Bounded Model Checking for Knowledge and Real Time

Alessio Lomuscio [a,1] Wojciech Penczek [b,2] Bożena Woźna [c,3]

[a] *Department of Computing, Imperial College London*
180 Queen's Gate, London SW7 2BZ, United Kingdom
e-mail: A.Lomuscio@imperial.ac.uk

[b] *Institute of Computer Science, PAS, Ordona 21, 01-237 Warsaw, and*
Institute of Informatics, Podlasie Academy, Sienkiewicza 51, Siedlce, Poland
e-mail: penczek@ipipan.waw.pl

[c] *Institute of Mathematics and Computer Science, Jan Dlugosz University*
Armii Krajowej 13/15, 42-200 Częstochowa, Poland
e-mail: b.wozna@ajd.czest.pl

Abstract

We present TECTLK, a logic to specify knowledge and real time in multi-agent systems. We show that the TECTLK model checking problem is decidable, and we present an algorithm for 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.

Key words: Temporal epistemic logics, model checking, interpreted systems, real time systems.

[1] The author acknowledges partial support from the EPSRC (grant GR/S49353).
[2] The author acknowledges partial support from the Royal Society (grant ESEP 2004/R3-EU).
[3] The research presented here was conducted while B. Woźna was supported by EPSRC (grant GR/S49353). The author also acknowledges partial support from the Ministry of Science and Information Society Technologies under grant number 3 T11C 011 28.

1 Introduction

Reasoning about knowledge [9] has always been a core concern in artificial intelligence. This is hardly surprising given that knowledge is a key concept to model intelligent, rational activities, human or artificial. A plethora of formalisms have been proposed and refined over the years, many of them based on formal logic. One of the most widely studied is based on variants of modal logics and is commonly referred to as temporal epistemic logic [9]. Rather than providing a computational engine for artificial agents' reasoning, epistemic logic, at least in this line, is seen as a specification language for modelling and reasoning about systems, much in common with formal methods in computer science. Formal properties of the logics such as completeness, decidability and complexity have been explored [13, 12, 20, 10].

Specification languages are most useful when they can be verified automatically. In this effort both theorem proving and model checking techniques as well as tools for epistemic logic have been developed. In the model checking approach the question of whether or not a system of agents S satisfies a property P is tackled by trying to establish whether or not $M_S \models \phi_P$, where M_S is a suitable model for S and ϕ_P is an appropriate logical formula representing P; we refer to [8] for more details.

In particular, for what concerns temporal epistemic logic, model checking techniques based on BDD [26, 29], bounded model checking [23], unbounded model checking [16] have been developed and their implementation either publicly released [26, 11] or made available via a web-interface [22].

While, one could now argue that verification via model checking of temporal epistemic logic has now become of age, in many respects the area is still lacking support for many essential functionalities. One of these is *real-time*. While the formalisms above deal with discrete sequence of events, it is often of both theoretical and practical interest to refer to a temporal model that assumes a dense sequence of events and uses operators able to represent dense temporal intervals. The aim of this work is to make a first step in this direction. In particular, recent contributions have focused on extending model checking techniques and tools [14, 23, 25, 26, 28, 32], to adapt them to the needs of multi-agent systems (MAS) formalisms [6, 9, 14, 15].

Specifically, we make two contributions: first we present a logic, that we call TECTLK, to reason about real time and knowledge in MAS; second, we present a bounded model checking technique for verifying automatically properties of multi-agent systems expressed in this logic.

The rest of the paper is organised as follows. The next section defines Real Time Interpreted Systems, the semantics on which we work with throughout the paper. In Section 3 the logic TECTLK is introduced. Section 4 deals with the discretisation process necessary for the bounded model checking algorithm, discussed in Section 5. Section 6 shows how this method can be applied to the "railroad crossing system", a typical multi-agent system example of time dependent systems. We conclude in Section 7 by discussing some related work.

2 Real Time Interpreted Systems

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

2.1 Timed Automata

Let $\mathbb{R} = [0, \infty)$ be a set of non-negative real numbers, $\mathbb{R}_+ = (0, \infty)$ 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:

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

The set of all the clock constraints over $\mathcal{X}$ is denoted by $\mathcal{C}(\mathcal{X})$. Note that inequalities involving differences of clocks are not in $\mathcal{C}(\mathcal{X})$.

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 $x \in \mathcal{X}$, $v'(x) = v(x) + \delta$. For a subset of clocks $X \subseteq \mathcal{X}$, $v[X := 0]$ denotes the valuation v' such that $v'(x) = 0$ for all $x \in X$, and $v'(x) = v(x)$ for all $x \in \mathcal{X} \backslash X$. 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:

$v \models true,$
$v \models (x \sim c)$ iff $v(x) \sim c,$
$v \models (\mathfrak{cc} \wedge \mathfrak{cc}')$ iff $v \models \mathfrak{cc}$ iff $v \models \mathfrak{cc}'.$

For a clock 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, and* $\mathfrak{I} : L \to \mathcal{C}(\mathcal{X})$ *is a* location invariant *function, assigning to each location* $l \in L$ *a clock constraint defining the conditions under which* $\mathcal{TA}$ *may stay in* l.

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

The clocks of a timed automaton are used to express its timing conditions. We differentiate between enabling conditions and invariant conditions. An enabling condition is a temporal constraint which must be satisfied for the transition to occur. An invariant condition $\mathfrak{I}(l)$ specifies the temporal constraint that must be satisfied for the automaton to remain in l.

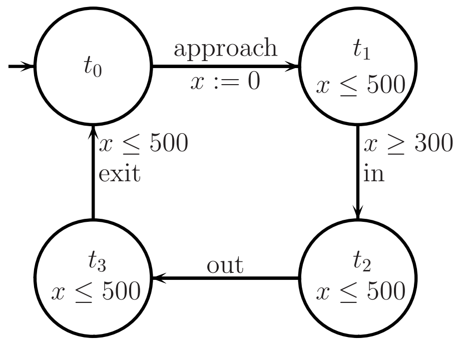


Fig. 1. A Timed Automaton.

Example 1 Figure 1 shows a timed automaton consisting of four locations: t_0, t_1, t_2, and t_3, where t_0 is the initial location, one clock x, the set of actions $\mathfrak{Z} = \{approach, in, out, exit\}$, and the following transitions: $t_0 \xrightarrow{approach,true,\{x\}} t_1$, $t_1 \xrightarrow{in,x\geq 300,\emptyset} t_2$, $t_2 \xrightarrow{out,true,\emptyset} t_3$, and $t_3 \xrightarrow{exit,x\leq 500,\emptyset} t_0$. The invariant of the location t_0 is $true$, whereas all the others locations are labelled with the invariant $x \leq 500$. Intuitively, the example models a system starting from t_0 and moving to t_1 by the action "*approach*" thereby causing the clock to be reset. The automaton must then execute the action "*in*" between the clock values of 300 and 500, thereby reaching location t_2. From t_2 the action "*out*" must be performed before the clock reaches the value of 500 resulting in t_3. From t_3 the action "*exit*" must be performed before the clock reaches the value of 500 resulting in t_0. Note that the enabling condition in t_3 is in this case redundant.

We take a timed-automaton as a fine-grained model of a real-time agent. A (real-time) multi-agent system will be defined as a set of communicating timed automata combined via parallel composition into a global timed automaton. In the composition the transitions not corresponding to a shared action are interleaved, whereas the transitions labelled with a shared action are synchronised. Several definitions of parallel composition exist. Here we use *multi-way synchronisation* [27], i.e., we require that each component with a communication transition (labelled by a shared action) has to perform this action when the global transition occurs.

Formally, 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 $i = 1, \ldots, m$,

$L_i \cap L_j = \emptyset$ for all $i, j \in \{1, \ldots, m\}$ and $i \neq j$, and let $\mathfrak{Z}(a) = \{1 \leq i \leq m \mid a \in \mathfrak{Z}_i\}$ denote the set of indices of the timed automata whose sets of actions contain the action a. The parallel composition is defined as follows.

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 $\mathfrak{Z} = \bigcup_{i=1}^m \mathfrak{Z}_i$, $L = \prod_{i=1}^m L_i$, $l^0 = (l_1^0, \ldots, l_m^0)$, $\mathcal{X} = \bigcup_{i=1}^m \mathcal{X}_i$, $\mathfrak{I}(l_1, \ldots, l_m) = \bigwedge_{i=1}^m \mathfrak{I}_i(l_i)$. Each global transition is such that*

$$((l_1, \ldots, l_m), a, \mathfrak{cc}, X, (l'_1, \ldots, l'_m)) \in E \textit{ iff } (\forall i \in \mathfrak{Z}(a))(l_i, a, \mathfrak{cc}_i, X_i, l'_i) \in E_i,$$
$$\mathfrak{cc} = \bigwedge_{i \in \mathfrak{Z}(a)} \mathfrak{cc}_i,\ X = \bigcup_{i \in \mathfrak{Z}(a)} X_i, \textit{ and } (\forall j \in \{1, \ldots, m\} \setminus \mathfrak{Z}(a))\ l'_j = l_j.$$

Note that the agents for which no communication action is available remain in the same location when this synchronisation action is performed.

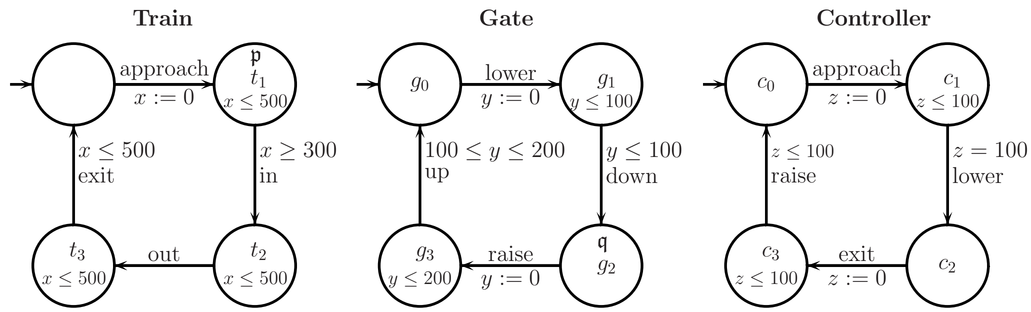


Fig. 2. Timed Automata for Train, Gate, and Controller.

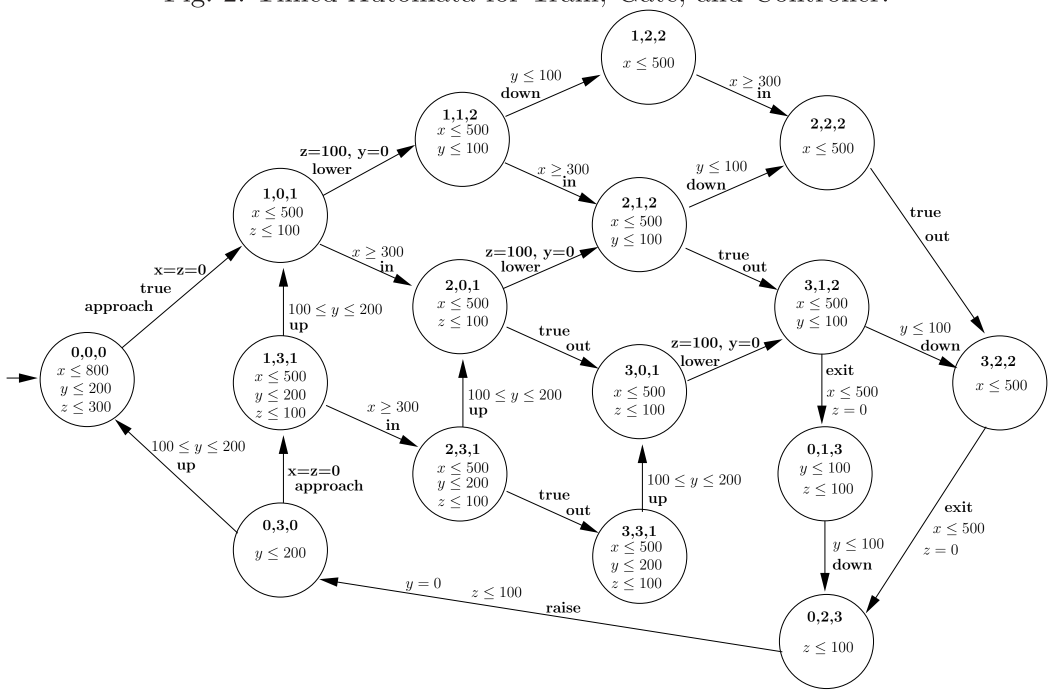


Fig. 3. The parallel composition of Train, Gate, and Controller.

Example 2 As an example of parallel composition let us consider the well-known *railroad crossing system* (RCS) [17]. The system consists of three timed automata: *Train*, *Gate* and *Controller*, as shown in Figure 2. The automaton Train is modelled via the timed automaton considered in Example 1. The automaton Gate consists of four locations: g_0, g_1, g_2, and g_3, one initial location g_0, one clock y, the set of actions $\mathfrak{Z} = \{lower, down, raise, up\}$, and the following transitions: $g_0 \xrightarrow{lower,true,\{y\}} g_1$, $g_1 \xrightarrow{down,y\leq 100,\emptyset} g_2$, $g_2 \xrightarrow{raise,true,\{y\}} g_3$, $g_3 \xrightarrow{up,100\leq y\leq 200,\emptyset} g_0$. The invariant of the locations g_0 and g_2 is *true*, whereas the locations g_1 and g_3 are labelled with the invariant $y \leq 100$ and $y \leq 200$, respectively. The automaton Controller consists of four locations: c_0, c_1, c_2, and c_3, one initial location c_0, one clock z, the set of actions $\mathfrak{Z} = \{approach, lower, exit, raise\}$, and the following transitions: $c_0 \xrightarrow{approach,true,\{z\}} c_1$, $c_1 \xrightarrow{lower,z=100,\emptyset} c_2$, $c_2 \xrightarrow{exit,true,\{z\}} c_3$, $c_3 \xrightarrow{raise,z\leq 100,\emptyset} c_0$. The invariant of the locations c_0 and c_2 is *true*, whereas the locations c_1 and c_3 are labelled with the invariant $z \leq 100$.

The automata Train, Gate, and Controller synchronise through the actions: *approach, exit, lower* and *raise*, and their parallel composition (known as the RCS system) is shown in Figure 3. The locations of RCS are given by triples (i, j, k) whose elements represent that Train, Gate, and Controller are at locations t_i, g_j and c_k, for $i, j, k \in \{0, 1, 2, 3\}$, respectively. The initial location of RCS is represented by the triple $(0, 0, 0)$, whereas the invariants of all the locations of RCS are the conjunction of the invariants of the three components.

2.2 Timed Automata

We use timed automata to interpret a logical language for real time and knowledge.

Let $\mathcal{TA} = (\mathfrak{Z}, L, l^0, E, \mathrm{X}, \mathfrak{I})$ be a timed automaton. An *instantaneous state* of $\mathcal{TA}$ is a pair (l, v), where $l \in L$ and $v \in \mathbb{R}^{|\mathcal{X}|}$.

Definition 3 *The* dense state space *of $\mathcal{TA}$ is a tuple $(L \times \mathbb{R}^{|\mathcal{X}|}, q^0, \rightarrow)$, where $L \times \mathbb{R}^{|\mathcal{X}|}$ is a set of all the instantaneous 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 $\rightarrow \subseteq (L \times \mathbb{R}^{|\mathcal{X}|}) \times (\mathfrak{Z} \cup \mathbb{R}) \times (L \times \mathbb{R}^{|\mathcal{X}|})$ is the transition relation, defined by:*

- *Action transition: 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 $v' \in [\![\mathfrak{I}(l')]\!]$,*
- *Time transition: for $\delta \in \mathbb{R}$, $(l, v) \xrightarrow{\delta} (l, v + \delta)$ iff v, $v + \delta \in [\![\mathfrak{I}(l)]\!]$.*

Intuitively, an action transition corresponds to an action performed by the automaton under consideration. Following this, its location changes accordingly, and all the clocks that are associated with the action are set to zero (i.e., the ones which belong to the set $X \subseteq \mathcal{X}$). Obviously, the action can be performed only if the underling enabling condition is satisfied. A time transition does not involve a location change, but an equal increase in the value of all the clocks, provided that the new clock valuations still satisfy all the location invariants.

For $(l, v) \in L \times \mathbb{R}^{|\mathcal{X}|}$, let $(l, v) + \delta$ denote $(l, v + \delta)$. A q_0-*run* ρ of $\mathcal{TA}$ is a finite or infinite sequence of instantaneous states:

$$q_0 \xrightarrow{\delta_0} q_0 + \delta_0 \xrightarrow{a_0} q_1 \xrightarrow{\delta_1} q_1 + \delta_1 \xrightarrow{a_1} q_2 \xrightarrow{\delta_2} \ldots$$

such that $q_i \in L \times \mathbb{R}^{|\mathcal{X}|}$, $a_i \in \mathfrak{Z}$, $\delta_0 \geq 0$, and $\delta_i \in \mathbb{R}_+$ for each $i \in \mathbb{N} \setminus \{0\}$. For the q^0-runs we require that $\delta_0 \in \mathbb{R}_+$. In other words, a run is a finite or infinite path of $\mathcal{TA}$, where action transitions are taken (in)finitely often and time transitions are aggregated. Notice that the semantics does not permit two consecutive action transitions to be performed one after the other, i.e., between each two action transitions some time must pass. This is a convenient way of representing a series of events to be taken in a continuous time.

Example 3 Given the automaton shown in Figure 3, let (l, v) be an instantaneous state of the automaton such that $l = (i, j, k)$ for $i \in \{0, 1, 2, 3\}$ and $v = (v(x), v(y), v(z))$. One of the possible q^0-runs is the following: $((0,0,0)(0,0,0)) \xrightarrow{50} ((0,0,0),(50,50,50)) \xrightarrow{approach} ((1,0,1),(0,50,0)) \xrightarrow{100} ((1,0,1),(100,150,100)) \xrightarrow{lower} ((1,1,2),(100,0,100)) \xrightarrow{30.5} \ldots$.

In line with much of the literature of the area we make the assumption that agents run continuously without termination. In a real-time context this requirement is normally expressed by distinguishing between *discrete progress* and *time progress*. Under discrete progress we allow for action transitions to happen infinitely often, that is, no instantaneous state occurs without action successors. Under time progress one assumes that time may pass without an upper bound; this is usually formalised by the notion of *non-zeno* runs.

Formally, an infinite run ρ is said to be *non-zeno* iff $\Sigma_{i \in \mathbb{N}} \delta_i$ is unbounded. An infinite run ρ is said to be *zeno* iff $\Sigma_{i \in \mathbb{N}} \delta_i$ is bounded by some real value. As an example, consider the automaton shown in Figure 4. Its q_0-run $(q_0, 0) \xrightarrow{1} (q_0, 1) \xrightarrow{a} (q_0, 1) \xrightarrow{0.5} (q_0, 1.5) \xrightarrow{a} (q_0, 1.5) \xrightarrow{0.25} (q_0, 1.75) \xrightarrow{a} (q_0, 1.75) \xrightarrow{0.125} (q_0, 1.875) \xrightarrow{a} (q_0, 1.875) \xrightarrow{0.0625} \ldots$ is zeno. On the other hand the following q_0-run $(q_0, 0) \xrightarrow{1} (q_0, 1) \xrightarrow{b} (q_1, 1) \xrightarrow{1} (q_1, 2) \xrightarrow{c} (q_1, 0) \xrightarrow{2} (q_1, 2) \xrightarrow{c} (q_1, 0) \xrightarrow{2} (q_1, 2) \xrightarrow{c} \ldots$ is non-zeno.

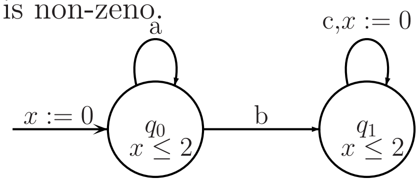


Fig. 4. An example of non-zeno and zeno runs.

We say that $\mathcal{TA}$ is *time-progressive* iff all its q^0-runs are non-zeno. For ease of presentation, we consider only time-progressive timed automata.

2.3 Real Time Interpreted Systems

We used timed-automata as a fine-grained semantics to reason about real time multi-agent systems. Technically we construct real-time traces generated by communicating automata upon which we interpret a temporal epistemic language. The standard (discrete time) semantics for temporal epistemic languages is the one of interpreted systems [9]. Here we introduce a real time version of them. First, in line with [1], we partition the set of clock valuations.

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 constraints occurring in all enabling conditions used in the transition relation E and all state invariants of $\mathcal{TA}$. Moreover, let c_{max} be the largest constant appearing in $\mathcal{C}(\mathcal{TA})$ and let $fr(\sigma)$ (respectively $\lfloor\sigma\rfloor$), $\sigma \in \mathbb{R}$, denote the fractional (respectively integral part) of σ. We define an equivalence relation $\simeq$ in the set of all the clock valuations as follows.

Definition 4 ([1]) *For two clock valuations $v, v' \in \mathbb{R}^{|\mathcal{X}|}$, $v \simeq v'$ if and only if the following conditions are met:*

1. *For all $x \in \mathcal{X}$, $v(x) > c_{max}$ iff $v'(x) > c_{max}$,*
2. *For all $x, y \in \mathcal{X}$, 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 $fr(v'(x)) \leq fr(v'(y))$.

In other words the valuations are equivalent if they return values greater than c_{max} for the same x and when their integral part is the same for any x, and the fractional parts are either both nil or preserve the order of any two clock values (see Figure 5 for an example).

The relation $\simeq$ partitions $\mathbb{R}^{|\mathcal{X}|}$ into *zones*, denoted by Z, Z', and so on. We will denote the set of all the zones by $Z(|\mathcal{X}|)$.

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 $i = 1, \ldots, m$, $\mathcal{TA} = (\mathfrak{Z}, L, l^0, E, \mathrm{X}, \mathfrak{I})$ the parallel composition of all the agents, and $l_i : L \to L_i$ be a function returning the location of agent i in a global location. Moreover, we take $\mathcal{PV}_i$ to be a set of propositional variables containing the constant *true* (denoted by $\top$) such that $\mathcal{PV}_i \cap \mathcal{PV}_j = \emptyset$ for all $i, j \in \{1, \ldots, m\}$, 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, we assume the existence of a (local) valuation function

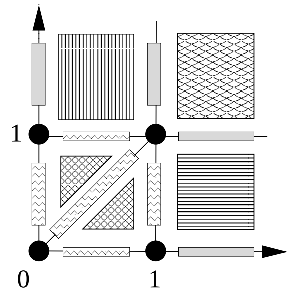


Fig. 5. Equivalence of clock valuations for two clocks with $c_{max} = 1$.

$\mathcal{V}_{\mathcal{TA}_i} : L_i \rightarrow 2^{\mathcal{PV}_i}$ for each agent i. We further require that $\top \in \mathcal{V}_{\mathcal{TA}_i}(l)$ for each $l \in L_i$. The (global) valuation function $\mathcal{V}_{\mathcal{TA}} : L \rightarrow 2^{\mathcal{PV}}$ 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 5 (Real Time Interpreted System) *A* real time interpreted system *is a tuple $M = (Q, q^0, \rightarrow, \sim_1, \ldots, \sim_m, \mathcal{V})$, where:*

- *Q is a subset of $L \times \mathbb{R}^{|\mathcal{X}|}$ such that all the instantaneous states in Q are reachable*[4] *.*
- *q^0, and $\rightarrow$ are defined as in Definition 3.*
- *$\sim_i \subseteq Q \times Q$ is an (equivalence) relation defined by $(l, v) \sim_i (l', v')$ iff $l_i(l) = l_i(l')$ and $v \simeq v'$, for each agent i.*
- *$\mathcal{V} : Q \rightarrow 2^{\mathcal{PV}}$ is a valuation function such that $\mathcal{V}((l, v)) = \mathcal{V}_{\mathcal{TA}}(l)$.*

In line with [9] and related literature $\sim_i$ is an epistemic accessibility relation. Two states are related for agent i if, according to all the information the agent has available these two states cannot be distinguished; in other words the two states are locally identical for agent i. In (discrete time) interpreted systems the definition of $\sim_i$ is based on the equality of the local states for agent i in the two global states. The definition we propose here extends that by assuming that not only the local locations of the agents are the same, but also the two clock valuations are in the same zone. In other words we assume the zones of the clock valuations to be visible to agent i: if two states have the same location but differ in the clock zone the agent is able to distinguish them, and, consequently, the states will not be in the same equivalence class induced by $\sim_i$.

[4] An instantaneous state $q \in L \times \mathbb{R}^{|\mathcal{X}|}$ is *reachable* iff there is a q^0−run ρ in $\mathcal{TA}$ such that there exists an instantaneous state in ρ equal to q.

3 The Logic TECTLK

To reason about MAS, we introduce TECTLK, a logic for knowledge and real time that is the fusion [5] of the two underlying languages: an existential fragment of TCTL for branching real time [1] and $S5_n$ for the knowledge operators. Obviously, defining the fusion with the full TCTL would not be problematic [31], but we use here the fragment only because it is more suited for the model checking method that is defined later on in the paper.

3.1 Syntax

Let $\mathcal{PV}$ be a set of propositional variables containing the symbol $\top$ that represents the constant *true*, $\mathcal{AG}$ 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 *formulae* is defined by the following grammar:

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

As customary the formula $\mathrm{E}(\varphi \mathrm{U}_I \psi)$ is read as "there exists a computation in which φ holds until, in the interval I, ψ holds". R is the operator for "Release"; $\mathrm{E}(\varphi \mathrm{R}_I \psi)$ represents "there exists a computation in which either ψ holds until, in the interval I, both ψ and φ hold, or ψ always holds in the interval I". $\overline{\mathrm{K}}_i$ is the dual for the standard epistemic modality, so $\overline{\mathrm{K}}_i \varphi$ is read as "agent i considers φ as possible". Similarly, the modalities $\overline{\mathrm{D}}_\Gamma, \overline{\mathrm{E}}_\Gamma, \overline{\mathrm{C}}_\Gamma$ are the diamonds for $\mathrm{D}_\Gamma, \mathrm{E}_\Gamma, \mathrm{C}_\Gamma$ representing distributed knowledge in the group Γ, "everyone in Γ knows", and common knowledge among agents in Γ.

The other basic temporal modalities can be introduced as usual: $\mathrm{EG}_I \varphi \stackrel{def}{=} \mathrm{E}(\bot \mathrm{R}_I \varphi)$, and $\mathrm{EF}_I \varphi \stackrel{def}{=} \mathrm{E}(\top \mathrm{U}_I \varphi)$. Moreover, $\bot \stackrel{def}{=} \neg \top$, $\alpha \rightarrow \beta \stackrel{def}{=} \neg \alpha \vee \beta$, and $\alpha \leftrightarrow \beta \stackrel{def}{=} (\alpha \rightarrow \beta) \wedge (\beta \rightarrow \alpha)$.

3.2 Semantics

Let $\mathcal{AG}$ be a set of m agents such that each agent is modelled by a timed automaton $\mathcal{TA}_i = (\mathfrak{Z}_i, L_i, l_i^0, E_i, \mathcal{X}_i, \mathfrak{I}_i)$. Further, let $\mathcal{TA} = (\mathfrak{Z}, L, l^0, E, \mathrm{X}, \mathfrak{I})$ be the parallel composition of the agents and $f_{\mathcal{TA}}(q)$ denote the set of all q-runs for $\mathcal{TA}$, that is, the set of all the runs in $\mathcal{TA}$ that start at the state q. In order to give a semantics to TECTLK, we introduce the notion of a *dense path* π_ρ *corresponding to a* q_0*-run* $\rho = q_0 \stackrel{\delta_0}{\rightarrow} q_0 + \delta_0 \stackrel{a_0}{\rightarrow} q_1 \stackrel{\delta_1}{\rightarrow} q_1 + \delta_1 \stackrel{a_1}{\rightarrow} q_2 \stackrel{\delta_2}{\rightarrow} \ldots$. Let $idx(\rho, r)$ be the greatest $i \in \mathbb{N}$ such that $\Sigma_{j=0}^{i-1} \delta_j \leq r$. Notice that for $i = 0$ we let $\Sigma_{j=0}^{i-1} \delta_j = 0$. So, for $r \leq \delta_0$, $idx(\rho, r) = 0$. *A dense path* π_ρ *corresponding to*

ρ is a mapping from $\mathbb{R}$ to the set of states Q such that $\pi_\rho(r) = q_i + r - \Sigma_{j=0}^{i-1}\delta_j$ where $i = idx(\rho, r)$. This can be done in a unique way because we assume that runs of a $\mathcal{TA}$ do not contain two consecutive action transitions.

Moreover, for the group modalities we also, as customary, define the following. If $\Gamma \subseteq \mathcal{AG}$, then $\sim_\Gamma^E \stackrel{def}{=} \bigcup_{i\in\Gamma} \sim_i$, $\sim_\Gamma^C \stackrel{def}{=} (\sim_\Gamma^E)^+$ (the transitive closure of $\sim_\Gamma^E$), and $\sim_\Gamma^D \stackrel{def}{=} \bigcap_{i\in\Gamma} \sim_i$.

Definition 6 (Satisfaction) *Let $M = (Q, q^0, \rightarrow, \sim_1, \ldots, \sim_m, \mathcal{V})$ be a* real time interpreted system. *$M, q \models \alpha$ denotes that α is true at state q in M. M is omitted, if it is implicitly understood. The satisfaction relation $\models$ is defined inductively as follows:*

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

Some examples of TECTLK formulae holding at state q of a real time interpreted system are shown in Figure 6.

A TECTLK formula φ is *satisfiable* iff there exists a real time interpreted system $M = (Q, q^0, \rightarrow, \sim_1, \ldots, \sim_m, \mathcal{V})$ and an instantaneous state q of M such that $M, q \models \varphi$. A TECTLK formula φ is *valid on* M (denoted $M \models \varphi$) iff $M, q^0 \models \varphi$, i.e., φ is true at the initial state of M; we use the term *model checking problem* to denote the problem of checking validity of φ when M is given explicitly [5] .

Note that the "full" logic of branching real time, i.e., TCTL, is undecidable [1] (in the sense that its theoremhood problem is undecidable). Since real time

[5] Note that some authors have recently used the term "model checking problem" only to refer to situations when M is given *implicitly* by means of a dedicated (programming) language.

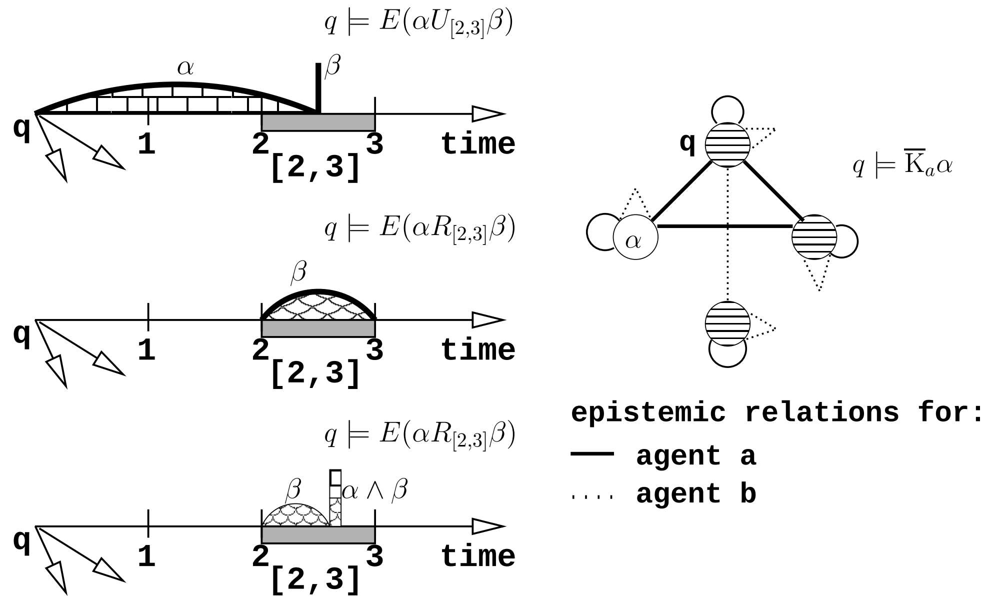


Fig. 6. Examples of TECTLK formulae which hold at state q of a real time interpreted system.

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 (see Lemma 1).

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

Proof: The correctness of the lemma follows from Lemma [correctness of the labelling algorithm] in [1] and Proposition 3.2.1 in [9]. □

4 Epistemic Region Graph and Its Discretisation

Any real time interpreted system is dense and hence infinite. To perform model checking efficiently, we consider an appropriately generated finite version of it. In particular we use an *epistemic region graph (ERG)*, defined as an extension of the *region graph* [1] augmented to include the relation $\sim_i$, for each agent $i \in \mathcal{AG}$.

Let $\mathcal{AG}$ be a set of m agents, where each agent is modelled via a timed automaton and $\mathcal{TA} = (\mathfrak{Z}, L, l^0, E, \mathcal{X}, \mathfrak{I})$ the parallel composition of them. *The epistemic region graph* for the timed automaton $\mathcal{TA}$ is a tuple

$$M_{rg} = (S, \iota, \rightarrow_{rg}, \sim_1^{rg}, \ldots, \sim_m^{rg}, \mathcal{V}_{rg})$$

where

- $S \subseteq L \times Z(|\mathcal{X}|)$ is a set of reachable states, called regions; note that each element of S is a pair (l, Z) where l is a location and Z is a zone.
- $\iota = (l^0, Z^0)$ is the initial region, where $Z^0 = \{v^0\}$; recall that $v^0(x) = 0$, for all $x \in \mathcal{X}$,
- $\rightarrow_{rg} \subseteq S \times (\mathfrak{Z} \cup \{\tau\}) \times S$ is defined by:
  · Time transition: $(l, Z) \xrightarrow{\tau}_{rg} (l, Z')$ iff there exist $v \in Z$ and $v' \in Z'$ such that
    a.) $(l, v) \xrightarrow{\delta} (l, v')$ for some $\delta \in \mathbb{R}_+$, and
    b.) if $(l, v) \xrightarrow{\delta'} (l, v'') \xrightarrow{\delta''} (l, v')$ and $(l, Z'') \in S$ for some Z'' such that $v'' \in Z''$, then either $v \simeq v''$ or $v'' \simeq v'$, and
    c.) if $v \simeq v'$, then $v \simeq v' + \delta''$ for each $\delta'' \in \mathbb{R}$.
  · Action transition: For any $a \in \mathfrak{Z}$, $(l, Z) \xrightarrow{a}_{rg} (l', Z')$ iff the following conditions hold:
    a.) (l, Z) is not boundary [6] and
    b.) either there exist $v \in Z$ and $v' \in Z'$ such that $(l, v) \xrightarrow{a} (l', v')$ or there exist Z'' and $v'' \in Z''$ such that $(l, Z) \xrightarrow{\tau}_{rg} (l, Z'')$ and $(l, v'') \xrightarrow{a} (l', v')$.
- $\sim_i^{rg} \subseteq S \times S$ is a relation defined by $(l, Z) \sim_i (l', Z')$ iff $l_i(l) = l_i(l')$ and $Z = Z'$, for each agent i. Obviously $\sim_i$ is an equivalence relation.
- $\mathcal{V}_{rg} : S \rightarrow 2^{\mathcal{PV}}$ is a valuation function that extends $\mathcal{V}_{\mathcal{TA}}$ as follows $\mathcal{V}_{rg}((l, Z)) = \mathcal{V}_{\mathcal{TA}}(l)$.

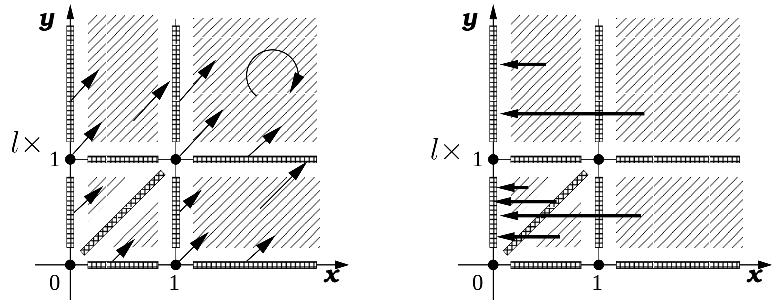
An illustration of the above definition of the action and time transition relation is shown in Figure 7.

The following lemma guarantees that the epistemic region graph preserves validity of TECTLK formulae.

Lemma 2 *Let $\mathcal{AG}$ be a finite set of agents modelled by timed automata, $\mathcal{TA} = (\mathfrak{Z}, L, l^0, E, \mathcal{X}, \mathfrak{I})$ their parallel composition, $V_{\mathcal{TA}}$ a valuation function for $\mathcal{TA}$, and M the real time interpreted system for $\mathcal{TA}$. Further, let $l \in L$, and v, $v' \in \mathbb{R}_+^{|\mathcal{X}|}$ with $v \simeq v'$. Then, for every* TECTLK *formula φ, $M, (l, v) \models \varphi$ iff $M, (l, v') \models \varphi$.*

Proof: The proof of the lemma follows directly from Lemma of equivalence of clock valuations ([1]) and the definition of the accessibility relation $\sim_i$ for

[6] A region (l, Z) is *boundary* if for each $\delta \in \mathbb{R}, v \in Z$, $\neg(v \simeq v + \delta)$.



Time transitions

Action transitions
by transition $l \xrightarrow{a,true,\{x\}} l$

Fig. 7. Time and action transitions in an epistemic region graph.

each agent. □

In Section 5 we define a bounded model checking (BMC) technique to verify TECTLK properties of real time interpreted systems. The BMC method relies on a symbolic encoding of the transition relations of the real time interpreted system under consideration as Boolean formulae. But, given Lemma 2, it is sufficient to define Boolean formulae that encode the transition relations of the epistemic region graph only. To perform this task we will discretise the state space by using a discretisation method described in [33] and shortly reported below.

4.1 Discretisation

Let $\mathcal{TA} = (\mathfrak{Z}, L, l^0, E, \mathcal{X}, \mathfrak{I})$ be a timed automaton, φ a TECTLK formula, and $c_{max}(\varphi)$ the largest constant appearing in $\mathcal{C}(\mathcal{TA})$ and in any interval of the temporal operators in φ. We choose $\Delta = 1/2^{\lceil log_2(2|\mathcal{X}|)\rceil}$ as the discretisation step[7], and we define a *discretised clock space* $\mathbb{D}^{|\mathcal{X}|}$ with

$$\mathbb{D} = \{k\Delta \mid 0 \leq k\Delta \leq 2c_{max}(\varphi) + 2,\ k \in \mathbb{N}\}.$$

Note that the clocks do not 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

$$\mathbb{E} = \{k\Delta \mid 0 \leq k\Delta < c_{max}(\varphi) + 1\}.$$

[7] A different discretisation step is also possible, but the one reported here is convenient for the model checking method described later on.

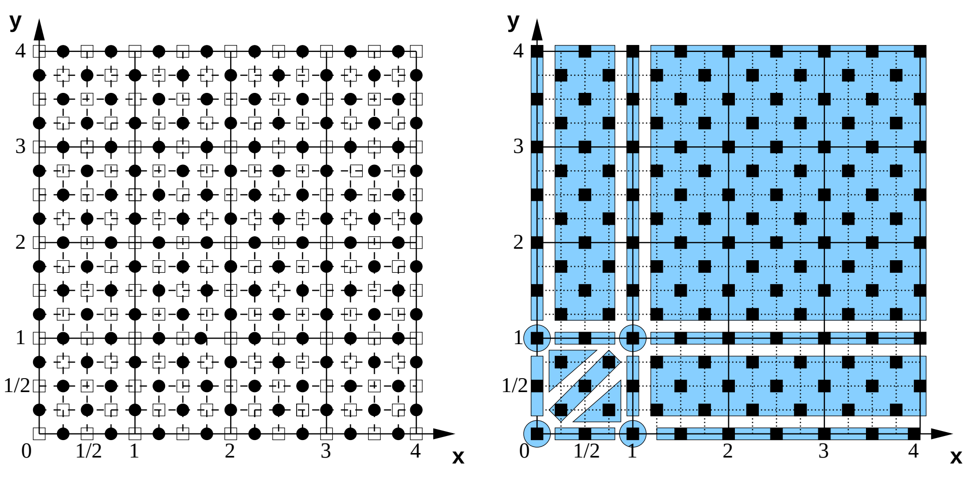
Next, we take a subset $\mathbb{U}^{|\mathcal{X}|}$ of $\mathbb{D}^{|\mathcal{X}|}$ that allows us to preserve the time transitions of the epistemic region graph by insisting that either the values of all the clocks in $v \in \mathbb{U}^{|\mathcal{X}|}$ are only even or only odd multiplications of Δ:

$$\mathbb{U}^{|\mathcal{X}|} = \{v \in \mathbb{D}^{|\mathcal{X}|} \mid (\forall x \in \mathcal{X})(\exists k \in \mathbb{N}) \text{ either } v(x) = 2k\Delta \text{ or } v(x) = (2k{+}1)\Delta\}.$$

To preserve action transitions of the epistemic region graph we use so called *adjust transitions* $\xrightarrow{\epsilon} \subseteq (L \times \mathbb{D}^{|\mathcal{X}|}) \times (L \times \mathbb{U}^{|\mathcal{X}|})$. The aim of these transitions is to replace points no longer in $\mathbb{U}^{|\mathcal{X}|}$ (after executing an action or time transition) by zone-equivalent points in $\mathbb{U}^{|\mathcal{X}|}$. Formally such adjust transitions are defined as follows. Let $(l,v),(l,v') \in (L \times \mathbb{D}^{|\mathcal{X}|})$. Then, $(l,v) \xrightarrow{\epsilon} (l,v')$ iff $v' \in \mathbb{U}^{|\mathcal{X}|}$, $(\forall x \in \mathcal{X})(v'(x) \leq c_{max}(\varphi)+1)$, and $v \simeq v'$.

Example 4 *Consider a timed automaton $\mathcal{TA}$ with two clocks x and y, and a* TECTLK *formula φ. Moreover, assume that $c_{max}(\varphi) = 1$. Figure 8 shows the discretised clock space $\mathbb{D}^2$ of $\mathcal{TA}$. The chosen discretisation step is $\Delta = \frac{1}{2^{\lceil log_2(2\cdot 2)\rceil}} = \frac{1}{4}$. Therefore,*

- $\mathbb{D} = \{0, \frac{1}{4}, \frac{2}{4}, \frac{3}{4}, 1, \ldots, 4\}$,
- $\mathbb{D}^2 = \{(0,0), (0,\frac{1}{4}), (0,\frac{2}{4}), \ldots, (0,4), (\frac{1}{4},0), (\frac{1}{4},\frac{1}{4}), (\frac{1}{4},\frac{2}{4}), \ldots, (\frac{1}{4},4), \ldots, (4,4)\}$,
- $\mathbb{U}^2 = \{(0,0), (0,\frac{2}{4}), (0,1), (0,1\frac{2}{4}), \ldots, (0,4), (\frac{1}{4},\frac{1}{4}), (\frac{1}{4},\frac{3}{4}), (\frac{1}{4},1\frac{1}{4}), \ldots, (\frac{1}{4},3\frac{3}{4}), (\frac{2}{4},\frac{2}{4}), (\frac{2}{4},1), \ldots, (\frac{2}{4},4), (\frac{3}{4},\frac{1}{4}), (\frac{3}{4},\frac{3}{4}), (\frac{3}{4},1\frac{1}{4}), \ldots, (\frac{3}{4},3\frac{3}{4}), \ldots, (4,4)\}$.



(A) (B)

Fig. 8. (A) Discretisation of $\mathbb{R}^2$ with $c_{max}(\varphi) = 1$; the elements of $\mathbb{D}^2$. Notice that both the black dots and the transparent rectangles are elements of $\mathbb{D}^2$, but only the transparent rectangles are elements of $\mathbb{U}^2$. (B) Zones of $\mathbb{R}^2$ with $c_{max}(\varphi) = 1$ and the elements of $\mathbb{U}^2$. The latter one are represented by black rectangles.

In this section, we define a *discretised interpreted system* and show that it enjoys the same property as the epistemic region graph M_{rg}, i.e., it preserves validity of TECTLK formulae.

Definition 7 (Discretised Interpreted System) *Let $\mathbb{E}_+$ denote the set $\mathbb{E} \setminus \{0\}$, and ; denote composition of two relations. A* discretised interpreted system *for the timed automaton $\mathcal{TA} = (\mathfrak{Z}, L, l^0, E, \mathcal{X}, \mathfrak{I})$ is a structure $M_d = (S_d, s^0, \rightarrow_d, \sim_1^d, \ldots, \sim_m^d, \mathcal{V}_d)$, where $S_d \subseteq L \times \mathbb{U}^{|\mathcal{X}|}$ is a set of reachable states, $s^0 = (l^0, v^0)$ is the initial state, and the relation $\rightarrow_d \subseteq S_d \times (\mathfrak{Z} \cup \{\tau\}) \times S_d$ is defined by:*

- *(Discrete) time transition: $(l, v) \xrightarrow{\tau}_d (l, v')$ iff $(l, v) \xrightarrow{\delta}; \xrightarrow{\epsilon} (l, v')$ for some $\delta \in \mathbb{E}_+$, 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 $\delta'' \in \mathbb{E}_+$.*
- *(Discrete) action transition: $(l, v) \xrightarrow{a}_d (l', v')$ iff (l, v) is not boundary* [8] *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 of the above, consider a region as a pair (l, Z) for a location $l \in L$ and a zone Z. A time transition relation represents a move to a region because of passage of time, but sharing the same location. In order to make sure that valuations of the clocks do not go beyond $2c_{max}(\varphi) + 2$, and that before taking any transition the value of every clock does not exceed $c_{max}(\varphi) + 1$, we adjust each time transition by an ϵ-move. An action transition represents a move by an action (adjusted by an ϵ-move in order to stay in $\mathbb{U}$) taken from a non-boundary region and possibly preceded by the time transition step. Note that an action transition cannot be taken from a boundary region to make sure that there are no two consecutive action transition steps in a run.

Lemma 3 (Discretisation Preserves Time Successor) *Let $\widetilde{Z} = Z \cap \mathbb{U}^{|\mathcal{X}|}$, for any zone $Z \in Z(|\mathcal{X}|)$. For every region (l, Z) and (l, Z'), if $(l, Z) \xrightarrow{\tau}_{rg} (l, Z')$, then there exist $v \in \widetilde{Z}, v' \in \widetilde{Z'}$ such that $(l, v) \xrightarrow{\tau}_d (l, v')$.*

Proof: The proof of the lemma follows directly from Lemmas 4.1 - 4.4 in [33]. □

Lemma 4 (Discretisation for Action Successor) *Let $\widetilde{Z} = Z \cap \mathbb{U}^{|\mathcal{X}|}$, for*

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

any zone $Z \in Z(|\mathcal{X}|)$*. For any* $a \in \mathfrak{Z}$ *and for every region* (l, Z) *and* (l, Z')*, if* $(l, Z) \xrightarrow{a}_{rg} (l, Z')$*, then there exists* $v \in \widetilde{Z}$ *and there exists* $v' \in \widetilde{Z'}$ *such that* $(l, v) \xrightarrow{a}_{d} (l, v')$*.*

Proof: The proof of the lemma follows directly from Lemma 4.2 in [33]. □

The reverse of Lemmas 4 and 3 also holds.

The following lemma guarantees that the discretised interpreted system preserves validity of the TECTLK formulae.

Lemma 5 *Let* $\mathcal{AG}$ *be a finite set of agents modelled by timed automata,* $\mathcal{TA} = (\mathfrak{Z}, L, l^0, E, \mathcal{X}, \mathfrak{I})$ *their parallel composition,* $V_{\mathcal{TA}}$ *a valuation function for* $\mathcal{TA}$*, and* M *the real time interpreted system for* $\mathcal{TA}$*,* $l \in L$*, and* $v \in \mathbb{R}_+^{|\mathcal{X}|}$*. Then, for every* TECTLK *formula* φ*,* $M, (l, v) \models \varphi$ *iff there exists* $v' \in \mathbb{U}^{|\mathcal{X}|}$ *such that* $v \simeq v'$ *and* $M, (l, v') \models \varphi$*.*

Proof: The proof of the lemma follows directly from Lemma 2, Lemma 3, and Lemma 4. □

5 TECTLK Bounded Model Checking

Bounded model checking (BMC) is a SAT-based technique for symbolic model checking. Compared to BDD-based model checking it offers the advantage of handling the verification of large state spaces, albeit for a smaller fragment of the language.

The main idea of BMC is to avoid the full state space generation and, instead, to look for witnesses of an existential specification on suitable subsets of the full model. Once a submodel is selected, the formula to be checked as well as the considered submodel are translated into propositional formulae and a propositional satisfiability problem is solved via specialised SAT solvers. If the test is positive, the specification holds on the submodel as well as on the whole model, given the particular existential syntax checked. If not, a larger submodel is selected and the whole procedure is run again.

Note that at times this procedure is used to find bugs on systems by attempting to find counter examples to universal formulas by checking their negations.

While this approach is not intrinsically more efficient than BDD-based approaches, in applications that it is often the case that faults can be identified on small fragments of a full model. In these cases BMC represents an extremely appealing alternative to more standard techniques. The efficiency of this approach has been experimentally demonstrated in, among others, [4, 18, 24, 25].

For the case of this paper, knowledge and real time, we extend the technique employed for TECTL [25] and ECTLK [23]. We first translate the model checking problem for TECTLK into the model checking problem of another logic, called ECTLK_y, and then we define BMC for ECTLK_y. Thanks to these translations the model checking problem on an infinite state space is translated into bounded model checking on a finite state space. Soundness and completeness of the translations is guaranteed by Theorem 1, Theorem 2, and Theorem 3 presented below.

5.1 *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 time-progressive timed automata only, to extend the procedure of [1] to TECTLK, we introduce a slightly different logic ECTLK_y, as presented below.

Let $\mathcal{AG}$ be a finite set of agents modelled by timed automata, $\mathcal{TA} = (\mathfrak{Z}, L, l^0, E, \mathcal{X}, \mathfrak{I})$ be their parallel composition, $\mathcal{V}_{\mathcal{TA}}$ a valuation function, and φ a TECTLK formula. First, we construct a new timed automaton $\mathcal{TA}_\varphi = (\mathfrak{Z}', L, l^0, E', \mathcal{X}', \mathfrak{I})$ by extending $\mathcal{TA}$ with: (1) a new clock y that corresponds to all the intervals appearing in φ, i.e., $\mathcal{X}' = \mathcal{X} \cup \{y\}$ [9]; (2) an action a_y, i.e., $\mathfrak{Z}' = \mathfrak{Z} \cup \{a_y\}$; (3) a set $E_y = \{(l, a_y, true, \{y\}, l) \mid l \in L\}$ of special transitions that are used to reset the new clock y, i.e., $E' = E \cup E_y$. These transitions are used to start the runs over which sub-formulae of φ are checked. We then extend the set of propositional variables $\mathcal{PV}$ to the set $\mathcal{PV}' = \mathcal{PV} \cup \{p_{y \in I} \mid I$ is an interval in $\varphi\} \cup \{p_b\}$, where $p_{y\in I}$ is a propositional variable true at the states where $y \in I$, and p_b is a propositional variable representing the fact that a state (region) is boundary. Further, we construct the discretised interpreted system for $\mathcal{TA}_\varphi$, and augment its valuation function with the set $\mathcal{PV}'$ of propositional variables. Finally, we translate the TECTLK formula φ into an ECTLK_y formula $\psi = \text{cr}(\varphi)$ such that model checking of φ over the discretised interpreted system for $\mathcal{TA}$ can be reduced to the model checking of ψ over the discretised interpreted system for $\mathcal{TA}_\varphi$.

In order to translate a TECTLK formula φ into the corresponding ECTLK_y formula ψ we map the ECTLK language into ECTLK_y by reinterpreting the temporal operators, denoted by E_yU and E_yR. This language is interpreted over the discretised interpreted system for $\mathcal{TA}_\varphi$. Formally, for $p \in \mathcal{PV}$, $i \in \mathcal{AG}$

[9] One clock is sufficient to perform the bounded model checking algorithm that is presented in the next section. Note other model checking methods may require one clock per interval appearing in the TECTLK formula under consideration.

and $\Gamma \subseteq \mathcal{AG}$, the set $\mathcal{WF}$ of ECTLK$_y$ formulae is defined by the following grammar:

$$\alpha := p \mid \neg p \mid \alpha \wedge \alpha \mid \alpha \vee \alpha \mid \mathrm{E}_y(\alpha \mathrm{U} \alpha) \mid \mathrm{E}_y(\alpha \mathrm{R} \alpha) \mid \overline{\mathrm{K}}_i\alpha \mid \overline{\mathrm{D}}_\Gamma\alpha \mid \overline{\mathrm{C}}_\Gamma\alpha \mid \overline{\mathrm{E}}_\Gamma\alpha.$$

Let $M_d = (S_d, s^0, \rightarrow_d, \sim_1^d, \ldots, \sim_m^d, \mathcal{V}_d)$ be a discretised interpreted system for $\mathcal{TA}_\varphi$ such that the set S_d contains reachable states only, $s \in S_d$, α, β formulae of ECTLK$_y$, $\rightarrow_{\mathcal{TA}}$ denote the part of $\rightarrow_d$, where transitions are labelled with elements of $\mathfrak{Z} \cup \{\tau\}$, and $\rightarrow_y$ denotes the transitions that reset the clock y. 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 $i \in \mathbb{N}$. The set of all the paths starting at s in M_d is denoted by $\Pi(s)$. Recall that, for $\Gamma \subseteq \mathcal{AG}$, $\sim_\Gamma^E \stackrel{def}{=} \bigcup_{i\in\Gamma} \sim_i^d$, $\sim_\Gamma^C \stackrel{def}{=} (\sim_\Gamma^E)^+$, and $\sim_\Gamma^D \stackrel{def}{=} \bigcap_{i\in\Gamma} \sim_i^d$. The satisfaction relation $\models$ for ECTLK$_y$ is defined inductively as follows:

$$
\begin{array}{lll}
M_d, s \models p & \text{iff} & p \in \mathcal{V}_d(s), \\
M_d, s \models \neg p & \text{iff} & p \notin \mathcal{V}_d(s), \\
M_d, s \models \alpha \vee \beta & \text{iff} & M_d, s \models \alpha \ \text{ or } \ M_d, s \models \beta, \\
M_d, s \models \alpha \wedge \beta & \text{iff} & M_d, s \models \alpha \ \text{ and } \ M_d, s \models \beta, \\
M_d, s \models \mathrm{E}_y(\alpha \mathrm{U} \beta) & \text{iff} & (\exists s' \in S)(s \rightarrow_y s' \text{ and } (\exists \pi \in \Pi(s'))(\exists m \geq 0) \\
 & & [M_d, \pi(m) \models \beta \text{ and } (\forall j < m)\ M_d, \pi(j) \models \alpha]), \\
M_d, s \models \mathrm{E}_y(\alpha \mathrm{R} \beta) & \text{iff} & (\exists s' \in S)(s \rightarrow_y s' \text{ and } (\exists \pi \in \Pi(s'))(\forall m \geq 0) \\
 & & [M_d, \pi(m) \models \beta \text{ or } (\exists j < m)\ M_d, \pi(j) \models \alpha]), \\
M_d, s \models \overline{\mathrm{K}}_i\alpha & \text{iff} & (\exists s' \in S)(s \sim_i^d s' \text{ and } s' \models \alpha), \\
M_d, s \models \overline{\mathrm{D}}_\Gamma\alpha & \text{iff} & (\exists s' \in S)(s \sim_\Gamma^D \ s' \text{ and } \ s' \models \alpha), \\
M_d, s \models \overline{\mathrm{E}}_\Gamma\alpha & \text{iff} & (\exists s' \in S)(s \sim_\Gamma^E \ s' \text{ and } \ s' \models \alpha), \\
M_d, s \models \overline{\mathrm{C}}_\Gamma\alpha & \text{iff} & (\exists s' \in S)(s \sim_\Gamma^C \ s' \text{ and } \ s' \models \alpha).
\end{array}
$$

An ECTLK$_y$ formula φ is *valid on* M_d (denoted $M_d \models \varphi$) iff $M_d, s^0 \models \varphi$, i.e., φ is true at the initial state of the model M_d.

Having defined syntax and semantics of the ECTLK$_y$ logic, we can now introduce the translation mentioned above. A TECTLK formula φ is translated inductively into the ECTLK$_y$ formula $\mathrm{cr}(\varphi)$ as follows:

- $\mathrm{cr}(p) = p$ if $p \in \mathcal{PV}'$,
- $\mathrm{cr}(\neg p) = \neg p$ if $p \in \mathcal{PV}'$,
- $\mathrm{cr}(\alpha \vee \beta) = \mathrm{cr}(\alpha) \vee \mathrm{cr}(\beta)$,
- $\mathrm{cr}(\alpha \wedge \beta) = \mathrm{cr}(\alpha) \wedge \mathrm{cr}(\beta)$,
- $\mathrm{cr}(\overline{\mathrm{K}}_i\alpha) = \overline{\mathrm{K}}_i\mathrm{cr}(\alpha)$,

- $\mathrm{cr}(\overline{\mathrm{D}}_\Gamma\alpha) = \overline{\mathrm{D}}_\Gamma \mathrm{cr}(\alpha)$,
- $\mathrm{cr}(\overline{\mathrm{E}}_\Gamma\alpha) = \overline{\mathrm{E}}_\Gamma \mathrm{cr}(\alpha)$,
- $\mathrm{cr}(\overline{\mathrm{C}}_\Gamma\alpha) = \overline{\mathrm{C}}_\Gamma \mathrm{cr}(\alpha)$,
- $\mathrm{cr}(\mathrm{E}(\alpha \mathrm{U}_{I_i}\beta)) = \mathrm{E}_y(\mathrm{cr}(\alpha)\mathrm{U}(\mathrm{cr}(\beta) \wedge p_{y\in I_i} \wedge (p_b \vee \mathrm{cr}(\alpha))))$,
- $\mathrm{cr}(\mathrm{E}(\alpha \mathrm{R}_{I_i}\beta)) = \mathrm{E}_y(\mathrm{cr}(\alpha)\mathrm{R}(\mathrm{cr}(\beta) \vee \neg p_{y\in I_i} \vee (\neg p_b \wedge \mathrm{cr}(\alpha))))$.

The translation of the propositional variables and their negations as well as conjunctions and disjunctions is intuitive. Notice that the formula $\mathrm{E}_y(\mathrm{cr}(\alpha)\mathrm{U}(\mathrm{cr}(\beta) \wedge p_{y\in I_i} \wedge (p_b \vee \mathrm{cr}(\alpha))))$ expresses the following conditions:

a.) there exists a path $\pi = (s_0, s_1, \ldots)$ in the discretised interpreted system for $\mathcal{TA}_\varphi$ that starts at a state with the value of the clock y equal to zero; this statement is expressed by using the quantifier E_y in $\mathrm{cr}(\mathrm{E}(\alpha \mathrm{U}_I \beta))$;

b.) there exists a state $s_i = (l, v)$ on π such that $v(y) \in I$ and the translation of β holds in the state; this is expressed by the requirement $\mathrm{cr}(\beta) \wedge p_{y\in I}$;

c.) the translation of α holds in all the states s_j on the path π, for $j < i$; this is expressed by employing the standard until operator, i.e., $\mathrm{cr}(\alpha)\mathrm{U}(\mathrm{cr}(\beta) \wedge p_{y\in I} \wedge (p_b \vee \mathrm{cr}(\alpha)))$,

   Regarding the conjunct $p_b \vee \mathrm{cr}(\alpha)$ notice that we have to take into consideration the shape of a region in which $\mathrm{cr}(\beta)$ holds. Namely, if this region is not boundary, then its borders are open, and therefore each state belonging to the region has some time predecessors that also belong to the same region. Thus, if we require that $\mathrm{E}(\alpha \mathrm{U}_I \beta)$ holds, then $\mathrm{cr}(\alpha)$ must hold continuously until $\mathrm{cr}(\beta)$ and $\mathrm{cr}(\alpha)$ must hold at all the states of the region where $\mathrm{cr}(\beta)$ holds; this is expressed by the condition $p_b \vee \mathrm{cr}(\alpha)$ put in conjunction with $\mathrm{cr}(\beta) \wedge p_{y\in I}$.

Note that the translation for $\mathrm{cr}(\mathrm{E}(\alpha \mathrm{R}_I \beta))$ is the dual of the one for $\mathrm{cr}(\mathrm{E}(\alpha \mathrm{U}_I \beta))$.

The following lemma shows that validity of a 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 $\mathcal{TA}_\varphi$.

Lemma 6 *$M \models \varphi$ iff $M_d \models cr(\varphi)$, for each* TECTLK *formula φ.*

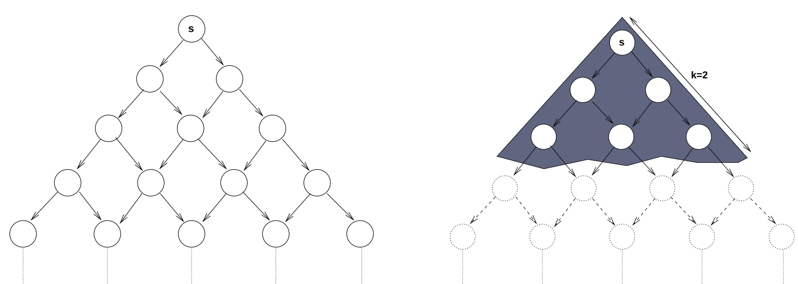
Proof: The proof follows directly from Lemma on Correctness of the labelling algorithm of [1] and Theorem 4.1 of [33] for TECTL fragment of TECTLK, and from the definition of the relation $\sim_i$ for the epistemic fragment of TECTLK. □

In the following we present a BMC method for ECTLK_y over discretised interpreted systems. This, paired with the translation just shown, gives a BMC method for TECTLK.

All the known BMC techniques are based on a notion of satisfaction on finite structures. In particular, BMC for ECTLK$_y$ is based on the $k-$bounded satisfaction for ECTLK$_y$, the definition of which we present below.

5.2.1 Bounded Satisfaction

We start with some auxiliary definitions. Let $M_d = (S_d, s^0, \rightarrow_d, \sim_1^d, \ldots, \sim_m^d, \mathcal{V}_d)$ be a discretised interpreted system, and $k \in \mathbb{N}_+$ a bound. As before, we denote by $\rightarrow_{\mathcal{TA}}$ the subset of $\rightarrow_d$, where transitions are labelled with elements of $\mathfrak{Z} \cup \{\tau\}$, and by $\rightarrow_y$ the set of transitions resetting the clock y. A $k-path$ π in M_d is a finite sequence of states $(s_0, \ldots, s_k)$ such that $s_i \rightarrow_{\mathcal{TA}} s_{i+1}$ for each $0 \leq i < k$. We will denote the set of all the k-paths starting at s in M_d by $\Pi_k(s)$. Note that this set is a convenient way of representing the $k-$bounded subtree rooted at s of the tree resulting from unwinding the discretised interpreted system from s (see Figure 9). A k-path $\pi = (s_0, \ldots, s_k)$ is a *loop* if there exists $0 \leq l \leq k$ such that $\pi(k) \rightarrow_{\mathcal{TA}} \pi(l)$ (see Figure 10).

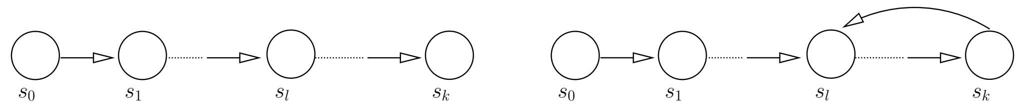


(a) (b)

Fig. 9. (a) Unwinding of a discretised interpreted system M_d from a state s of M_d; (b) $\Pi_2(s)$ for M_d

Definition 8 (k-model) *Let $M_d = (S_d, s^0, \rightarrow_d, \sim_1^d, \ldots, \sim_m^d, \mathcal{V}_d)$ be a discretised interpreted system, and $k \in \mathbb{N}_+$ a bound. A k-*model *for M_d is a structure $M_k = (S_d, s^0, P_k, P_y, \sim_1^d, \ldots, \sim_m^d, \mathcal{V}_d)$, where $P_k = \bigcup_{s \in S_d} \Pi_k(s)$ and $P_y = \{(s, s') \mid s \rightarrow_y s'$ and $s, s' \in S_d\}$.*

Satisfaction of the temporal operator $\mathrm{E}_y\mathrm{R}$ on a k-path π in the bounded case depends on whether or not π is a loop. Therefore, we introduce a function $loop$: $P_k \longrightarrow 2^{\mathbb{N}}$ which allows for the identification of the k-paths that are actually loops. This function is defined by $loop(\pi) = \{i \mid 0 \leq i \leq k$ and $\pi(k) \rightarrow_{\mathcal{TA}} \pi(i)\}$, and it returns the set of all the indices of the states for which there is a transition from the last state of a k-path π. Note that if a k-path is a loop, then it represents an *infinite* path (see Figure 10).



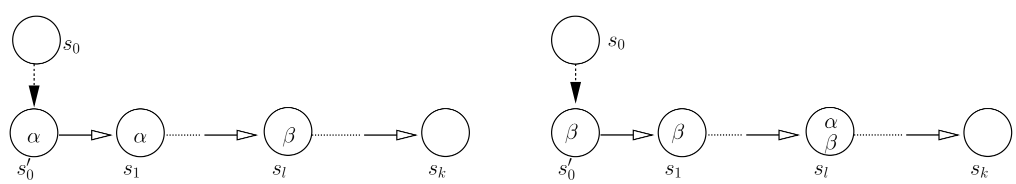
A k-path π with $loop(\pi) = \emptyset$ A k-path π with $loop(\pi) = \{l\}$

Fig. 10. Two kinds of k-paths.

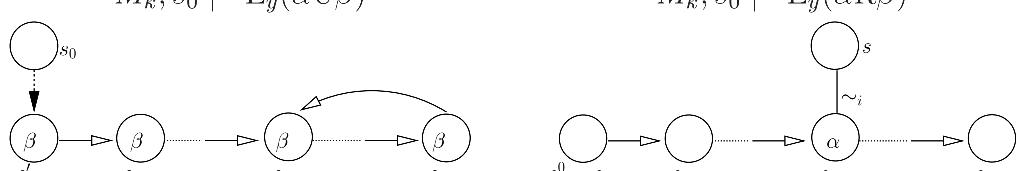
Now we can define a notion of (bounded) satisfaction for ECTLK_y formulae on bounded structures. Let $k \in \mathbb{N}_+$, M_d be a discretised interpreted system, M_k its k-model, and α, β ECTLK_y formulae. $M_k, s \models \alpha$ denotes that α is true at the state s of M_k. The satisfaction relation $\models$ is defined inductively as follows:

$$
\begin{aligned}
&M_k, s \models p && \text{iff } p \in \mathcal{V}_d(s),\\
&M_k, s \models \neg p && \text{iff } p \notin \mathcal{V}_d(s),\\
&M_k, s \models \alpha \vee \beta && \text{iff } M_k, s \models \alpha \text{ or } M_k, s \models \beta,\\
&M_k, s \models \alpha \wedge \beta && \text{iff } M_k, s \models \alpha \text{ and } M_k, s \models \beta,\\
&M_k, s \models \overline{\mathrm{K}}_i\alpha && \text{iff } (\exists \pi \in \Pi_k(s^0))(\exists 0 \le j \le k)(M_k, \pi(j) \models \alpha \text{ and } s \sim_i^d \pi(j)),\\
&M_k, s \models \overline{\mathrm{D}}_\Gamma\alpha && \text{iff } (\exists \pi \in \Pi_k(s^0))(\exists 0 \le j \le k)(M_k, \pi(j) \models \alpha \text{ and } s \sim_\Gamma^D \pi(j)),\\
&M_k, s \models \overline{\mathrm{E}}_\Gamma\alpha && \text{iff } (\exists \pi \in \Pi_k(s^0))(\exists 0 \le j \le k)(M_k, \pi(j) \models \alpha \text{ and } s \sim_\Gamma^E \pi(j)),\\
&M_k, s \models \overline{\mathrm{C}}_\Gamma\alpha && \text{iff } (\exists \pi \in \Pi_k(s^0))(\exists 0 \le j \le k)(M_k, \pi(j) \models \alpha \text{ and } s \sim_\Gamma^C \pi(j)),\\
&M_k, s \models \mathrm{E}_y(\alpha \mathrm{U} \beta) && \text{iff } (\exists s' \in S_d)((s, s') \in P_y \text{ and } (\exists \pi \in \Pi_k(s'))(\exists 0 \le j \le k)\\
& && \quad (M_k, \pi(j) \models \beta \text{ and } (\forall 0 \le i < j)\ M_k, \pi(i) \models \alpha)),\\
&M_k, s \models \mathrm{E}_y(\alpha \mathrm{R} \beta) && \text{iff } (\exists s' \in S_d)((s, s') \in P_y \text{ and } (\exists \pi \in \Pi_k(s'))[(\exists 0 \le j \le k)\\
& && \quad (M_k, \pi(j) \models \alpha \text{ and } (\forall 0 \le i \le j) M_k, \pi(i) \models \beta) \text{ or}\\
& && \quad (\forall 0 \le j \le k)(M_k, \pi(j) \models \beta \text{ and } loop(\pi) \neq \emptyset)]).
\end{aligned}
$$

We use the definition above to interpret ECTLK_y on finite structures. Pictorial descriptions for bounded satisfaction of ECTLK_y formulae are shown in Figure 11.



$M_k, s_0 \models \mathrm{E}_y(\alpha \mathrm{U} \beta)$ $M_k, s_0 \models \mathrm{E}_y(\alpha \mathrm{R} \beta)$



$M_k, s_0 \models \mathrm{E}_y(\alpha \mathrm{R} \beta)$ $M_k, s \models \overline{\mathrm{K}}_i\alpha$

Fig. 11. Examples of satisfaction for ECTLK_y formulae on bounded models.

An ECTLK_y formula φ is *valid on k-model* M_k (denoted $M_d \models_k \varphi$) iff $M_k, s^0 \models \varphi$, i.e., φ is true at the initial state of the k-model M_k. $|M_d|$ denotes the size of M_d, i.e., the sum of the elements of the set S_d and the elements of $\rightarrow_d$.

We can now describe how the model checking problem ($M_d \models \varphi$) can be reduced to the bounded model checking problem ($M_d \models_k \varphi$).

Lemma 7 *Let $k \in \mathbb{N}_+$, M_d be a discretised interpreted system, M_k its k-model, and φ an* ECTLK_y *formula. Then, for any s in M_d, $M_k, s \models \varphi$ implies $M_d, s \models \varphi$.*

Proof: By straightforward induction on the length of φ. $\square$

Lemma 8 *Let M_d be a discretised interpreted system, M_k its k-model, $k = |M_d|$, φ an* ECTLK_y *formula and s a state of M_d. If $M_d, s \models \varphi$, then $M_k, s \models \varphi$.*

Proof:[By induction on the length of φ] The lemma follows directly for the propositional variables and their negations. Next, assume that the hypothesis holds for all the proper sub-formulae of φ. If φ is equal to either $\alpha \wedge \beta$ or $\alpha \vee \beta$, then it is easy to check that the lemma holds. Consider φ to be of the following forms:

(1) $\varphi = \mathrm{E}_y(\alpha \mathrm{U} \beta)$. By the definition of *unbounded* satisfaction we have that there is a state s' in M_d such that $s \rightarrow_y s'$ and there is a path $\pi \in \Pi(s')$ such that there exists $m > 0$ with $(M_d, \pi(m) \models \beta$ and $(\forall 0 \leq i < m)$ $M_d, \pi(i) \models \alpha)$. Since the set of states of M_d is finite, we have that $m \leq k$ (i.e., $m \leq |M_d|$). Thus, by the inductive assumption we have that $M_k, \pi(m) \models \beta$, and $M_k, \pi(i) \models \alpha$ for all $0 \leq i < m$. Now, consider the prefix π_k of length k of the path π. We have that $\pi_k \in \Pi_k(s')$. By the definition of the k-model, $(s, s') \in P_y$. Therefore, by the definition of *bounded* satisfaction we have that $M_k, s \models \mathrm{E}_y(\alpha \mathrm{U} \beta)$.

(2) $\varphi = \mathrm{E}_y(\alpha \mathrm{R} \beta)$. By the definition of *unbounded* satisfaction we have that there is a state s' in M_d such that $s \rightarrow_y s'$ and there is a path $\pi \in \Pi(s')$ such that $(\forall m \geq 0)(M_d, \pi(m) \models \beta$ or $(\exists 0 \leq i < m) M_d, \pi(i) \models \alpha)$. This implies that either (1) $(\forall m \geq 0)(M_d, \pi(m) \models \beta)$, or (2) $(\exists i \leq k)(M_d, \pi(i) \models \alpha$ and $(\forall j \leq i)(M_d, \pi(j) \models \beta))$. Let us consider the following two cases:
   - Assume that (1) holds. Since the set of state of M_d is finite, we have that the path π must be of the following form $(\pi(0), \ldots, \pi(i-1))(\pi(i), \ldots, \pi(k))^\omega$ for some $i \leq k$. Thus, we have that $loop(\pi) \neq \emptyset$, and that the prefix of π of the length k belongs to $\Pi_k(s')$. Further, by the definition of the k-model we have that $(s, s') \in P_y$, and by the inductive assumption we have that $M_k, \pi(m) \models \beta$ for all $0 \leq m \leq k$. Therefore, by the definition of *bounded* satisfaction we have that

$M_k, s \models \mathrm{E}_y(\alpha \mathrm{R} \beta)$.

- Assume that (2) holds. Since the set of states of M_d is finite, we have that $i \leq k$ (i.e., $i \leq |M_d|$). Thus, by the inductive assumption we have that $M_k, \pi(i) \models \alpha$, and $M_k, \pi(j) \models \beta$ for all $0 \leq j \leq i$. Now, consider the prefix π_k of length k of the path π. It is obvious that $\pi_k \in \Pi_k(s')$. Further, by the definition of the k-model, $(s, s') \in P_y$. So, by the definition of *bounded* satisfaction we have that $M_k, s \models \mathrm{E}_y(\alpha \mathrm{R} \beta)$.

(3) $\varphi = \overline{\mathrm{K}}_i \alpha$. By the definition of *unbounded* satisfaction, there is a state s' in M_d such that $s \sim_i^d s'$ and $M_d, s' \models \alpha$. By the inductive assumption, we have that $M_k, s' \models \alpha$. Since s' is reachable, it is reachable from s^0 in $k = |M_d|$ steps at most. Thus, there is a $k-$path $\pi \in P_k(s^0)$ such that $\pi(i) = s'$ for some $i \leq k$. So, we have $M_k, s \models \overline{\mathrm{K}}_i \alpha$.

(4) $\varphi = \overline{\mathrm{E}}_\Gamma \alpha$. $\varphi = \overline{\mathrm{E}}_\Gamma \alpha = \bigvee_{i \in \Gamma} \overline{\mathrm{K}}_i \alpha$. Therefore the result follows from the case above for a specific $i \in \Gamma$, and the basic case for the Boolean connectives.

(5) $\varphi = \overline{\mathrm{D}}_\Gamma \alpha$. Straightforward by definition from the case $\varphi = \overline{\mathrm{K}}_i \alpha$.

(6) $\varphi = \overline{\mathrm{C}}_\Gamma \alpha$. Note that $M_d, s \models \overline{\mathrm{C}}_\Gamma \alpha$ iff $M_d, s \models \bigvee_{i \leq |M_d|} (\overline{\mathrm{E}}_\Gamma)^i \alpha$. So, by induction and the former case, we have $M_k, s \models \overline{\mathrm{C}}_\Gamma \alpha$.

□

The main theorem of this section states that $|M_d|$-bounded satisfaction is equivalent to the unbounded one.

Theorem 1 *Let M_d be a discretised interpreted system M_k its k-model where $k = |M_d|$ and ψ an* ECTLK_y *formula. Then, $M_d \models \psi$ iff $M_d \models_k \psi$.*

Proof: The proof follows from Lemmas 7 and 8. □

5.2.2 Submodels of k-models

The previous subsection ends with the following conclusion: to check that an ECTLK_y formula ψ holds on a discretised interpreted system M_d, it is enough to show that ψ holds on its k-model M_k, for some $k \leq |M_d|$. In this subsection we show a stronger property. Namely, we prove that ψ holds on M_d if and only if ψ holds on a *submodel* of M_k.

Definition 9 (Submodel) *A* submodel *of a k-model $M_k = (S_d, s^0, P_k, P_y, \sim_1^d, \ldots, \sim_m^d, \mathcal{V}_d)$ is a tuple $M'(s) = (S', s, P'_k, P'_y, \sim'_1, \ldots, \sim'_m, \mathcal{V}')$ rooted at state $s \in S_d$, such that $P'_k \subseteq P_k$, $S' = \{r \in S_d \mid (\exists \pi \in P'_k)(\exists i \leq k)\pi(i) = r\} \cup \{s\}$, $P'_y \subseteq P_y \cap (S' \times S')$, $\sim'_i = \sim_i \cap (S' \times S')$ for each $i \in \{1, \ldots, m\}$, and $\mathcal{V}' = \mathcal{V}_d \restriction S'$.*

Satisfaction for ECTLK_y over a submodel $M'(s)$ is defined as for M_k.

We now introduce a definition of a function f_k that gives a bound on the number of k-paths in the submodel $M'(s)$, and a function f_y that gives a bound on the number of elements of the set P'_y in the submodel $M'(s)$. We will show later that the validity of ψ in M_k is equivalent to the validity of ψ in $M'(s)$ provided that the bound k is chosen appropriately considering f_k and f_y, where these are given below.

The function $f_k : \mathcal{WF} \to \mathbb{N}$ is defined by:

- $f_k(p) = f_k(\neg p) = 0$, where $p \in \mathcal{PV}'$,
- $f_k(\alpha \vee \beta) = max\{f_k(\alpha), f_k(\beta)\}$,
- $f_k(\alpha \wedge \beta) = f_k(\alpha) + f_k(\beta)$,
- $f_k(\mathrm{E}_y(\alpha \mathrm{U} \beta)) = k \cdot f_k(\alpha) + f_k(\beta) + 1$,
- $f_k(\mathrm{E}_y(\alpha \mathrm{R} \beta)) = (k+1) \cdot f_k(\beta) + f_k(\alpha) + 1$,
- $f_k(Y\alpha) = f_k(\alpha) + 1$, 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 function $f_y : \mathcal{WF} \to \mathbb{N}$ is defined by:

- $f_y(p) = f_y(\neg p) = 0$, where $p \in \mathcal{PV}'$,
- $f_y(\alpha \vee \beta) = max\{f_y(\alpha), f_y(\beta)\}$,
- $f_y(\alpha \wedge \beta) = f_y(\alpha) + f_y(\beta)$,
- $f_y(\mathrm{E}_y(\alpha \mathrm{U} \beta)) = k \cdot f_y(\alpha) + f_y(\beta) + 1$,
- $f_y(\mathrm{E}_y(\alpha \mathrm{R} \beta)) = (k+1) \cdot f_y(\beta) + f_y(\alpha) + 1$,
- $f_y(Y\alpha) = f_k(\alpha)$, for $Y \in \{\overline{\mathrm{K}}_i, \overline{\mathrm{D}}_\Gamma, \overline{\mathrm{E}}_\Gamma, \overline{\mathrm{C}}_\Gamma\}$.

Lemma 9 *Let $M'(s)$ and $M''(s)$ be two submodels of M_k with $P'_k \subseteq P''_k$, $P'_y \subseteq P''_y$, and ψ an* ECTLK$_y$ *formula. If $M'(s) \models_k \psi$, then $M''(s) \models_k \psi$.*

Proof: By straightforward induction on the length of ψ. □

The lemma below shows that the validity of ψ in M_k is equivalent to the validity of ψ in $M'(s)$ provided that the bound k is chosen by means of f_k and f_y functions.

Lemma 10 *$M_k, s \models \psi$ iff there is a submodel $M'(s)$ of M_k with $|P'_k| \leq f_k(\psi)$ and $|P'_y| \leq f_y(\psi)$ such that $M'(s), s \models \psi$.*

Proof: The implication from right to left is straightforward. To prove the implication left to right, we will use induction on the length of ψ.

The “left-to-right” implication follows directly for the propositional variables and their negations. Consider the following cases:

- Let $\psi = \alpha \vee \beta$ and $M_k, s \models \alpha \vee \beta$. By the definition of bounded satisfaction we have that $M_k, s \models \alpha$ or $M_k, s \models \beta$. Hence, by induction we have that either there is a submodel $M'(s)$ of M_k such that $M'(s), s \models \alpha$ and $|P'_k| \leq f_k(\alpha)$, $|P'_y| \leq f_y(\alpha)$, or there is a submodel $M''(s)$ of M_k such that

$M''(s), s \models \beta$ and $|P''_k| \leq f_k(\beta)$ and $|P''_y| \leq f_y(\beta)$.

Now, consider a submodel $M'''(s)$ of M_k such that:

- $P'''_k = P'_k$ and $P'''_y = P'_y$ if $M'(s), s \models \alpha$,
- $P'''_k = P''_k$ and $P'''_y = P''_y$ otherwise.

Thus, $|P'''_k| \leq max\{f_k(\alpha), f_k(\beta)\}$ and $|P'''_y| \leq max\{f_y(\alpha), f_y(\beta)\}$. It is obvious that $M'''(s), s \models \alpha$ or $M'''(s), s \models \beta$. Therefore, by the definition of bounded satisfaction we have that $M'''(s), s \models \alpha \vee \beta$.

- Let $\psi = \alpha \wedge \beta$ and $M_k, s \models \alpha \wedge \beta$. By the definition of bounded satisfaction we have that $M_k, s \models \alpha$ and $M_k, s \models \beta$. Hence, by induction we have that there is a submodel $M'(s)$ of M_k such that $M'(s), s \models \alpha$ and $|P'_k| \leq f_k(\alpha)$ and $|P'_y| \leq f_y(\alpha)$, and there is a submodel $M''(s)$ of M_k such that $M''(s), s \models \beta$ and $|P''_k| \leq f_k(\beta)$ and $|P''_y| \leq f_y(\beta)$. Now, consider the submodel $M'''(s)$ of M_k such that $P'''_k = P'_k \cup P''_k$ and $P'''_y = P'_y \cup P''_y$. It is easy to observe that $|P'''_k| \leq f_k(\alpha) + f_k(\beta)$ and $|P'''_y| \leq f_y(\alpha) + f_y(\beta)$. So, by Lemma 9, we have that $M'''(s), s \models \alpha$ and $M'''(s), s \models \beta$. Therefore, by the definition of bounded satisfaction we have that $M'''(s), s \models \alpha \wedge \beta$.
- Let $\psi = \mathrm{E}_y(\alpha \mathrm{U} \beta)$ and $M_k, s \models \mathrm{E}_y(\alpha \mathrm{U} \beta)$. By the definition, there is a state $s' \in S_d$ such that $(s, s') \in P_y$ and there is a $k-$path $\pi \in \Pi_k(s')$ such that

$$(\exists 0 \leq m \leq k)(M_k, \pi(m) \models \beta \text{ and } (\forall 0 \leq i < m) M_k, \pi(i) \models \alpha). \tag{1}$$

Hence, by the inductive assumption, for all i such that $0 \leq i < m$ there are submodels $M^i(\pi(i))$ of M_k with $|P^i_k| \leq f_k(\alpha)$ and $|P^i_y| \leq f_y(\alpha)$ and

$$M^i(\pi(i)), \pi(i) \models \alpha. \tag{2}$$

and there is a submodel $M^m(\pi(m))$ of M_k with $|P^m_k| \leq f_k(\beta)$ and $|P^m_y| \leq f_y(\beta)$ and

$$M^m(\pi(m)), \pi(m) \models \beta. \tag{3}$$

Consider a submodel $M'(s)$ of M_k such that $P'_k = \bigcup_{i=0}^m P^i_k \cup \{\pi\}$ and $P'_y = \bigcup_{i=0}^m P^i_y \cup \{(s, s')\}$. Thus, by the construction of $M'(s)$, we have that $(s, s') \in P'_y$ and $\pi \in P'_k$. Therefore, since conditions (1), (2), and (3) hold, by the definition of bounded satisfaction, we have that $M', s \models \mathrm{E}_y(\alpha \mathrm{U} \beta)$ and $|P'_k| \leq k \cdot f_k(\alpha) + f_k(\beta) + 1$ and $|P'_y| \leq k \cdot f_y(\alpha) + f_y(\beta) + 1$.
- Let $\psi = \mathrm{E}_y(\alpha \mathrm{R} \beta)$ and $M_k, s \models \mathrm{E}_y(\alpha \mathrm{R} \beta)$. By the definition, there is a state $s' \in S_d$ such that $(s, s') \in P_y$ and there is a $k-$path $\pi \in \Pi_k(s')$ such that

$$(\exists 0 \leq j \leq k)(M_k, \pi(j) \models \alpha \text{ and } (\forall 0 \leq i \leq j) M_k, \pi(i) \models \beta) \text{ or} \tag{4}$$
$$(\forall 0 \leq j \leq k)(M_k, \pi(j) \models \beta \text{ and } loop(\pi) \neq \emptyset). \tag{5}$$

Let us consider the two cases. First, assume that condition (4) holds. Then, by the inductive assumption, for all i such that $0 \leq i \leq j$ there are submodels $M^i(\pi(i))$ of M_k with $|P^i_k| \leq f_k(\beta)$ and $|P^i_y| \leq f_y(\beta)$ and

$$M^i(\pi(i)), \pi(i) \models \beta, \tag{6}$$

and there is a submodel $M^{''}(\pi(m))$ of M_k with $|P_k^{''}| \leq f_k(\alpha)$ and $|P_y^{''}| \leq f_y(\alpha)$ and

$$M^{''}(\pi(m)), \pi(m) \models \alpha. \tag{7}$$

Consider the submodel $M'(s)$ of M_k such that $P_k' = \bigcup_{i=0}^{j} P_k^i \cup P_k^{''} \cup \{\pi\}$ and $P_y' = \bigcup_{i=0}^{j} P_y^i \cup P_y^{''} \cup \{(s,s')\}$. Thus, by the construction of $M'(s)$, we have that $(s,s') \in P_y'$ and $\pi \in P_k'$. Therefore, since the conditions (4), (6) and (7) hold, by the definition of bounded satisfaction we have that $M'(s), s \models \mathrm{E}_y(\alpha \mathrm{R} \beta)$ and $|P_k'| \leq (k+1) \cdot f_k(\beta) + f_k(\alpha) + 1$ and $|P_y'| \leq (k+1) \cdot f_y(\beta) + f_y(\alpha) + 1$.

Assume now that condition (5) holds. Then, by the inductive assumption, for all j such that $0 \leq j \leq k$ there are submodels $M^j(\pi(j))$ of M_k with $|P_k^j| \leq f_k(\beta)$ and $|P_y^j| \leq f_y(\beta)$ and

$$(M^j(\pi(j)), \pi(j) \models \beta). \tag{8}$$

Consider the submodel $M'(s)$ of M_k such that $P_k' = \bigcup_{j=0}^{k} P_k^j \cup \{\pi\}$ and $P_y' = \bigcup_{i=0}^{k} P_y^i \cup \{(s,s')\}$. Thus, by the construction of $M'(s)$, we have that $(s,s') \in P_y'$ and $\pi \in P_k'$. Therefore, since conditions (4) and (8) hold, by the definition of bounded satisfaction we have that $M'(s), s \models \mathrm{E}_y(\alpha \mathrm{R} \beta)$ and $|P_k'| \leq (k+1) \cdot f_k(\beta) + f_k(\alpha) + 1$ and $|P_y'| \leq (k+1) \cdot f_y(\beta) + f_y(\alpha) + 1$.

- Let $\psi = \overline{\mathrm{K}}_i \alpha$ and $M_k, s \models \overline{\mathrm{K}}_i \alpha$. By the definition, we have that there exists $\pi \in \Pi_k(s^0)$ such that

$$(\exists 0 \leq j \leq k)(s \sim_i \pi(j) \text{ and } \pi(j) \models \alpha). \tag{9}$$

By the inductive assumption there is a submodel $M'(\pi(j))$ of M_k with $|P_k'| \leq f_k(\alpha)$ and $|P_y'| \leq f_y(\alpha)$ such that $M'(\pi(j)), \pi(j) \models \alpha$. Consider a submodel $M''(s)$ of M_k such that $P_k'' = P_k' \cup \{\pi\}$ and $P_y'' = P_y'$. Since $\pi \in P_k''$, $s \in S''$, and condition (9) holds, by the construction of $M''(s)$ and the definition of bounded satisfaction, we have that $M'', s \models \overline{\mathrm{K}}_i \alpha$ and $|P_k''| \leq f_k(\alpha) + 1$ and $|P_y''| \leq f_y(\alpha)$.

- Let $\psi = \overline{\mathrm{E}}_\Gamma \alpha$ and $M_k, s \models \overline{\mathrm{E}}_\Gamma \alpha$. By the definition, we have that there exists $\pi \in \Pi_k(s^0)$ such that

$$(\exists 0 \leq j \leq k)(M_k, \pi(j) \models \alpha \text{ and } s \sim_\Gamma^E \pi(j)). \tag{10}$$

By the inductive assumption there is a submodel $M'(\pi(j))$ of M_k with $|P_k'| \leq f_k(\alpha)$ and $|P_y'| \leq f_y(\alpha)$ such that $M'(\pi(j)), \pi(j) \models \alpha$. Consider a submodel $M''(s)$ of M_k such that $P_k'' = P_k' \cup \{\pi\}$ and $P_y'' = P_y'$. Since $\pi \in P_k''$, $s \in S''$, and condition (10) holds, by the construction of $M''(s)$ and the definition of bounded satisfaction, we have that $M''(s), s \models \overline{\mathrm{E}}_\Gamma \alpha$ and $|P_k''| \leq f_k(\alpha) + 1$ and $|P_y''| \leq f_y(\alpha)$.

- Let $\psi = \overline{\mathrm{D}}_\Gamma\alpha$. This case can be proven similarly to the two above.
- Let $\psi = \overline{\mathrm{C}}_\Gamma\alpha$ and $M_k, s \models \overline{\mathrm{C}}_\Gamma\alpha$. Below, we only prove that $f_k(\overline{\mathrm{C}}_\Gamma\alpha) = f_k(\alpha) + k$ is a sufficient number of paths in a submodel $M'(s)$ validating φ and that $f_y(\overline{\mathrm{C}}_\Gamma\alpha) = f_y(\alpha)$. The actual construction of $M'(s)$ can be given similarly to the case $\psi = \overline{\mathrm{K}}_i\alpha$ and $\psi = \alpha \vee \beta$.

  Note that $\overline{\mathrm{C}}_\Gamma\alpha = \bigvee_{1\leq i\leq k}(\overline{\mathrm{E}}_\Gamma)^i\alpha$, $f_k((\overline{\mathrm{E}}_\Gamma)^1\alpha) = f_k(\overline{\mathrm{E}}_\Gamma\alpha) = f_k(\alpha) + 1$, and $f_y((\overline{\mathrm{E}}_\Gamma)^1\alpha) = f_y(\overline{\mathrm{E}}_\Gamma\alpha) = f_y(\alpha)$. It is easy to show, by induction on i, that $f_k((\overline{\mathrm{E}}_\Gamma)^i\alpha) = f_k(\alpha) + i$ and $f_y((\overline{\mathrm{E}}_\Gamma)^i\alpha) = f_y(\alpha)$, for $i \in \{1, \ldots, k\}$. Therefore, $f_k(\psi) = f_k(\bigvee_{1\leq i\leq k}(\overline{\mathrm{E}}_\Gamma)^i\alpha) = max\{f_k((\overline{\mathrm{E}}_\Gamma)^1\alpha), \ldots, f_k((\overline{\mathrm{E}}_\Gamma)^k\alpha)\}$ $= f_k((\overline{\mathrm{E}}_\Gamma)^k\alpha) = f_k(\alpha) + k$, and $f_y(\psi) = f_y(\bigvee_{1\leq i\leq k}(\overline{\mathrm{E}}_\Gamma)^i\alpha) = max\{f_y((\overline{\mathrm{E}}_\Gamma)^1\alpha), \ldots, f_y((\overline{\mathrm{E}}_\Gamma)^k\alpha)\} = f_y((\overline{\mathrm{E}}_\Gamma)^k\alpha) = f_y(\alpha)$.

□

From Lemma 10 we can now derive the following.

Corollary 1 *$M_k, s^0 \models \psi$ iff there is a submodel $M'(s^0)$ of M_k with $|P'_k| \leq f_k(\psi)$ and $|P'_y| \leq f_y(\psi)$ such that $M'(s^0), s^0 \models \psi$.*

Proof: It follows from the definition of bounded satisfaction and Lemma 10, by using $s = s^0$. □

Theorem 2 *Let M_d be a discretised interpreted system, M_k its k-model, ψ an* ECTLK$_y$ *formula, and $k = |M_d|$. Then, $M_d \models \psi$ iff there exists submodel $M'(s^0)$ of M_k with $P'_k \leq f_k(\psi)$ and $|P'_y| \leq f_y(\psi)$ such that $M'(s^0) \models_k \psi$.*

Proof: Follows from Theorem 1 and Corollary 1. □

5.2.3 Translation to Boolean formulae

As it was mentioned before, the main idea of BMC for ECTLK$_y$ consists in translating the model checking problem for ECTLK$_y$ into the problem of satisfiability of a propositional formula. Given an ECTLK$_y$ formula ψ and a discretised interpreted system M_d, this propositional formula is of the following form:

$$[M_d, \psi]_k = [M_d^{\psi,s^0}]_k \wedge [\psi]_{M_k}. \tag{11}$$

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, whereas the second conjunct encodes a number of constraints that must be satisfied on the '$f_k(\psi)$-submodels' 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. In order to define the formula $[M_d, \psi]_k$, we proceed as follows.

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). Notice that we distinguish between states s encoded as sequences of 0's and 1's and their representations in terms of propositional variables $w[i]$. A finite sequence $(w_0, \ldots, w_k)$ of global state variables is called a *symbolic k-path*. In general, we 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. The j-th symbolic k-path is denoted by $w_{0,j}, \ldots, w_{k,j}$, where $w_{i,j}$ are global state variables for $1 \leq j \leq f_k(\psi)$, $0 \leq i \leq k$. For two global state variables w, w', we define the following propositional formulae:

- $I_s(w)$ is a formula over w, which is true for a valuation s_w of w iff $s_w = s$.
- $p(w)$ is a formula over w, which is true for a valuation s_w of w iff $p \in \mathcal{V}_d(s_w)$, where $p \in \mathcal{PV}'$,
- $H_i(w, w')$ 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 $s_\mathfrak{v} \simeq s_{\mathfrak{v}'}$ (encodes equivalence of local states of agent i).
- $\mathcal{R}(w, w')$ is a formula over w, w', which is true for two valuations s_w of w and $s_{w'}$ of w' iff $s_w \rightarrow_{\mathcal{TA}} s_{w'}$ (encodes the non-resetting transition relation of M_d),
- $R_y(w, w')$ is a formula over w, w', which is true for two valuations s_w of w and $s_{w'}$ of w' iff $s_w \rightarrow_y s_{w'}$ (encodes the transitions resetting the clock y).

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 $1 \leq n \leq f_k(\psi)$. We start off with the definition of its first conjunct, i.e., the definition of $[M_d^{\psi,s^0}]_k$, which constrains the $f_k(\psi)$ symbolic k-paths to be valid k-path of M_k. Namely,

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

The second conjunct, i.e., the formula $[\psi]_{M_k} = [\psi]_k^{[0,0]}$, is inductively defined as follows:

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

This fully defines the encoding of Formula 11.

Now we show that the validity of an ECTLK_y formula ψ on a submodel $M'(s)$, defined by using the functions f_k and f_y, is equivalent to the satisfiability of Formula 11. Once we have shown this fact, we can conclude that the validity of ψ on the discretised interpreted system M_d is equivalent to the satisfiability of Formula 11 (see Theorem 3, p. 32). Further, by taking into account Lemma 6 we can claim that the validity of a TECTLK formula φ over the real time interpreted system for $\mathcal{TA}$ is equivalent to the satisfiability of Formula 11; note that this propositional formula encodes the translation of the ECTLK_y formula $\mathrm{cr}(\varphi)$ over the discretised interpreted system for $\mathcal{TA}_\varphi$.

Lemma 11 *Let M_d be discretised interpreted system, M_k its k-model, and ψ an* ECTLK_y *formula. For each state s of M_d, the following holds: $[M_d^{\psi,s}]_k \wedge [\psi]_{M_k}$ is satisfiable iff there is a submodel $M'(s)$ of M_k with $|P'_k| \leq f_k(\psi)$ and $|P'_y| \leq f_y(\psi)$ such that $M'(s), s \models \psi$.*

Proof: (=>) Let $[M_d^{\psi,s}]_k \wedge [\psi]_{M_k}$ be satisfiable. By the definition of the translation, the propositional formula $[\psi]_{M_k}$ encodes all the sets of $k-$paths of size $f_k(\psi)$ which satisfy the formula ψ and all the sets of transitions resetting the clock y of size $f_y(\psi)$. By the definition of the unfolding of the transition relation, the propositional formula $[M^{\psi,s}]_k$ encodes $f_k(\psi)$ symbolic k-paths to be valid $k-$paths of M_k. Hence, there is a set of $k-$paths in M_k, which satisfies the formula ψ of size smaller or equal to $f_k(\psi)$, and there is a set of transitions resetting the clock y of size $f_y(\psi)$. Thus, we conclude that there is a submodel $M'(s)$ of M_k with $|P'_k| \leq f_k(\psi)$ and $|P'_y| \leq f_y(\psi)$ such that $M'(s), s \models \psi$.

(<=) The proof is by induction on the length of ψ. The lemma follows directly for the propositional variables and their negations. Consider the following cases:

(A) For $\psi = \alpha \vee \beta, \alpha \wedge \beta$, or the temporal operators the proof is like in [24].

(B) Let $\psi = \overline{\mathrm{K}}_l\alpha$. Let $M'(s), s \models \overline{\mathrm{K}}_l\alpha$ with $|P'_k| \leq f_k(\overline{\mathrm{K}}_l\alpha)$ and $|P'_y| \leq f_y(\overline{\mathrm{K}}_l\alpha)$. By definition of bounded satisfaction we have that there is a $k-$path π such that $\pi(0) = s^0$ and $(\exists j \leq k)\ s \sim_l^d \pi(j))$ and $M'(s), \pi(j) \models \alpha$. Hence, by induction we obtain that for some $j \leq k$ the propositional formula $[\alpha]_k^{[0,0]} \wedge [M^{\alpha,\pi(j)}]_k$ is satisfiable. Let $ii = f_k(\alpha) + 1$ be the index of a new symbolic $k-$path which satisfies the formulae $I_{s^0}(w_{0,ii})$ and $H_l(w_{0,0}, w_{j,ii})$ for some $j \in \{1, \ldots, k\}$. Therefore, by the construction above, it follows that the propositional formula $I_{s^0}(w_{0,ii}) \wedge \bigvee_{j=0}^{k} ([\alpha]_k^{[j,ii]} \wedge H_l(w_{0,0}, w_{j,ii}))\ \wedge [M^{\overline{\mathrm{K}}_l\alpha,s}]_k$ is satisfiable. Therefore, the following propositional formula is satisfiable:

$$\bigvee_{1\leq i\leq f_k(\overline{\mathrm{K}}_l\alpha)} \Big(I_{s^0}(w_{0,i})\ \wedge\ \bigvee_{j=0}^{k} ([\alpha]_k^{[j,i]} \wedge H_l(w_{0,0}, w_{j,i})) \wedge [M^{\overline{\mathrm{K}}_l\alpha,s}]_k\Big).$$

Hence, by the definition of the translation of an ECTLK_y formula, the above formula is equal to the propositional formula $[\overline{\mathrm{K}}_l\alpha]_k^{[0,0]} \wedge [M^{\overline{\mathrm{K}}_l\alpha,s}]_k$.

(C) Let $\psi = \overline{\mathrm{E}}_\Gamma\alpha$. Let $M'(s), s \models \overline{\mathrm{E}}_\Gamma\alpha$ with $|P'_k| \leq f_k(\overline{\mathrm{E}}_\Gamma\alpha)$ and $|P'_y| \leq f_y(\overline{\mathrm{E}}_\Gamma\alpha)$. By definition of bounded satisfaction we have that there is a $k-$path π such that $\pi(0) = s^0$ and $(\exists j \leq k)\ s \sim_\Gamma^E \pi(j))$ and $M'(s), \pi(j) \models \alpha$. Hence, by induction we obtain that for some $j \leq k$ the propositional formula $[\alpha]_k^{[0,0]} \wedge [M^{\alpha,\pi(j)}]_k$ is satisfiable. Let $ii = f_k(\alpha) + 1$ be the index of a new symbolic $k-$path which satisfies the formulae $I_{s^0}(w_{0,ii})$ and $H_l(w_{0,0}, w_{j,ii})$ for some $j \in \{1, \ldots, k\}$ and $l \in \Gamma$. Therefore, by the construction above, it follows that the propositional formula $I_{s^0}(w_{0,ii})\ \wedge \bigvee_{j=0}^{k} ([\alpha]_k^{[j,ii]} \wedge \bigvee_{l\in\Gamma} H_l(w_{0,0}, w_{j,ii}))\ \wedge [M^{\overline{\mathrm{E}}_\Gamma\alpha,s}]_k$ is satisfiable. Therefore, the following propositional formula is satisfiable:

$$\bigvee_{1\leq i\leq f_k(\overline{\mathrm{E}}_\Gamma\alpha)} \Big(I_{s^0}(w_{0,i})\ \wedge\ \bigvee_{j=0}^{k} ([\alpha]_k^{[j,i]} \wedge \bigvee_{l\in\Gamma} H_l(w_{0,0}, w_{j,i})) \wedge [M^{\overline{\mathrm{E}}_\Gamma\alpha,s}]_k\Big).$$

Hence, by the definition of the translation of an ECTLK_y formula, the above formula is equal to the propositional formula $[\overline{\mathrm{E}}_\Gamma\alpha]_k^{[0,0]} \wedge [M^{\overline{\mathrm{E}}_\Gamma\alpha,s}]_k$.

(D) Let $\psi = \overline{\mathrm{D}}_\Gamma\alpha$. Let $M'(s), s \models \overline{\mathrm{D}}_\Gamma\alpha$ with $|P'_k| \leq f_k(\overline{\mathrm{D}}_\Gamma\alpha)$ and $|P'_y| \leq f_y(\overline{\mathrm{D}}_\Gamma\alpha)$. By definition of bounded satisfaction we have that there is a $k-$path π such that $\pi(0) = s^0$ and $(\exists j \leq k)\ s \sim_\Gamma^D \pi(j))$ and $M'(s), \pi(j) \models \alpha$. Hence, by induction we obtain that for some $j \leq k$ the propositional formula $[\alpha]_k^{[0,0]} \wedge [M^{\alpha,\pi(j)}]_k$ is satisfiable. Let $ii = f_k(\alpha) + 1$ be the index of a new symbolic $k-$path which satisfies the formulae $I_{s^0}(w_{0,ii})$

and $H_l(w_{0,0}, w_{j,ii})$ for some $j \in \{1,\ldots,k\}$ and for all $l \in \Gamma$. Therefore, by the construction above, it follows that the propositional formula $I_{s^0}(w_{0,ii}) \wedge \bigvee_{j=0}^{k} ([\alpha]_k^{[j,ii]} \wedge \bigwedge_{l\in\Gamma} H_l(w_{0,0}, w_{j,ii})) \wedge [M^{\overline{\mathrm{D}}_\Gamma \alpha, s}]_k$ is satisfiable. Therefore, the following propositional formula is satisfiable:

$$\bigvee_{1\leq i\leq f_k(\overline{\mathrm{D}}_\Gamma\alpha)} \Big(I_{s^0}(w_{0,i}) \wedge \bigvee_{j=0}^{k} ([\alpha]_k^{[j,i]} \wedge \bigwedge_{l\in\Gamma} H_l(w_{0,0}, w_{j,i})) \wedge [M^{\overline{\mathrm{D}}_\Gamma \alpha, s}]_k \Big).$$

Hence, by the definition of the translation of an ECTLK_y formula, the above formula is equal to the propositional formula $[\overline{\mathrm{D}}_\Gamma \alpha]_k^{[0,0]} \wedge [M^{\overline{\mathrm{D}}_\Gamma \alpha, s}]_k$.

(E) Let $\psi = \overline{\mathrm{C}}_\Gamma \alpha$. This can be shown by noting that $\overline{\mathrm{C}}_\Gamma \alpha = \bigvee_{i=1}^{k} (\overline{\mathrm{E}})^i \alpha$ and by a simple induction on i and case C.

□

Theorem 3 *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 $[\psi]_{M_k} \wedge [M^{\psi, s^0}]_k$ is satisfiable.*

Proof: It follows from Theorem 2 and Lemma 11. □

6 Railroad Crossing System

To exemplify the use of the techniques of this paper we verify an extension of the *railroad crossing system* (RCS) [17], a well-known example in the literature of real time verification. In the following we not only verify temporal properties, as it is customary in reactive systems, but a specification that includes epistemic concepts too. The system consists of three agents: Train, Gate, and Controller running in parallel and synchronising through the events: *approach, exit, lower* and *raise* (see Figure 2). When a train approaches the crossing, Train sends an *approach* signal to Controller and enters the crossing between 300 and 500 milliseconds (ms) from this event. When Train leaves the crossing, it sends an *exit* signal to Controller. Controller sends a signal *lower* to Gate exactly 100ms after the *approach* signal is received, and sends a *raise* signal within 100ms after *exit*. Gate performs the transition *down* within 100ms of receiving the request *lower*, and responds to *raise* by moving *up* between 100ms and 200ms.

To model the scenario 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 in Figure 2. 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 $l = (l_1, l_2, l_3) \in L_1 \times L_2 \times L_3$.

In addition to verifying standard specifications based on temporal properties of the system, we can now check a variety of temporal epistemic properties. For instance we could check specifications formalising that:

- There exists a behaviour of RCS such that agent Train considers possible a situation in which it sends an approach signal but agent Gate does not send the signal down within 50 milliseconds.
- There exists a behaviour of RCS such that agent Controller considers possible a situation in which it sends a lower signal but agent Gate does not send the signal down within 50 milliseconds.
- There exists a behaviour of RCS such that agent Train considers possible a situation in which it sends an approach signal and agent Controller sends a lower signal within 10 milliseconds but still agent Gate does not send the signal down within 50 milliseconds.

In the following, as an example, we verify the first property above. This 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 system against property φ, all the states of the discretised interpreted system M_d for RCS with the additional clock y have to be represented as bit vectors first. 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 for 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 $g_3 = (1,1)$; 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 $(\mathfrak{l}[0], \ldots, \mathfrak{l}[5])$ to encode all the possible configuration of locations of the RCS system.

In order to encode the clock valuations of significance for RCS, we have to encode the valuations in $\mathbb{D} = \{k \cdot \Delta \mid 0 \leq k \cdot \Delta \leq 1002\} = \{0, \frac{1}{8}, \frac{2}{8}, \frac{3}{8}, \frac{4}{8}, \frac{5}{8}, \frac{6}{8}, \frac{7}{8}, 1, \ldots, 1002\}$ 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. Since the largest integral value is 1002 and the largest value of the numerators is 8, it is enough to take 10+3 state variables to encode these values for one clock; this is because $2^{10} = 1024$ and $2^3 = 8$. Therefore, 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 all 4 clocks. So, a global state variable for the RCS system is $w = ((\mathfrak{l}[0], \ldots, \mathfrak{l}[5]), (\mathfrak{v}[0], \ldots, \mathfrak{v}[51])) = (w[0], \ldots, w[57])$.

To proceed with the verification of the formula in question, the transition relation of M_d has to be translated into a Boolean formula and $cr(\varphi) = \mathrm{E}_y\mathrm{F}(\overline{\mathrm{K}}_{Train}(\mathfrak{p} \wedge \mathrm{E}_y\mathrm{F}(\neg\mathfrak{q} \wedge p_{y\in[0,50]} \wedge (p_b \vee \top)) \wedge (p_b \vee \top))) = \mathrm{E}_y\mathrm{F}(\overline{\mathrm{K}}_{Train}(\mathfrak{p} \wedge \mathrm{E}_y\mathrm{F}(\neg\mathfrak{q} \wedge p_{y\in[0,50]})))$ has to be translated considering all the possible $f_k(cr(\varphi)) =$ 3 submodels of M_d as described in the previous section.

To proceed with the translation of the transition relation of M_d, we first consider the initial state $s^0 = ((t_0, g_0, c_0), v^0)$ of RCS, where s^0 is represented as a bit vector of 58 consecutive 0's. With the representation above this is encoded by the following propositional formula:

$$I_{s^0}(w_{0,0}) = \bigwedge_{i=0}^{57} \neg w_{0,0}[i].$$

The next step is to encode the transitions of M_d by the formula $\mathcal{R}(w_{i,j}, w_{i+1,j})$ with $j = 1, 2, 3$ and $i \leq k$.

As an example we encode here the witness for depth $k = 2$:

$$[(t_0, g_0, c_0), (0,0,0,0)] \xrightarrow{\tau} [(t_0, g_0, c_0), (\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4})] \xrightarrow{approach} [(t_1, g_0, c_1), (0, \frac{1}{4}, 0, \frac{1}{4})].$$

The formula encoding the first transition for our witness has the following form:

$$\begin{aligned} \mathcal{R}_{rcs}(w_{0,1}, w_{1,1}) := &\bigwedge_{i=0}^{5}(\neg w_{0,1}[i] \wedge \neg w_{1,1}[i]) \wedge \bigwedge_{i=6}^{57} \neg w_{0,1}[i] \wedge \\ &\bigwedge_{i=6}^{17} \neg w_{1,1}[i] \wedge w_{1,1}[18] \wedge \bigwedge_{i=19}^{30} \neg w_{1,1}[i] \wedge w_{1,1}[31] \wedge \\ &\bigwedge_{i=32}^{43} \neg w_{1,1}[i] \wedge w_{1,1}[44] \wedge \bigwedge_{i=45}^{56} \neg w_{1,1}[i] \wedge w_{1,1}[57]. \end{aligned} \tag{12}$$

The formula encoding the second transition for our witness has the form:

$$\mathcal{R}_{rcs}(w_{1,1}, w_{2,1}) := \bigwedge_{i=0}^{5} \neg w_{1,1}[i] \wedge \neg w_{2,1}[0] \wedge \neg w_{2,1}[2] \wedge \neg w_{2,1}[3] \wedge \tag{13}$$

$$\neg w_{2,1}[4] \wedge w_{2,1}[1] \wedge w_{2,1}[5] \wedge \bigwedge_{i=6}^{17} \neg w_{1,1}[i] \wedge w_{1,1}[18] \wedge \bigwedge_{i=19}^{30} \neg w_{1,1}[i] \wedge$$
$$w_{1,1}[31] \wedge \bigwedge_{i=32}^{43} \neg w_{1,1}[i] \wedge w_{1,1}[44] \wedge \bigwedge_{i=45}^{56} \neg w_{1,1}[i] \wedge w_{1,1}[57] \wedge$$
$$\bigwedge_{i=6}^{30} \neg w_{2,1}[i] \wedge w_{2,1}[31] \wedge \bigwedge_{i=32}^{56} \neg w_{2,1}[i] \wedge w_{2,1}[57].$$

Note that in fact Formulae 12 and 13 are fragments of the formulae $\mathcal{R}(w_{0,1}, w_{1,1})$ and $\mathcal{R}(w_{1,1}, w_{2,1})$, respectively. In order to encode the whole example we should model, in a similar way to the above, all the possible transitions of M_d, and encode them as formulae $\mathcal{R}(w_{i,j}, w_{i+1,j})$ with $j = 1, 2, 3$ and $i \leq k$. This is a process that can be automated.

To encode the translation of $cr(\varphi)$, first we need to encode the propositions used in $cr(\varphi)$. For $\mathfrak{p}$ we have $\mathfrak{p}(w) := (\neg w[0] \wedge w[1])$, representing the fact that $\mathfrak{p}$ holds at all the global states with the first local locations equal to $(0, 1)$. For $\mathfrak{q}$ we have $\mathfrak{q}(w) := (w[4] \wedge \neg w[5])$, representing the fact that $\mathfrak{q}$ holds at all the global states with the third local locations equal to $(1, 0)$. To give the translation of the proposition $p_{y \in [0,50]}(w)$, assume the following definition of propositional formulae. For the vectors of state variables $a = (a[1], \ldots, a[t])$ and $b = (b[1], \ldots, b[t])$ we define:

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

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

$$p_{y \in [0,50]}(w) \stackrel{def}{=} geq((w[45], \ldots, w[54]), \mathbf{0}) \wedge$$
$$[le((w[45], \ldots, w[54]), \mathbf{50}) \vee (eq((w[45], \ldots, w[54]), \mathbf{50}) \wedge \bigwedge_{i=55}^{57} \neg w[i])].$$

Further, we have to define the formulae $R_y(w, v)$ and $H_l(w, v)$. The formula $R_y(w, v)$ is defined as follows:

$$R_y(w, v) = \bigwedge_{j=0}^{44} (w[j] \leftrightarrow v[j]) \wedge \bigwedge_{j=45}^{57} (v[j] \leftrightarrow \perp). \qquad (14)$$

Let Idx_l be a set of the indexes of the bits of the local states of agent l. Then, the formula $H_l(w,v)$ is defined as follows:

$$H_l(w,v) = \bigwedge_{i \in Idx_l} w[i] \Leftrightarrow v[i]. \tag{15}$$

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 29. Namely,

$$\begin{gathered}
[\mathrm{cr}(\varphi)]_k^{0,0} = [\mathrm{E}_y\mathrm{F}(\overline{\mathrm{K}}_{Train}(\mathfrak{p} \wedge \mathrm{E}_y\mathrm{F}(\neg\mathfrak{q} \wedge p_{y\in[0,50]} \wedge (p_b \vee \top)) \wedge (p_b \vee \top)))]_k^{0,0} = \\
[\mathrm{E}_y\mathrm{F}(\overline{\mathrm{K}}_{Train}(\mathfrak{p} \wedge \mathrm{E}_y\mathrm{F}(\neg\mathfrak{q} \wedge p_{y\in[0,50]})))]_k^{0,0} = \\
[\mathrm{E}_y(\top\mathrm{U}(\overline{\mathrm{K}}_{Train}(\mathfrak{p} \wedge \mathrm{E}_y\mathrm{F}(\neg\mathfrak{q} \wedge p_{y\in[0,50]}))))]_k^{0,0} = \\
\bigvee_{i=1}^{3}(R_y(w_{0,0}, w_{0,i}) \wedge \bigvee_{j=0}^{k}[\overline{\mathrm{K}}_{Train}(\mathfrak{p} \wedge \mathrm{E}_y\mathrm{F}(\neg\mathfrak{q} \wedge p_{y\in[0,50]}))]_k^{[j,i]}) = \\
(R_y(w_{0,0}, w_{0,1}) \wedge \bigvee_{j=0}^{k}[\overline{\mathrm{K}}_{Train}(\mathfrak{p} \wedge \mathrm{E}_y\mathrm{F}(\neg\mathfrak{q} \wedge p_{y\in[0,50]}))]_k^{[j,1]}) \vee \\
(R_y(w_{0,0}, w_{0,2}) \wedge \bigvee_{j=0}^{k}[\overline{\mathrm{K}}_{Train}(\mathfrak{p} \wedge \mathrm{E}_y\mathrm{F}(\neg\mathfrak{q} \wedge p_{y\in[0,50]}))]_k^{[j,2]}) \vee \\
(R_y(w_{0,0}, w_{0,3}) \wedge \bigvee_{j=0}^{k}[\overline{\mathrm{K}}_{Train}(\mathfrak{p} \wedge \mathrm{E}_y\mathrm{F}(\neg\mathfrak{q} \wedge p_{y\in[0,50]}))]_k^{[j,3]}) = \\
(R_y(w_{0,0}, w_{0,1}) \wedge \bigvee_{j=0}^{k}(\bigvee_{t=1}^{3}(I_{s^0}(w_{0,t}) \wedge \bigvee_{l=0}^{k}([\mathfrak{p} \wedge \mathrm{E}_y\mathrm{F}(\neg\mathfrak{q} \wedge p_{y\in[0,50]})]_k^{[l,t]} \\
\wedge H_l(w_{j,1}, w_{l,t}))))) \vee \\
(R_y(w_{0,0}, w_{0,2}) \wedge \bigvee_{j=0}^{k}(\bigvee_{t=1}^{3}(I_{s^0}(w_{0,t}) \wedge \bigvee_{l=0}^{k}([\mathfrak{p} \wedge \mathrm{E}_y\mathrm{F}(\neg\mathfrak{q} \wedge p_{y\in[0,50]})]_k^{[l,t]} \\
\wedge H_l(w_{j,2}, w_{l,t}))))) \vee \\
(R_y(w_{0,0}, w_{0,3}) \wedge \bigvee_{j=0}^{k}(\bigvee_{t=1}^{3}(I_{s^0}(w_{0,t}) \wedge \bigvee_{l=0}^{k}([\mathfrak{p} \wedge \mathrm{E}_y\mathrm{F}(\neg\mathfrak{q} \wedge p_{y\in[0,50]})]_k^{[l,t]} \\
\wedge H_l(w_{j,3}, w_{l,t}))))) = \\
[R_y(w_{0,0}, w_{0,1}) \wedge \bigvee_{j=0}^{k}(\bigvee_{t=1}^{3}(I_{s^0}(w_{0,t}) \wedge \bigvee_{l=0}^{k}(\mathfrak{p}(w_{l,t}) \wedge [\mathrm{E}_y\mathrm{F}(\neg\mathfrak{q} \wedge p_{y\in[0,50]})]_k^{[l,t]} \wedge \\
H_l(w_{j,1}, w_{l,t}))))] \vee [R_y(w_{0,0}, w_{0,2}) \wedge \bigvee_{j=0}^{k}(\bigvee_{t=1}^{3}(I_{s^0}(w_{0,t}) \wedge \bigvee_{l=0}^{k}(\mathfrak{p}(w_{l,t}) \wedge
\end{gathered}$$

$$[\mathrm{E}_y\mathrm{F}(\neg\mathfrak{q} \wedge p_{y\in[0,50]})]_k^{[l,t]} \wedge H_l(w_{j,2}, w_{l,t}))))] \vee [R_y(w_{0,0}, w_{0,3}) \wedge \bigvee_{j=0}^{k}(\bigvee_{t=1}^{3}(I_{s^0}(w_{0,t}) \wedge$$

$$\bigvee_{l=0}^{k}(\mathfrak{p}(w_{l,t}) \wedge [\mathrm{E}_y\mathrm{F}(\neg\mathfrak{q} \wedge p_{y\in[0,50]})]_k^{[l,t]} \wedge H_l(w_{j,3}, w_{l,t}))))] =$$

$$[R_y(w_{0,0}, w_{0,1}) \wedge \bigvee_{j=0}^{k}(\bigvee_{t=1}^{3}(I_{s^0}(w_{0,t}) \wedge \bigvee_{l=0}^{k}(\mathfrak{p}(w_{l,t}) \wedge H_l(w_{j,1}, w_{l,t}) \wedge$$

$$(\bigvee_{n=1}^{3}(R_y(w_{l,t}, w_{0,n}) \wedge \bigvee_{m=0}^{k}[\neg\mathfrak{q} \wedge p_{y\in[0,50]}]_k^{[m,n]}))))))] \vee$$

$$[R_y(w_{0,0}, w_{0,2}) \wedge \bigvee_{j=0}^{k}(\bigvee_{t=1}^{3}(I_{s^0}(w_{0,t}) \wedge \bigvee_{l=0}^{k}(\mathfrak{p}(w_{l,t}) \wedge H_l(w_{j,2}, w_{l,t}) \wedge$$

$$\bigvee_{n=1}^{3}(R_y(w_{l,t}, w_{0,n}) \wedge \bigvee_{m=0}^{k}[\neg\mathfrak{q} \wedge p_{y\in[0,50]}]_k^{[m,n]}))))] \vee$$

$$[R_y(w_{0,0}, w_{0,3}) \wedge \bigvee_{j=0}^{k}(\bigvee_{t=1}^{3}(I_{s^0}(w_{0,t}) \wedge \bigvee_{l=0}^{k}(\mathfrak{p}(w_{l,t}) \wedge H_l(w_{j,3}, w_{l,t})$$

$$\wedge \bigvee_{n=1}^{3}(R_y(w_{l,t}, w_{0,n}) \wedge \bigvee_{m=0}^{k}[\neg\mathfrak{q} \wedge p_{y\in[0,50]}]_k^{[m,n]}))))] =$$

$$[R_y(w_{0,0}, w_{0,1}) \wedge \bigvee_{j=0}^{k}(\bigvee_{t=1}^{3}(I_{s^0}(w_{0,t}) \wedge \bigvee_{l=0}^{k}(\mathfrak{p}(w_{l,t}) \wedge H_l(w_{j,1}, w_{l,t}) \wedge$$

$$\bigvee_{n=1}^{3}(R_y(w_{l,t}, w_{0,n}) \wedge \bigvee_{m=0}^{k}(\neg\mathfrak{q}(w_{m,n}) \wedge p_{y\in[0,50]}(w_{m,n}))))))] \vee$$

$$[R_y(w_{0,0}, w_{0,2}) \wedge \bigvee_{j=0}^{k}(\bigvee_{t=1}^{3}(I_{s^0}(w_{0,t}) \wedge \bigvee_{l=0}^{k}(\mathfrak{p}(w_{l,t}) \wedge H_l(w_{j,2}, w_{l,t}) \wedge$$

$$\bigvee_{n=1}^{3}(R_y(w_{l,t}, w_{0,n}) \wedge \bigvee_{m=0}^{k}(\neg\mathfrak{q}(w_{m,n}) \wedge p_{y\in[0,50]}(w_{m,n}))))))] \vee$$

$$[R_y(w_{0,0}, w_{0,3}) \wedge \bigvee_{j=0}^{k}(\bigvee_{t=1}^{3}(I_{s^0}(w_{0,t}) \wedge \bigvee_{l=0}^{k}(\mathfrak{p}(w_{l,t}) \wedge H_l(w_{j,3}, w_{l,t}) \wedge$$

$$\bigvee_{n=1}^{3}(R_y(w_{l,t}, w_{0,n}) \wedge \bigvee_{m=0}^{k}(\neg\mathfrak{q}(w_{m,n}) \wedge p_{y\in[0,50]}(w_{m,n}))))))].$$

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

It is worth noting that the logic under analysis in this paper provides for a richer specification language for verification when compared to existing approaches. For instance, in the RCS above we can specify and verify via BMC the TECTLK specification "there exists a behaviour of RCS such that within 100 milliseconds agent Train considers possible a situation in which it sends an approach signal but agent Gate does not send the signal down within 50 milliseconds", represented by the formula:

$$\mathrm{EF}_{[0,100]}\overline{\mathrm{K}}_{Train}(\mathfrak{p} \wedge \mathrm{EF}_{[0,50]}(\neg\mathfrak{q})).$$

Other bounded model checking formalisms have been defined for TCTL [25] and CTLK [23]. With TCTL we can verify dense time but not knowledge. So we could check a less expressive property, e.g., "there exists a behaviour of RCS such that within 100 milliseconds agent Train sends an approach signal but agent Gate does not send the signal down within 50 milliseconds", expressible by the TECTL formula:

$$\mathrm{EF}_{[0,100]}(\mathfrak{p} \wedge \mathrm{EF}_{[0,50]}(\neg\mathfrak{q})).$$

Conversely in a bounded model checking framework for CTLK we can express combinations of knowledge and time but only limited to a discrete model of time. In this weaker language we could for instance verify the specification "there exists a behaviour of RCS such that agent Train considers possible a situation in which it sends an approach signal but agent Gate does not send the signal down", expressible by the CTLK formula

$$\mathrm{EF}\overline{\mathrm{K}}_{Train}(\mathfrak{p} \wedge \mathrm{EF}(\neg\mathfrak{q})).$$

Clearly these two options are not as expressive as our original specification. In the first we have no way of referring to agent Train's knowledge, whereas in the second we cannot make explicit the temporal interval the events should be referring to.

7 Related Work and Conclusions

BMC was initially developed for the verification of reactive systems, and then extended for Multi-Agent Systems [18, 23, 32]. In particular, BMC has been extended to ACTL* [30], TACTL [25], and ACTLKD [32]. These are logics able to represent not only branching time but also modalities of concern in Artificial Intelligence (individual and group knowledge, and correctness of behaviour with respect to specifications). In separate developments BMC has been explored for real time temporal logic [3, 25, 33].

In this paper we have tried to combine these directions and have developed BMC to a new logic that combines real time and knowledge. There is no obstacle to extend the method presented here to handle operators representing correct functioning behaviour [19].

Combinations of real time and knowledge have been defined previously [7, 21] but to our knowledge no verification mechanism has ever been defined for them. To solve the difficulty of dense time, we have made use of discretisation on equal intervals, already employed in [25, 33]. It is worth noting that intervals with explicit 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 formulae produced in the translation can be large, as the example of the paper demonstrates. To evaluate its effectiveness in practical applications, we are currently implementing the method in view of comparing experimental results. We are encouraged that implementations of other BMC-based tools [18, 25, 24] showed largely positive results. We are therefore hopeful that the technique of this paper, once implemented, will produce comparably fast results.

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 *Proceedings of the International Colloquium on Automata, Languages and Programming (ICALP'90)*, volume 443 of *LNCS*, pages 322–335. Springer-Verlag, 1990.

[3] G. Audemard, A. Cimatti, A. Kornilowicz, and R. Sebastiani. Bounded model checking for timed systems. In *Proceedings of the 22nd International Conference on Formal Techniques for Networked and Distributed Systems (FORTE'02)*, volume 2529 of *LNCS*, pages 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 *Proceedings of the ACM/IEEE Design Automation Conference (DAC'99)*, pages 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 J. S. Rosenschein, T. Sandholm, W. Michael, and M. Yokoo, editors, *Proceedings of the Second International Joint Conference on Autonomous Agents and Multi-agent systems (AAMAS-03)*, pages 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 the ACM*, 44(5):633–668, 1997.

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

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

[10] R. Fagin, J. Y. Halpern, and M. Y. Vardi. What can machines know? On the properties of knowledge in distributed systems. *Journal of the ACM*, 39(2):328–376, 1992.

[11] P. Gammie and R. van der Meyden. MCK: Model checking the logic of knowledge. In *Proceedings of 16th International Conference on Computer Aided Verification (CAV'04)*, volume 3114 of *LNCS*, pages 479–483. Springer-Verlag, 2004.

[12] J. Halpern, R. van der Meyden, and M. Y. Vardi. Complete axiomatisations for reasoning about knowledge and time. *SIAM Journal on Computing*, 33(3):674–703, 2003.

[13] J. Halpern and Y. Moses. A guide to completeness and complexity for modal logics of knowledge and belief. *Artificial Intelligence*, 54:319–379, 1992.

[14] W. van der Hoek and M. Wooldridge. Model checking knowledge and time. In *SPIN 2002 – Proceedings of the Ninth International SPIN Workshop on Model Checking of Software*, Grenoble, France, April 2002.

[15] 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.

[16] M. Kacprzak, A. Lomuscio, and W. Penczek. Verification of multiagent systems via unbounded model checking. In N. R. Jennings, C. Sierra, L. Sonenberg, and M. Tambe, editors, *Proceedings of the Third International Conference on Autonomous Agents and Multiagent Systems (AAMAS'04)*, volume II, pages 638–645. ACM, July 2004.

[17] I. Kang and I. Lee. An efficient state space generation for analysis of real-time systems. In *Proceedings of the International Symposium on Software testing and analysis(ISSTA '96)*, pages 4–13. ACM Press, 1996.

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

[19] A. Lomuscio and M. Sergot. Deontic interpreted systems. *Studia Logica*, 75(1):63–92, 2003.

[20] R. van der Meyden and K. Wong. Complete axiomatizations for reasoning about knowledge and branching time. *Studia Logica*, 75(1):93–123, 2003.

[21] Y. Moses and B. Bloom. Knowledge, timed precedence and clocks. In *Proceedings of the 13th ACM symposium on Principles of Distributed Computing (PODC '94)*, pages 274–303. ACM Press, 1994.

[22] W. Nabialek, A. Niewiadomski, W. Penczek, A. Pólrola, and M. Szreter. Verics 2004: A model checker for real time and multi-agent systems. In *Proceedings of the International Workshop on Concurrency, Specification and Programming (CS&P'04)*, volume 170 of *Informatik-Berichte*, pages 88–99. Humboldt University, 2004.

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

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

[25] W. Penczek, B. Woźna, and A. Zbrzezny. Towards bounded model checking for the universal fragment of TCTL. In *Proceedings of the 7th International Symposium on Formal Techniques in Real-Time and Fault Tolerant Systems (FTRTFT'02)*, volume 2469 of *LNCS*, pages 265–288. Springer-Verlag, 2002.

[26] F. Raimondi and A. Lomuscio. Automatic verification of multi-agent systems by model checking via OBDDs. *Journal of Applied Logic*, 2007. To appear in Special issue on Logic-based agent verification.

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

[28] R. van der Meyden and H. Shilov. Model checking knowledge and time in systems with perfect recall. In *Proceedings of the 19th Conference on Foundations of Software Technology and Theoretical Computer Science (FST&TCS'99)*, volume 1738 of *LNCS*, pages 432–445. Springer-Verlag, 1999.

[29] R. van der Meyden and Kaile Su. Symbolic model checking the knowledge of the dining cryptographers. In *Proceedings of the 17th IEEE Computer Security Foundations Workshop (CSFW'04)*, pages 280–291, Washington, DC, USA, 2004. IEEE Computer Society.

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

[31] B. Woźna and A. Lomuscio. A logic for knowledge, correctness, and real time. In *Proceedings of the 5th International Workshop on Computational Logic in Multi-Agent Systems (CLIMA'04)*, volume 3487 of *LNAI*, pages 1–15. Springer-Verlag, 2005.

[32] B. Woźna, A. Lomuscio, and W. Penczek. Bounded model checking for

deontic interpreted systems. In *Proc. of the 2nd Workshop on Logic and Communication in Multi-Agent Systems (LCMAS'04)*, volume 126 of *ENTCS*, pages 93–114. Elsevier, 2004.

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