $\mathcal{TA}_i$ is a timed automaton $\mathcal{TA} = (\mathfrak{Z}, L, l^0, E, \mathrm{X}, \mathfrak{I})$, where $\mathfrak{Z} = \bigcup_{i=1}^m \mathfrak{Z}_i$, $L = \prod_{i=1}^m L_i$, $l^0 = (l_1^0, \ldots, l_m^0)$, $\mathcal{X} = \bigcup_{i=1}^m \mathcal{X}_i$, $\mathfrak{I}(l_1, \ldots, l_m) = \bigwedge_{i=1}^m \mathfrak{I}_i(l_i)$, and a transition*

$$((l_1, \ldots, l_m), a, \mathfrak{cc}, X, (l_1', \ldots, l_m')) \in E \textit{ iff}$$
$$(\forall i \in \mathfrak{Z}(a))(l_i, a, \mathfrak{cc}_i, X_i, l_i') \in E_i,\ \mathfrak{cc} = \bigwedge_{i \in \mathfrak{Z}(a)} \mathfrak{cc}_i,$$
$$X = \bigcup_{i \in \mathfrak{Z}(a)} X_i, \textit{ and } (\forall j \in \{1, \ldots, m\} \setminus \mathfrak{Z}(a))\ l_j' = l_j.$$

2.3 Real Time Interpreted System

In line with much literature in multi-agent systems, we use interpreted systems as a semantics for a temporal epistemic language. For this, we need to adapt them to work on real time: this is why we take timed automata as the underlying modelling concept (as opposed to the standard protocols of interpreted systems). To define *real time interpreted systems*, we first partition the set of clock valuations as in [1].

Let $\mathcal{TA}$ be a timed automaton, $\mathcal{C}(\mathcal{TA}) \subseteq \mathcal{C}(\mathcal{X})$ be a non-empty set containing all the clock constrains occurring in any enabling condition used in the transition relation E or in a state invariant of $\mathcal{TA}$. Moreover, let c_{max} be the largest constant appearing in $\mathcal{C}(\mathcal{TA})$, and $fr(\sigma)$ ($\lfloor \sigma \rfloor$), for $\sigma \in \mathbb{R}$, denote the fractional (integral part of σ, resp.). We define an equivalence relation $\simeq$ in the set of all the clock valuations as follows (see Figure 1 for an intuition).

Definition 3 ([1]). *For two clock valuations $v, v' \in \mathbb{R}^{|\mathcal{X}|}$, we say that $v \simeq v'$ iff for all $x, y \in \mathcal{X}$ the following conditions are met:*

1. $v(x) > c_{max}$ *iff* $v'(x) > c_{max}$
2. *if* $v(x) \leq c_{max}$ *and* $v(y) \leq c_{max}$ *then*
   - *a.)* $\lfloor v(x) \rfloor = \lfloor v'(x) \rfloor$,
   - *b.)* $fr(v(x)) = 0$ *iff* $fr(v'(x)) = 0$, *and*
   - *c.)* $fr(v(x)) \leq fr(v(y))$ *iff* $fr(v'(x)) \leq fr(v'(y))$.

This partitions $\mathcal{C}(\mathcal{TA})$ into *(detailed) zones*, denoted by Z, Z', and so on.

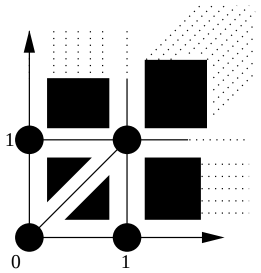

Figure 1: Equivalence of clock valuations for two clocks with $c_{max} = 1$.

Let $\mathcal{AG}$ be a set of m agents such that each agent is modelled by a timed automaton $\mathcal{TA}_i = (\mathfrak{Z}_i, L_i, l_i^0, E_i, \mathcal{X}_i, \mathfrak{I}_i)$, for $i = 1, \ldots, m$, $\mathcal{TA} = (\mathfrak{Z}, L, l^0, E, \mathrm{X}, \mathfrak{I})$ be the parallel composition of all the agents, and $l_i : Q \to L_i$ be a function which returns the location of agent i from a global state. Moreover, let $\mathcal{PV}_i$ be a set of propositional variables containing the symbol $\top$, for $i \in \{1, \ldots, m\}$, and $\mathcal{PV} = \bigcup_{i=1}^{m} \mathcal{PV}_i$. In order to reason about multi-agent systems, where each agent is represented by a timed automaton, an existence of a (local) valuation function $\mathcal{V}_{\mathcal{TA}_i} : L_i \to 2^{\mathcal{PV}_i}$ for the i-the agent is assumed. We require that $\top \in \mathcal{V}_{\mathcal{TA}_i}(l)$ for each $l \in L_i$. The (global) valuation function $\mathcal{V}_{\mathcal{TA}} : L \to 2^{\mathcal{PV}}$ for the parallel composition is defined by $\mathcal{V}_{\mathcal{TA}}((l_1, \ldots, l_m)) = \bigcup_{i=1}^{m} \mathcal{V}_{\mathcal{TA}_i}(l_i)$. Given this, a *real time interpreted system* is defined as follows.

Definition 4. *A* real time interpreted system *is a tuple* $M = (Q, q^0, \to, \sim_1, \ldots, \sim_m, \mathcal{V})$*, where*

- Q*,* q^0*, and* $\to$ *are defined as in the definition of the dense state space for* $\mathcal{TA}$*.*
- $\sim_i \subseteq Q \times Q$ *is an (accessibility) relation defined by* $(l, v) \sim_i (l', v')$ *iff* $l_i((l, v)) = l_i((l', v'))$ *and* $v \simeq v'$*, for each agent* i*. Obviously* $\sim_i$ *is an equivalence relation.*
- $\mathcal{V} : Q \to 2^{\mathcal{PV}}$ *is a valuation function that extends* $\mathcal{V}_{\mathcal{TA}}$ *as follows* $\mathcal{V}((l, v)) = \mathcal{V}_{\mathcal{TA}}(l)$*.*

In the above definition the notion of $\sim_i$ requires an explanation. Two states $(l, v), (l', v') \in Q$ are in the accessibility relation for an agent i if their i-local states are the same and in addition their clock valuations v and v' are elements of the same zone. The latter condition seems to be the weakest one, which can be imposed in our framework.

3. THE LOGIC TECTLK

We now introduce TECTLK, a logic for knowledge and real time. This extends TECTL [1] by means of epistemic operators.

3.1 Syntax

Let $\mathcal{PV}$ be a set of propositional variables containing the symbol $\top$, $\mathcal{AG}$ a set of m agents, and I an interval in $\mathbb{R}$ with integer bounds of the form $[n, n']$, $[n, n')$, $(n, n']$, (n, n'), (n, ∞), and $[n, \infty)$, for $n, n' \in \mathbb{N}$. Let $p \in \mathcal{PV}$,$i \in \mathcal{AG}$, and $\Gamma \subseteq \mathcal{AG}$, the *set of* TECTLK *formulas* is defined by the following grammar:

$$\begin{array}{c} \varphi := p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \mathrm{E}(\varphi \mathrm{U}_I \varphi) \mid \mathrm{E}(\varphi \mathrm{R}_I \varphi) \mid \\ \overline{\mathrm{K}}_i \varphi \mid \overline{\mathrm{D}}_\Gamma \varphi \mid \overline{\mathrm{C}}_\Gamma \varphi \mid \overline{\mathrm{E}}_\Gamma \varphi \end{array}$$

The other basic temporal modalities are defined as usual: $\bot \overset{def}{=} \neg\top$, $\mathrm{EG}_I \varphi \overset{def}{=} \mathrm{E}(\bot \mathrm{R}_I \varphi)$, $\mathrm{EF}_I \varphi \overset{def}{=} \mathrm{E}(\top \mathrm{U}_I \varphi)$. Moreover, $\alpha \to \beta \overset{def}{=} \neg\alpha \vee \beta$.

Note that TECTLK is a subset of the fusion [5] of the two underlying logics TCTL and $S5$ for the knowledge operators. Obviously, defining the full fusion would not be problematic [25] but we use a fragment because it is more suited for the model checking method that we use later.

3.2 Semantics

Let $\mathcal{AG}$ be a set of m agents, where each agent is modelled by a timed automaton $\mathcal{TA}_i = (\mathfrak{Z}_i, L_i, l_i^0, E_i, \mathcal{X}_i, \mathfrak{I}_i)$, for $i = 1, \ldots, m$, $\mathcal{TA} = (\mathfrak{Z}, L, l^0, E, \mathrm{X}, \mathfrak{I})$ be their parallel composition, and $M = (Q, q^0, \to, \sim_1, \ldots, \sim_m, \mathcal{V})$ be a *real time interpreted system.* Moreover, let $\rho = q_0 \xrightarrow{\delta_0} q_0 + \delta_0 \xrightarrow{a_0} q_1 \xrightarrow{\delta_1} q_1 + \delta_1 \xrightarrow{a_1} q_2 \xrightarrow{\delta_2} \ldots$ be a run of $\mathcal{TA}$ such that $\delta_i \in \mathbb{R}_+$ for $i \in \mathbb{N}$, and let $f_{\mathcal{TA}}(q_0)$ denote the set of all such q_0-runs of $\mathcal{TA}$. In order to give a semantics to TECTLK, we introduce the notation of a *dense path* π_ρ *corresponding to run* ρ*. A dense path* π_ρ *corresponding to* ρ is a mapping from $\mathbb{R}$ to a set of states Q[1], such that $\pi_\rho(r) = s_i + \delta$ for $r = \Sigma_{j=0}^{i} \delta_j + \delta$ with $i \in \mathbb{N}$ and $0 \leq \delta < \delta_i$. Moreover, we define the following epistemic relations: $\sim_\Gamma^E = \bigcup_{i \in \Gamma} \sim_i$, and $\sim_\Gamma^C = (\sim_\Gamma^E)^+$ (the transitive closure of $\sim_\Gamma^E$), and $\sim_\Gamma^D = \bigcap_{i \in \Gamma} \sim_i$, where $\Gamma \subseteq \mathcal{AG}$.

Definition 5 (Satisfaction). *Let* $M = (Q, q^0, \to, \sim_1, \ldots, \sim_m, \mathcal{V})$ *be a* real time interpreted system *such that the set* Q *contains reachable states*[2] *only.* $M, q \models \alpha$ *denotes that* α *is true at state* s *in the model* M*.* M *is omitted, if it is implicitly understood. The satisfaction relation* $\models$ *is defined inductively as follows:*

$$\begin{array}{lll} q_0 \models p & \textit{iff} & p \in \mathcal{V}(q_0), \\ q_0 \models \neg p & \textit{iff} & p \notin \mathcal{V}(q_0), \\ q_0 \models \varphi \vee \psi & \textit{iff} & q_0 \models \varphi \textit{ or } q_0 \models \psi, \\ q_0 \models \varphi \wedge \psi & \textit{iff} & q_0 \models \varphi \textit{ and } q_0 \models \psi, \\ q_0 \models \mathrm{E}(\varphi \mathrm{U}_I \psi) & \textit{iff} & (\exists\, \rho \in f_{\mathcal{TA}}(q_0))(\exists r \in I)[\pi_\rho(r) \models \psi \textit{ and} \\ & & (\forall r' < r)\ \pi_\rho(r') \models \varphi], \\ q_0 \models \mathrm{E}(\varphi \mathrm{R}_I \psi) & \textit{iff} & (\exists\, \rho \in f_{\mathcal{TA}}(q_0))(\forall r \in I)[\pi_\rho(r) \models \psi \textit{ or} \\ & & (\exists r' < r)\ \pi_\rho(r') \models \varphi], \\ q_0 \models \overline{\mathrm{K}}_i \alpha & \textit{iff} & (\exists q' \in Q)(q_0 \sim_i q' \textit{ and } q' \models \alpha), \\ q_0 \models \overline{\mathrm{D}}_\Gamma \alpha & \textit{iff} & (\exists q' \in Q)(q_0 \sim_\Gamma^D q' \textit{ and } q' \models \alpha), \\ q_0 \models \overline{\mathrm{E}}_\Gamma \alpha & \textit{iff} & (\exists q' \in Q)(q_0 \sim_\Gamma^E q' \textit{ and } q' \models \alpha), \\ q_0 \models \overline{\mathrm{C}}_\Gamma \alpha & \textit{iff} & (\exists q' \in Q)(q_0 \sim_\Gamma^C q' \textit{ and } q' \models \alpha). \end{array}$$

A TECTLK formula φ is *satisfiable* iff there exists a real time interpreted system $M = (Q, q^0, \to, \sim_1, \ldots, \sim_m, \mathcal{V})$ and a state q of M, such that $M, q \models \varphi$. A TECTLK formula φ is *valid in* M (denoted $M \models \varphi$) iff $M, q^0 \models \varphi$, i.e., φ is true at the initial state of the model M; checking validity given M and φ is called the *model checking problem.*

Note that the "full" logic of real time TCTL is undecidable [1]. Since real time interpreted systems can be shown to be as expressive as the TCTL-structure of a time graph in [1], and the fusion [5] between TCTL and S5 for knowledge is a proper extension of TCTL, it follows that problem of satisfiability for the full fusion is also undecidable. Still, the decidability of TECTL is not known; if TECTL were decidable, it would be straightforward to show that TECTLK is also decidable on real time interpreted systems. In fact, we do not have decidability results for the satisfiability problem for TECTLK but for our application purposes we are interested in *the model checking problem* for TECTLK, and this can be shown to be decidable (Lemma 1).

Lemma 1. *Given a real time interpreted system* M_d *and a* TECTLK *formula* φ*, there is a decision procedure for checking whether or not* M_d *satisfies* φ*.*

Proof Sketch. Construct the region graph as in [1], and extend it by taking the epistemic relation $\sim_i$ as defined in Definition 4. The proof in [1] can now be extended to the full TECTLK syntax. $\square$

---

[1] This can be done because of the assumption that $\delta_i \in \mathbb{R}_+$.

[2] A state $s \in Q$ is reachable if there is a q^0-run ρ such that there exists a state in ρ equal to s.

4. TECTLK BOUNDED MODEL CHECKING

Bounded model checking (BMC) is one of the SAT-based (satisfiability checking) methods, and it was introduced as a technique complementary to the BDD-based symbolic model checking for LTL [4]. The main idea of BMC is to search for an execution (or a set of executions) of the system of some length k, which constitutes a counterexample for a tested property. If no counterexample of length k can be found, then k is incrementally increased by one until it reaches the size of the model. The efficiency of this method is based upon the observation that if a system is faulty, then often only a (small) fragment of its state space is sufficient for finding an error. Obviously, when testing large models and complex formulas the efficiency of the BMC method is dependent on the speed of the chosen SAT solver, on which the test is carried out. As SAT checkers have been progressively becoming more effective, the efficiency of BMC has improved, an observation experimentally demonstrated in, among others, [4, 14, 18, 19].

To perform Bounded Model Checking on TECTLK we proceed by extending the technique employed for TCTL [17] and ECTLK [16]: first we discretise real time interpreted system; second we translate the model checking problem from TECTLK to another logic, called ECTLK$_y$; third we define BMC for ECTLK$_y$.

4.1 Discretisation

Let $\mathcal{AG}$ be a finite set of agents, where each agent is modelled by a timed automaton, $\mathcal{TA} = (\mathfrak{Z}, L, l^0, E, \mathcal{X}, \mathfrak{I})$ be their parallel composition, $\mathcal{V}_{\mathcal{TA}}$ be a valuation function for $\mathcal{TA}$, φ be a TECTLK formula, and $M = (Q, q^0, \rightarrow, \sim_1, \ldots, \sim_m, \mathcal{V})$ be the real time interpreted system for $\mathcal{TA}$. The discretisation scheme [27] consists in representing zones by one or more (but finitely many) specially chosen representatives. Formally, we proceed as follows.

First, a discretisation step is chosen. Here we take $\Delta = 1/d$, where $d = 2^{\lceil log_2(2|\mathcal{X}|)\rceil}$. Note that we could take a different discretisation step (see [10] for a "survey"), but we have chosen this one because it preserves time successors for clock valuations. This property is important, because it allows us to represent the time successor in a straightforward way, and makes its the Boolean encoding feasible. The chosen step does not preserve the action successors, but this is not problematic. A *discretised clock space* is defined as $\mathbb{D}^{|\mathcal{X}|} = \{k\Delta \mid 0 \leq k\Delta \leq 2c_{max}(\varphi) + 2,\ k \in \mathbb{N}\}^{|\mathcal{X}|}$, where $c_{max}(\varphi)$ is the largest constant appearing in $\mathcal{C}(\mathcal{TA})$ and in any timed interval in φ. In other words, the clocks cannot go beyond $2c_{max}(\varphi)+2$; this is because while evaluating TECTLK formula φ over timed automata we do not need to distinguish between clock valuations above $c_{max}(\varphi) + 1$. Therefore, the maximal values of time delays can be restricted to $c_{max}(\varphi)+1$, and the set of values that can change a valuation in a zone can be defined as $\mathbb{E} = \{k\Delta \mid 0 \leq k\Delta < c_{max}(\varphi) + 1\}$. To make sure that the above two definitions can be applied we will guarantee below that before any time transition the value of every clock does not exceed $c_{max}(\varphi) + 1$ (this is obtianed by "adjust" transitions).

Next, a subset $\mathbb{U} \subseteq \mathbb{D}^{|\mathcal{X}|}$ is taken that preserves time delays by insisting that either the values of all the clocks in $v \in \mathbb{U}$ are only even or only odd multiplications of Δ. To preserve action successors we will later use "adjust" transitions (see Figure 2 for an example of a discretised clock space).

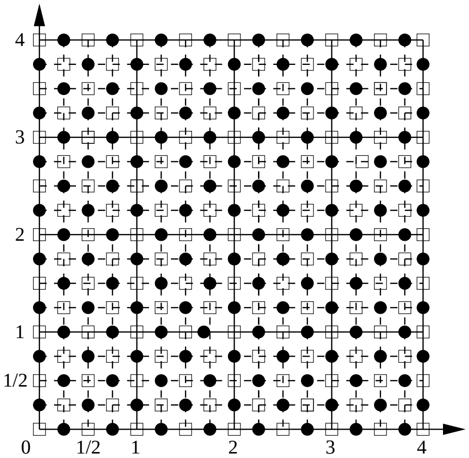

Figure 2: The discretised clock space for two clocks, and a TECTLK fromula φ with $c_{max}(\varphi) = 1$. The square points are the elements of the set $\mathbb{U}$ while the circled and square points together are the elements of the set $\mathbb{D}^2$.

Definition 6. *A* discretised interpreted system *is a structure $M_d = (S_d, s^0, \rightarrow_d, \sim_1^d, \ldots, \sim_m^d, \mathcal{V}_d)$, where $S_d = L \times \mathbb{U}$, $s^0 = (l^0, v^0)$ is the initial state, and the relation $\rightarrow_d \subseteq S_d \times (\mathfrak{Z} \cup \{\tau\}) \times S_d$ is defined by:*

1.) *Time successor: $(l, v) \xrightarrow{\tau}_d (l, v')$ iff $(l, v) \xrightarrow{\delta};\xrightarrow{\epsilon} (l, v')$ for some $\delta \in \mathbb{E} \setminus \{0\}$, and $(\forall \delta' \leq \delta)(v' + \delta' \simeq v$ or $v' + \delta' \simeq v')]$, and if $v \simeq v'$, then $v \simeq v' + \delta''$ for each $\delta'' \in \mathbb{E} \setminus \{0\}$, where*

   $(l, v) \xrightarrow{\epsilon} (l, v')$ iff $v' \in \mathbb{U}$, $(\forall x \in \mathcal{X})(v'(x) \leq c_{max}(\varphi) + 1)$, and $v \simeq v'$ (adjust transition)

2.) *Action successor: $(l, v) \xrightarrow{a}_d (l', v')$ iff (l, v) is not boundary*[3] *and $[(l, v) \xrightarrow{a};\xrightarrow{\epsilon} (l', v')$ or $(l, v) \xrightarrow{\tau}_d;\xrightarrow{a};\xrightarrow{\epsilon} (l', v')]$, for $a \in \mathfrak{Z}$.*

The accessibility relation $\sim_i^d = \sim_i \cap (S_d \times S_d)$, for $i \in \mathcal{AG}$, where $\sim_i$ is the accessibility relation in M. The valuation function $\mathcal{V}_d : S_d \rightarrow 2^{\mathcal{PV}}$ is given by $\mathcal{V}_d((l, v)) = \mathcal{V}_{\mathcal{TA}}(l)$.

For an intuition in the above, consider a region as a pair (l, Z) for a location $l \in L$ and a zone Z. A time successor represents a move to the time successor region, which clearly shares the same location. In order to make sure that valuations of the clocks do not go beyond $2c_{max}(\varphi) + 2$ and before any transition the value of every clock does not exceed $c_{max}(\varphi) + 1$, we adjust each time successor transition by an ϵ-move. An action successor represents a move by an action transition (adjusted by an ϵ-move in order to stay in $\mathbb{U}$) taken from a non-boundary region and possibly preceded by the time successor step. Note that an action successor cannot be taken from a boundary region to make sure that there are no two consecutive action successor steps in a run.

---

[3] A state (l, v) is boundary if for any $\delta \in \{k\Delta \mid 0 < k\Delta < 1\}$, it is not the case that $(v \simeq v + \delta)$.

4.2 Translation from TECTLK to ECTLK$_y$

In general, the model checking problem for TECTL can be translated into the model checking problem for a fair version of ECTL [1]. Since here we have assumed that we deal with progressive timed automata only, to extend the procedure of [1] to TECTLK, we introduce slightly different logic ECTLK$_y$, as presented below.

The idea is as follows. Let $\mathcal{AG}$ be a finite set of agents modelled by timed automata, $\mathcal{TA}$ be their parallel composition, $\mathcal{V}_{\mathcal{TA}}$ a valuation function, and φ a TECTLK formula. First, we extend $\mathcal{TA}$ with a new clock (denoted by y), an action, and transitions to obtain an automaton $\mathcal{TA}_\varphi$. The clock y corresponds to all the timing intervals $\{I_1, \ldots, I_r\}$ appearing in φ, and special transitions are used to reset the new clock. These transitions are used to start the runs over which subformulas of φ are checked. We then construct the discretised interpreted system for $\mathcal{TA}_\varphi$ and augment its valuation function with the set of propositional variables which contains a new proposition $p_{y \in I_i}$ for every interval I_i appearing in φ, and a new proposition p_b representing that a state is boundary. Finally, we translate the TECTLK formula φ into an ECTLK$_y$ formula $\psi = \mathrm{cr}(\varphi)$ such that model checking of φ over the discretised interpreted system for $\mathcal{TA}$ can be reduced to model checking of ψ over the discretised interpreted system for $\mathcal{TA}_\varphi$.

We follow [17] for the first two steps of the translation and we refer to it for more details; here we focus on the final step.

In order to translate a TECTLK formula φ into the corresponding ECTLK formula ψ we need to modify the ECTLK language into ECTLK$_y$ by reinterpreting the next-time operator, denoted now by X_y. This language is interpreted over discretised interpreted system for $\mathcal{TA}_\varphi$. The modality X_y is interpreted only over the new transitions that reset the new clock y[4], whereas the other operators are interpreted over all other old transitions. Formally, for $p \in \mathcal{PV}$, $i \in \mathcal{AG}$ and $\Gamma \subseteq \mathcal{AG}$, the set $\mathfrak{F}$ of ECTLK$_y$ formulas is defined by the grammar:

$$\begin{array}{c}\alpha := p \mid \neg p \mid \alpha \wedge \alpha \mid \alpha \vee \alpha \mid \mathrm{X}_y\alpha \mid \mathrm{E}(\alpha \mathrm{U} \alpha) \mid \\ \mathrm{E}(\alpha \mathrm{R} \alpha) \mid \overline{\mathrm{K}}_\Gamma\alpha \mid \overline{\mathrm{D}}_\Gamma\alpha \mid \overline{\mathrm{C}}_\Gamma\alpha \mid \overline{\mathrm{E}}_\Gamma\alpha\end{array}$$

The satisfaction relation $\models$ for ECTLK$_y$ formulas is defined as the corresponding satisfaction relation for ECTLK formulas [16]. It only differs in the operator X_y, which is defined as follows:

$$M_d, (l, v) \models \mathrm{X}_y\alpha \text{ iff } M_d, (l, v[\{y\} := 0]) \models \alpha.$$

The TECTLK formula φ is translated inductively into the ECTLK$_y$ formula $\mathrm{cr}(\varphi)$ as follows:

- $\mathrm{cr}(p) = p$ for $p \in \mathcal{PV}'$,
- $cr(\neg p) = \neg cr(p)$ for $p \in \mathcal{PV}'$,
- $\mathrm{cr}(\alpha \vee \beta) = \mathrm{cr}(\alpha) \vee \mathrm{cr}(\beta)$,
- $\mathrm{cr}(\alpha \wedge \beta) = \mathrm{cr}(\alpha) \wedge \mathrm{cr}(\beta)$,
- $\mathrm{cr}(\overline{\mathrm{K}}_i\alpha) = \overline{\mathrm{K}}_i\mathrm{cr}(\alpha)$,
- $\mathrm{cr}(\overline{\mathrm{D}}_i\alpha) = \overline{\mathrm{D}}_i\mathrm{cr}(\alpha)$,
- $\mathrm{cr}(\overline{\mathrm{E}}_i\alpha) = \overline{\mathrm{E}}_i\mathrm{cr}(\alpha)$,
- $\mathrm{cr}(\overline{\mathrm{C}}_i\alpha) = \overline{\mathrm{C}}_i\mathrm{cr}(\alpha)$,
- $\mathrm{cr}(\mathrm{E}(\alpha \mathrm{U}_{I_i} \beta)) = \mathrm{X}_y(\mathrm{E}(\mathrm{cr}(\alpha)\mathrm{U}(\mathrm{cr}(\beta) \wedge p_{y \in I_i} \wedge \gamma)))$,
- $\mathrm{cr}(\mathrm{E}(\alpha \mathrm{R}_{I_i} \beta)) = \mathrm{X}_y(\mathrm{E}(\mathrm{cr}(\alpha)\mathrm{R}(\neg p_{y \in I_i} \vee (\mathrm{cr}(\beta) \wedge \gamma)))))$,

where $\gamma = p_b \vee \mathrm{cr}(\alpha)$.

The following lemma shows that validity of the TECTLK formula φ over the real time interpreted system for $\mathcal{TA}$ is equivalent to the validity of the corresponding ECTLK$_y$ formula $\mathrm{cr}(\varphi)$ over the discretised interpreted system for $\mathcal{TA}_\varphi$ with the extended valuation function.

LEMMA 2. *$M \models \varphi$ iff $M_d \models cr(\varphi)$, for each* TECTLK *formula φ.*

PROOF. The proof follows directly from Lemma on Correctness of the labelling algorithm of [1], Theorem 4.1 of [27] for TECTL part of TECTLK, and from the definition of the relation $\sim_i$ for the epistemic part of TECTLK. ☐

Next, we show a BMC method for ECTLK$_y$ over discretised interpreted system. Since we have defined a translation from TECTLK to ECTLK$_y$, we obtain a BMC method for TECTLK.

4.3 ECTLK$_y$ Bounded Model Checking

Consider a discretised interpreted system $M_d = (S_d, s^0, \rightarrow_d, \sim_1^d, \ldots, \sim_m^d, \mathcal{V}_d)$, an ECTLK$_y$ formula $\psi = \mathrm{cr}(\varphi)$, where φ is a TECTLK formula, and a bound $k \in \mathbb{N}_+$. The main idea of BMC for ECTLK$_y$ consists in translating the model checking problem of an ECTLK$_y$ formula into the problem of satisfiability of a propositional formula $[M_d, \psi]_k = [M_d^{\psi, s^0}]_k \wedge [\psi]_k^{0,0}$. The way we interpret this translation for ECTLK$_y$ is a combination of the techniques presented in [16, 17], i.e., BMC for ECTLK and BMC for ECTL$_y$. The translation is based on $k-$bounded semantics for ECTLK$_y$, which is defined as follows.

Let us denote by $\rightarrow_{\mathcal{TA}}$ the part of $\rightarrow_d$, where transitions are labelled with elements of $\mathfrak{Z} \cup \{\tau\}$, and by $\rightarrow_y$ the transitions that reset the clock y. Then, a *path* π in M_d is a sequence $(s_0, s_1, \ldots)$ of states such that $s_i \rightarrow_{\mathcal{TA}} s_{i+1}$ for each $i \in \mathbb{N}$. A path of length k is called *$k-$path*, and the set of all the k-paths starting at s in M_d is denoted by $\Pi_k(s)$. Furthermore, let α, β be ECTL$_y$ subformulas of ψ, $k \in \mathbb{N}_+$ be a bound, then $(M_d, k), s \models \alpha$ denotes that α is true at the state s of M_d with the bound k. (M_d, k) is omitted if it is clear from the context. The relation $\models$ is defined inductively as follows:

$$\begin{array}{ll}
s \models p & \text{iff } p \in \mathcal{V}_d(s) \\
s \models \neg p & \text{iff } p \notin \mathcal{V}_d(s), \\
s \models \alpha \vee \beta & \text{iff } s \models \alpha \text{ or } s \models \beta, \\
s \models \alpha \wedge \beta & \text{iff } s \models \alpha \text{ and } s \models \beta, \\
s \models \mathrm{X}_y\alpha & \text{iff } \exists s' \in S\ (s \rightarrow_y s' \text{ and } s' \models \alpha), \\
s \models \overline{\mathrm{K}}_i\alpha & \text{iff } \exists_{\pi \in P_k(s^0)} \exists_{0 \leq j \leq k} (\pi(j) \models \alpha \text{ and } s \sim_i \pi(j)), \\
s \models \overline{\mathrm{D}}_\Gamma\alpha & \text{iff } \exists_{\pi \in P_k(s^0)} \exists_{0 \leq j \leq k} (\pi(j) \models \alpha \text{ and } s \sim_\Gamma^D \pi(j)), \\
s \models \overline{\mathrm{E}}_\Gamma\alpha & \text{iff } \exists_{\pi \in P_k(s^0)} \exists_{0 \leq j \leq k} (\pi(j) \models \alpha \text{ and } s \sim_\Gamma^E \pi(j)), \\
s \models \overline{\mathrm{C}}_\Gamma\alpha & \text{iff } \exists_{\pi \in P_k(s^0)} \exists_{0 \leq j \leq k} (\pi(j) \models \alpha \text{ and } s \sim_\Gamma^C \pi(j)), \\
s \models \mathrm{E}(\alpha \mathrm{U} \beta) & \text{iff } \exists_{\pi \in \Pi_k(s)} \exists_{0 \leq j \leq k} (\pi(j) \models \beta \text{ and} \\
 & \quad \forall_{0 \leq i < j}\ \pi(i) \models \alpha), \\
s \models \mathrm{E}(\alpha \mathrm{R} \beta) & \text{iff } \exists_{\pi \in \Pi_k(s)} (\exists_{0 \leq j \leq k} (\pi(j) \models \alpha \text{ and} \\
 & \quad \forall_{0 \leq i \leq j}\ \pi(i) \models \beta)) \text{ or } (\forall_{0 \leq j \leq k}\ \pi(j) \models \beta \\
 & \quad \text{and } \exists_{0 \leq i \leq k} \pi(k) \rightarrow_{\mathcal{TA}} \pi(i)).
\end{array}$$

The first conjunct of $[M_d, \psi]_k$ represents all the possible submodels of M_d which consist of $f_k(\psi)$ $k-$paths of M_d. The function f_k gives a bound for the number of k-paths in the submodel M_k of M_d such that the validity of ψ in M_k (i.e., validity in M_d with the bound k) is equivalent to the validity of ψ in M_d. The function $f_k : \mathfrak{F} \rightarrow \mathbb{N}$ is defined by:

- $f_k(p) = f_k(\neg p) = 0$, where $p \in \mathcal{PV}$,

---

[4] These transitions can be executed from the boundary regions.

- $f_k(\mathrm{X}_y\alpha) = f_k(\alpha)$,
- $f_k(\alpha \vee \beta) = max\{f_k(\alpha), f_k(\beta)\}$,
- $f_k(\alpha \wedge \beta) = f_k(\alpha) + f_k(\beta)$,
- $f_k(\mathrm{E}(\alpha \mathrm{U} \beta)) = k \cdot f_k(\alpha) + f_k(\beta) + 1$,
- $f_k(\mathrm{E}(\alpha \mathrm{R} \beta)) = k \cdot f_k(\beta) + f_k(\alpha) + 1$,
- $f_k(Y\alpha) = f_k(\alpha) + 1$, for $Y \in \{\overline{\mathrm{K}}_i, \overline{\mathrm{D}}_\Gamma, \overline{\mathrm{E}}_\Gamma\}$,
- $f_k(\overline{\mathrm{C}}_\Gamma\alpha) = f_k(\alpha) + k$.

The second conjunct of $[M_d, \psi]_k$ encodes a number of constraints that must be satisfied on the submodel M_k of M_d, which consists of all the k-paths of M_d, for ψ to be satisfied. Once this translation is defined, checking satisfiability of an ECTLK_y formula can be done by means of a SAT-checker.

Let us assume that each state s of the discretised interpreted system M_d is encoded by a bit-vector whose length, say b, depends on the number of locations, the number of clocks, the discretisation step, and $c_{max}(\varphi)$. So, each state s of M_d can be represented by a vector $w = (w[1], \ldots, w[b])$ (called *global state variable*), where each $w[i]$, for $i = 1, \ldots, b$, is a propositional variable (called state variable). A finite sequence $(w_0, \ldots, w_k)$ of global state variables is called a *symbolic k-path*[5]. Moreover, we assume familiarity with basic BMC contributions as the definitions of propositional formulas $I_s(w)$, $p(w)$, $H(w, w')$, $\mathcal{R}(w, w')$, and $R_y(w, w')$ as defined in [17], while for the propositional formula H_i representing logical equivalence between local state encodings of agent i we take the following definition: $H_i(w, w')$ is a formula over two global state variables $w = (\mathfrak{l}, \mathfrak{v})$, $w' = (\mathfrak{l}', \mathfrak{v}')$, which is true for valuations $s_\mathfrak{l}$ of $\mathfrak{l}$, $s_{\mathfrak{l}'}$ of $\mathfrak{l}'$, $s_\mathfrak{v}$ of $\mathfrak{v}$, and $s_{\mathfrak{v}'}$ of $\mathfrak{v}'$ iff $l_i(s_\mathfrak{l}) = l_i(s_{\mathfrak{l}'})$ and $s_\mathfrak{v} \simeq s_{\mathfrak{v}'}$.

The propositional formula $[M_d, \psi]_k$ is defined over state variables $w_{0,0}$, $w_{n,m}$, for $0 \leq m \leq k$ and $1 \leq n \leq f_k(\psi) + r$[6]; note that the index n denotes the number of a symbolic path, whereas the index m the position at that path. The formal definition of its first conjunct is the following:

$$[M_d^{\psi,s^0}]_k := I_{s^0}(w_{0,0}) \wedge \bigwedge_{n=1}^{f_k(\psi)} \bigwedge_{m=0}^{k-1} \mathcal{R}(w_{m,n}, w_{m+1,n})$$

Let $H = H(w_{m,n}, w_{0,i})$, $H_l = H_l(w_{m,n}, w_{j,i})$. The second conjunct of $[M_d, \psi]_k$, i.e., the formula $[\psi]_k^{[0,0]}$ is defined as follows.

$$\begin{array}{ll}
[p]_k^{[m,n]} & := p(w_{m,n}), \\
[\neg p]_k^{[m,n]} & := \neg p(w_{m,n}), \\
[\alpha \wedge \beta]_k^{[m,n]} & := [\alpha]_k^{[m,n]} \wedge [\beta]_k^{[m,n]}, \\
[\alpha \vee \beta]_k^{[m,n]} & := [\alpha]_k^{[m,n]} \vee [\beta]_k^{[m,n]}, \\
[\mathrm{E}(\alpha \mathrm{U} \beta)]_k^{[m,n]} & := \bigvee_{i=1}^{f_k(\psi)} (H \wedge \bigvee_{j=0}^{k} ([\beta]_k^{[j,i]} \wedge \bigwedge_{l=0}^{j-1} [\alpha]_k^{[l,i]})), \\
[\mathrm{E}(\alpha \mathrm{R} \beta)]_k^{[m,n]} & := \bigvee_{i=1}^{f_k(\psi)} (H \wedge (\bigvee_{j=0}^{k} ([\alpha]_k^{[j,i]} \wedge \ \bigwedge_{l=0}^{j} [\beta]_k^{[l,i]}) \\
 & \quad \vee \bigwedge_{j=0}^{k} [\beta]_k^{[j,i]} \wedge \bigvee_{l=0}^{k} \mathcal{R}(w_{k,i}, w_{l,i}))), \\
[\mathrm{X}_y \alpha]_k^{[m,n]} & := \bigvee_{j=1}^{r} (R_y(w_{m,n}, w_{0,f_k(\psi)+j}) \wedge [\alpha]_k^{[0,f_k(\psi)+j]}), \\
[\overline{\mathrm{K}}_l \alpha]_k^{[m,n]} & := \bigvee_{i=1}^{f_k(\psi)} (I_{s^0}(w_{0,i}) \wedge \bigvee_{j=0}^{k} ([\alpha]_k^{[j,i]} \wedge H_l)), \\
[\overline{\mathrm{D}}_\Gamma \alpha]_k^{[m,n]} & := \bigvee_{i=1}^{f_k(\psi)} (I_{s^0}(w_{0,i}) \wedge \bigvee_{j=0}^{k} ([\alpha]_k^{[j,i]} \wedge \bigwedge_{l \in \Gamma} H_l)), \\
[\overline{\mathrm{E}}_\Gamma \alpha]_k^{[m,n]} & := \bigvee_{i=1}^{f_k(\psi)} (I_{s^0}(w_{0,i}) \wedge \bigvee_{j=0}^{k} ([\alpha]_k^{[j,i]} \wedge \bigvee_{l \in \Gamma} H_l)), \\
[\overline{\mathrm{C}}_\Gamma \alpha]_k^{[m,n]} & := [\bigvee_{i=1}^{k} (\overline{\mathrm{E}}_\Gamma)^i \alpha]_k^{[m,n]}.
\end{array}$$

We now have the encoding.

Theorem 1. *Let M_d be a discretised interpreted system, and ψ an ECTLK_y formula. Then, $M_d \models \psi$ iff there exists $k \in \mathbb{N}_+$ such that $[\psi]_k^{0,0} \wedge [M^{\psi,s^0}]_k$ is satisfiable.*

5. RAILROAD CROSSING SYSTEM

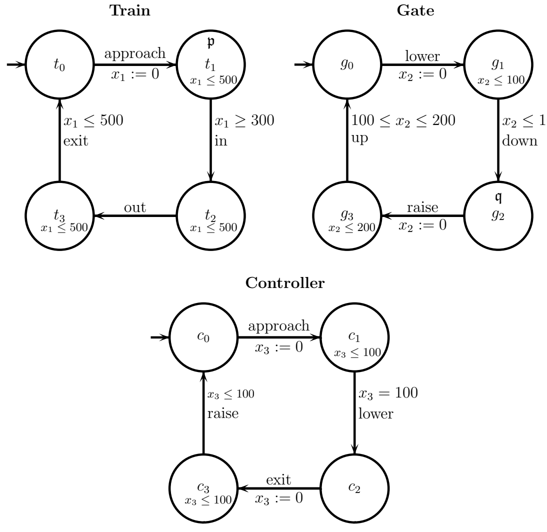

Figure 3: Timed Automata for Train, Gate, and Controller

The *railroad crossing system* (RCS) [13] is a well-known example in the literature of real-time verification. Here we not only check temporal properties but epistemic ones as well. The system consists of three agents, Train, Gate and Controller, as shown in Figure 3, which run in parallel and synchronize through the events: *approach, exit, lower* and *raise.* When a train approaches the crossing, Train sends an *approach* signal to Controller and enters the crossing between 300 and 500 seconds from this event. When Train leaves the crossing, it sends an *exit* signal to Controller. Controller sends a signal *lower* to Gate exactly 100 seconds after the *approach* signal is received, and sends a *raise* signal within 100 seconds after *exit.* Gate performs the transition *down* within 100 seconds of receiving the request *lower*, and responds to *raise* by moving *up* between 100 and 200 seconds.

We assume the following set of propositions: $\mathcal{PV} = \{\mathfrak{p}, \mathfrak{q}\}$ with $\mathcal{PV}_{Train} = \{\mathfrak{p}\}$, and $\mathcal{PV}_{Gate} = \{\mathfrak{q}\}$, and denote by L_1, L_2, L_3 sets of locations for Train, Gate, and Controller respectively. The valuation functions for Train ($\mathcal{V}_{Train}$), Gate ($\mathcal{V}_{Gate}$), and Controller ($\mathcal{V}_{Cont}$) are shown on Figure 3. The valuation function $\mathcal{V}_{RCS} : L_1 \times L_2 \times L_3 \rightarrow 2^{\mathcal{PV}}$ for the parallel composition, i.e., RCS system, is defined by $\mathcal{V}_{RCS}(l) = \mathcal{V}_{Train}(l_1) \cup \mathcal{V}_{Gate}(l_2) \cup \mathcal{V}_{Cont}(l_3)$, for all $l = (l_1, l_2, l_3) \in L_1 \times L_2 \times L_3$.

As an example, let us verify whether *there exists a behaviour of RCS such that agent Train considers possible a situation in which both it sends an approach signal and agent Gate does not send the signal down within 50 seconds.* This property can be formalised by the following TECTLK formula: $\varphi := \mathrm{EF}_{[0,\infty]}\overline{\mathrm{K}}_{Train}(\mathfrak{p} \wedge \mathrm{EF}_{[0,50]}(\neg\mathfrak{q}))$.

According to the BMC algorithm for TECTLK, presented in the previous section, to perform BMC for the RCS sys-

---

[5] In general we shall need to consider not just one but a number of symbolic k-paths. This number depends on the formula ψ under investigation, and it is returned as the value $f_k(\psi)$ of the function f_k.

[6] Recall that r is the number of the non-trivial intervals in φ, where $\psi = \mathrm{cr}(\varphi)$.

tem and property φ, first, all the states of the discretised interpreted system M_d for RCS with the additional clock y have to be represented by bit vectors. To do this we have to encode all the possible configurations in terms of both the locations, and the clock valuations of the RCS system.

Assume that we have the following bit representation of local locations; for Train we take $t_0 = (0,0)$, $t_1 = (0,1)$, $t_2 = (1,0)$, and $t_3 = (1,1)$, for Gate $g_0 = (0,0)$, $g_1 = (0,1)$, $g_2 = (1,0)$, and $g_3 = (1,1)$, and for Controller $c_0 = (0,0)$, $c_1 = (0,1)$, $c_2 = (1,0)$, and $c_3 = (1,1)$. So, the (global) locations of the RCS system have the following encoding: $t_1 \times g_0 \times c_1 = (0,1;0,0;0,1)$, $t_1 \times g_0 \times c_1 = (0,1;0,0;0,1)$, $t_1 \times g_1 \times c_2 = (0,1;0,1;1,0)$, etc. In other words we need 6 state variables $(\mathfrak{l}[0],\ldots,\mathfrak{l}[6])$ to encode all the possible configuration of locations of the RCS system.

In order to encode all the "important" clock valuations of RCS, we have to encode the valuations in $\mathbb{D} = \{k \cdot \Delta \mid 0 \leq k \cdot \Delta \leq 1002\}$ for the clocks: x_1, x_2, x_3, y by means of the discretisation step $\Delta = \frac{1}{8}$, and $c_{max}(\varphi) = 500$. Note that to do this, it is sufficient to encode the integral parts of the valuations, and the numerators of the fractional parts. It is easy to see that we need 13 state variables to encode all the clock valuations for one clock, and respectively $4 \cdot 13$ state variables $(\mathfrak{v}[0],\ldots,\mathfrak{v}[51])$ to encode all the clock valuations for 4 clocks. So, the global state variable for the RCS system is the following: $w = ((\mathfrak{l}[0],\ldots,\mathfrak{l}[5]),(\mathfrak{v}[0],\ldots,\mathfrak{v}[51])) = (w[0],\ldots,w[57])$.

In so doing, the transition relation of M_d has to be encoded by a Boolean formula, and $cr(\varphi) = \mathrm{X}_y(\mathrm{EF}\ (\overline{\mathrm{K}}_{Train}(\mathfrak{p} \wedge \mathrm{X}_y(\mathrm{EF}(\neg\mathfrak{q} \wedge p_{y\in[0,50]} \wedge (p_b \vee \top))) \wedge (p_b \vee \top)))$ has to be translated over all the possible $f_k(cr(\varphi)) = 3$ submodels of M_d.

To proceed with the translation of the transition relation of M_d, the first thing we need to translate is the initial state $s^0 = ((t_0,g_0,c_0),v^0)$ of RCS, where s^0 is represented by the binary of 58 0's. With the representation above this will be encoded by the propositional formula $I_{s^0}(w_{0,0}) = \bigwedge_{i=0}^{57} \neg w_{0,0}[i]$. The next step is to translate the transitions $\mathcal{R}(w_{i,j}, w_{i+1,j})$, for $i = 0,1,2$ and $j = 1,2,3$. For simplicity we report only on the formula $\mathcal{R}(w_{0,1}, w_{1,1})$ representing the first transition of the first path[7]. Let us consider the transition $[(t_0,g_0,c_0),(0,0,0,0)] \xrightarrow{\tau} [(t_0,g_0,c_0),(\frac{1}{4},\frac{1}{4},\frac{1}{4},\frac{1}{4})]$ of our counterexample. The corresponding formula is:

$$\begin{gathered}\mathcal{R}(w_{0,1}, w_{1,1}) := \bigwedge_{i=0}^{5}(\neg w_{0,1}[i] \wedge \neg w_{1,1}[i]) \wedge \bigwedge_{i=6}^{57} \neg w_{0,1} \\ \wedge \bigwedge_{i=6}^{15} \neg w_{1,1} \wedge \bigwedge_{i=16}^{18} w_{1,1} \wedge \bigwedge_{i=19}^{28} \neg w_{1,1} \\ \wedge \bigwedge_{i=29}^{31} w_{1,1} \wedge \bigwedge_{i=32}^{41} \neg w_{1,1} \wedge \bigwedge_{i=42}^{44} \neg w_{1,1} \wedge \bigwedge_{i=45}^{54} \neg w_{1,1} \\ \wedge \bigwedge_{i=55}^{57} \neg w_{1,1}.\end{gathered}$$

In order to encode the whole example we should model all the transitions for all the k's starting from $k := 1$. We do not do it here.

To encode the translation of $cr(\varphi)$, first we need to encode the propositions used in $cr(\varphi)$. This is $\mathfrak{p}(w) := (\neg w[0] \wedge w[1])$, which means that $\mathfrak{p}$ holds at all the global states with the first local locations equal to $(0,1)$, and $\mathfrak{q}(w) := (w[4] \wedge \neg w[5])$, which means that $\mathfrak{q}$ holds at all the global states with the third local locations equal to $(1,0)$. To encode $p_b(w)$, we have to encode all the clock valuations v of RCS such that $\forall \delta \in \{k\Delta \mid 0 < k\Delta < 1\}$ it is not the case that $(v \simeq v + \delta)$. Note that this condition holds if at least one of the clocks is in integer, i.e., its fractional part is equal to zero. So,

$$p_b(w) := (\neg w[10] \wedge \neg w[11] \wedge \neg[12]) \vee (\neg w[23] \wedge \neg w[24] \wedge \neg[25]) \vee (\neg w[35] \wedge \neg w[36] \wedge \neg[37]) \vee (\neg w[49] \wedge \neg w[50] \wedge \neg[51]).$$

To give the translation of the proposition $p_{y\in[0,50]}(w)$, assume that the following definition of propositional formulas are given. For the vectors of state variables $a = (a[1],\ldots,a[t])$ and $b = (b[1],\ldots,b[t])$ we define:

- $eq(a,b) \overset{def}{=} \bigwedge_{i=1}^{t} a[i] \Leftrightarrow b[i]$,
- $ge(a,b) \overset{def}{=} \bigvee_{i=1}^{t} \big(a[i] \wedge \neg b[i] \wedge \bigwedge_{j=i+1}^{t} a[j] \Leftrightarrow b[j]\big)$,
- $geq(a,b) \overset{def}{=} eq(a,b) \vee ge(a,b)$,
- $le(a,b) \overset{def}{=} \neg geq(a,b)$.

Then, for $\mathbf{0} := (0,0,0,0,0,0,0,0,0,0,0)$, and $\mathbf{50} := (0,0,0,0,1,1,0,0,1,0)$, we define $p_{y\in[0,50]}(w)$ as follows:

$$\begin{gathered}p_{y\in[0,50]}(w) \overset{def}{=} geq((w[44],\ldots,w[54]),\mathbf{0}) \wedge \\ [le((w[44],\ldots,w[54]),\mathbf{50}) \vee (eq((w[44],\ldots,w[54]),\mathbf{50}) \wedge \\ \bigwedge_{i=55}^{57} \neg w[i])]\end{gathered}$$

In so doing, it is sufficient to unfold the formula $[cr(\varphi)]_k^{0,0}$, for $k = 1,2,\ldots$, according to the definition on page .

Checking that the RCS system satisfies the TECTLK formula above can now be done by feeding a SAT solver with the propositional formula generated by this method. This would produce a solution, thereby proving that the propositional formula is satisfiable.

6. RELATED WORK AND CONCLUSIONS

BMC was initially developed for the verification of reactive systems, and then extended for real time systems [3, 19, 27] and MAS [14, 16, 26]. In particular, BMC has been extended to ACTL* [24], TACTL [19], and ACTLKD [26]. In this paper we have tried to combine these directions and have applied BMC to a new logic that combines real time and knowledge. In addition, there is no obstacle to extend the method presented here to handle an extension of the logic that includes operators representing correct functioning behaviour. We can do this in the same way as in [26].

The main point that we wished to bring across here is that the line of work of the past few years that defines model checking on combinations of discrete time temporal logics (like LTL or CTL) with epistemic operators can be extended to real time. Combinations of real time and knowledge have been defined previously [7, 15] but to our knowledge no verification mechanism has ever been defined onn them. To solve the difficulty of dense time, we have employed discretisation on equal intervals, already employed in [19, 27]. It is worth noting that intervals with length could be also used in principle. To do so one would have to encode more information (the maximum value of each clock, different lengths of bit-vectors that encode the integral parts of values of the clock, etc.), and as a result any implementation of the method would suffer in terms of speed.

Like every SAT-based approach the size of formulas produced in the translation can be large, as the example of the paper demonstrates. To evaluate its effectiveness in practical application, we are currently investigating implementing the method and performing experimental results. We are

---

[7] Note that the counterexample for the tested property can be found on the depth $k = 2$, i.e., it is of the following form: $[(t_0,g_0,c_0),(0,0,0,0)] \xrightarrow{\tau} [(t_0,g_0,c_0),(\frac{1}{4},\frac{1}{4},\frac{1}{4},\frac{1}{4})] \xrightarrow{approach} [(t_1,g_0,c_1),(\frac{1}{4},\frac{1}{4},\frac{1}{4},\frac{1}{4})]$. So, one symbolic $k-$path is sufficient to check validity of φ over the discretised model for RCS, although the value of $f_k(\varphi)$ is 3.

encouraged that implementations of other BMC-based tools [14, 19, 18] showed largely positive results. We are therefore hopeful that the technique of this paper, once implemented, will produce comparably fast results.

7. 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. Automata for modelling real-time systems. In *Proc. of ICALP'90*, volume 443 of *LNCS*, pp. 322–335. Springer-Verlag, 1990.

[3] G. Audemard, A. Cimatti, A. Kornilowicz, and R. Sebastiani. Bounded model checking for timed systems. In *Proc. of FORTE'02*, volume 2529 of *LNCS*, pp. 243–259. Springer-Verlag, 2002.

[4] A. Biere, A. Cimatti, E. Clarke, M.Fujita, and Y. Zhu. Symbolic model checking using SAT procedures instead of BDDs. In *Proc. of DAC'99*, pp. 317–320, 1999.

[5] P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*, volume 53 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2001.

[6] R. H. Bordini, M. Fisher, C. Pardavila, and M. Wooldridge. Model checking agentspeak. In *Proc. of AAMAS'03*, pp. 409–416. ACM Press, 2003.

[7] R. I. Brafman, J. C. Latombe, Y. Moses, and Y. Shoham. Application of a logic of knowledge to motion planning under uncertainty. *Journal of ACM*, 44(5), 1997.

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

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

[10] A. Göllü and A. Puri and P. Varaiya. *Discretization of Timed Automata*. In Proc. of CDC'94, pp. 957–958, 1994.

[11] J. Y. Halpern and M. Y. Vardi. The complexity of reasoning about knowledge and time 1: lower bounds. *Journal of Computer and System Sciences*, 38(1):195–237, 1989.

[12] W. van der Hoek and M. Wooldridge. Model checking knowledge and time. In *Proc. of SPIN'02*, 2002.

[13] I. Kang and I. Lee. An efficient state space generation for analysis of real-time systems. In *Proc. of ISSTA '96*, pp. 4–13. ACM Press, 1996.

[14] A. Lomuscio, T. Łasica, and W. Penczek. Bounded model checking for interpreted systems: preliminary experimental results. In *Proc. of FAABS II*, volume 2699 of *LNCS*. Springer Verlag, 2003.

[15] Y. Moses and B. Bloom. Knowledge, timed precedence and clocks. In *Proc. of PODC '94*, pp. 274–303. ACM Press, 1994.

[16] W. Penczek and A. Lomuscio. Verifying epistemic properties of multi-agent systems via bounded model checking. *Fundamenta Informaticae*, 55(2):167–185, 2003.

[17] W. Penczek and A. Pólrola. Specification and model checking of temporal properties in time Petri nets and timed automata. In *Proc. of ATPN'04*, volume 3099 of *LNCS*, pp. 37–76. Springer-Verlag, 2004.

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

[19] W. Penczek, B. Woźna, and A. Zbrzezny. Towards bounded model checking for the universal fragment of TCTL. In *Proc. of FTRTFT'02*, volume 2469 of *LNCS*, pp. 265–288. Springer-Verlag, 2002.

[20] F. Raimondi and A. Lomuscio. Verification of multiagent systems via ordered binary decision diagrams: an algorithm and its implementation. In *Proc. of AAMAS'04*, volume II. ACM, July 2004.

[21] S. Tripakis and S. Yovine. Analysis of timed systems using time-abstracting bisimulations. *Formal Methods in System Design*, 18(1):25–68, 2001.

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

[23] R. van der Meyden and H. Shilov. Model checking knowledge and time in systems with perfect recall. In *Proc. of FST&TCS'99*, volume 1738 of *LNCS*, pp. 432–445. Springer-Verlag, 1999.

[24] B. Woźna. Bounded Model Checking for the universal fragment of CTL*. *Fundamenta Informaticae*, 63(1):65–87, 2004.

[25] B. Woźna and A. Lomuscio. A logic for knowledge, correctness, and real time. In *Proc. of CLIMA'04*, LNAI. Springer-Verlag, 2005. To appear.

[26] B. Woźna, A. Lomuscio, and W. Penczek. Bounded model checking for deontic interpreted systems. In *Proc. of LCMAS'04*, volume 126 of *ENTCS*, pp. 93–114. Elsevier, 2004.

[27] A. Zbrzezny. Improvements in SAT-based reachability analysis for timed automata. *Fundamenta Informaticae*, 60(1-4):417–434, 2004.