Bounded model checking real-time multi-agent systems with clock differences: theory and implementation

Alessio Lomuscio[1]⋆, Bożena Woźna[2]⋆⋆, and Andrzej Zbrzezny[2]⋆⋆⋆

[1] Department of Computing, Imperial College London, London SW72BZ, UK.
email: A.Lomuscio@doc.imperial.ac.uk
[2] IMCS, Jan Długosz University. Al. Armii Krajowej 13/15, 42-200 Częstochowa, Poland.
email: {b.wozna,a.zbrzezny}@ajd.czest.pl

Abstract. We present a methodology for verifying epistemic and real-time temporal properties of multi-agent systems. We introduce an interpreted systems semantics based on diagonal timed automata and use a real-time temporal epistemic language to describe properties of multi-agent systems. We develop a bounded model checking algorithm for this setting and present experimental results for a real-time version of the alternating bit-transmission problem obtained by means of a preliminary implementation of the technique.

1 Introduction

Reasoning about knowledge has always been a core concern in AI and in multi-agent systems. This is no surprise 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 logic. One of the most widely studied is based on variants of modal logics and is commonly referred to as epistemic logic [10]. 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.

Specification languages are most useful when they can be verified automatically. In this effort both theorem proving and model checking techniques and tools have been made available for epistemic logic. In particular, model checking techniques based on BDD [18, 20], bounded model checking [16], unbounded model checking [11] have been developed and their implementation either publicly released [18, 12] or made available via a web-interface [15].

Given the above, one may be forgiven for thinking that verification via model checking of temporal epistemic logic has now become of age; however, in many respects the

⋆ The author acknowledge support from the EPSRC (grant GR/S49353).
⋆⋆ The research presented here was conducted while B. Woźna was at University College London supported by EPSRC (grant GR/S49353). The author also acknowledges support from the Ministry of Science and Information Society Technologies under grant number 3 T11C 011 28.
⋆⋆⋆ The author acknowledges partial support from the Ministry of Science and Information Society Technologies under grant number 3 T11C 011 28.

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 use operators able to represent dense temporal intervals. The only work in this line we are aware of is [21], where a bounded model checking algorithm for TECTLK was suggested. In this paper we aim to extend two key limitations of that work in that: 1) we assume a computationally more expressive underlying semantical model (diagonal timed automata), 2) we report on an in-house implementation of this technique and discuss experimental results. Further, to exemplify the use of the techniques described in the paper we present a real-time version of the alternating bit transmission problem — a key requirement of this example is the expressive power of a semantics based on diagonal timed automata as the one presented here.

The rest of the paper is organised as follows. In Section2 we present real-time interpreted systems, a semantics for knowledge and real-time, based on diagonal timed automata. In Section 3 we present syntax and semantics for TECTLK, the logic for which the verification method is defined. In Section 4 we define a bounded model checking algorithm for the logic; given the state-spaces in question are infinite the method involves a tailored discretisation process. Finally we test these techniques on a novel real-time variant of the alternating bit protocol.

2 Diagonal real-time interpreted systems

In [21] a semantics for real-time and knowledge based on non-diagonal timed automata was proposed. Automata are given as the finer grained semantics on which real-time interpreted systems are defined. In that framework the only clock conditions that can be used are of the form $x \sim c$, where x is a clock, c a constant and $\sim$ an equality/inequality relation. While this is appropriate for some scenarios (like the "railroad crossing system"), it is known that in others more expressive tests are required. Crucially, we may need to *compare two clocks of the system as an enabling condition for a transition*. Aim of this paper is to analyse this setting for the case of real-time and epistemic properties by means of diagonal automata.

Of course from a theoretical point of view, every diagonal timed automaton can be transformed into non-diagonal timed automaton [3], but the transformation suffers from an exponential blow up in the size of the automaton's states. However the approach presented here is known to generate considerable complications in the verification methodology [5] and results in a loss of completeness in the resulting bounded model checking technique [14].

To define diagonal real-time interpreted systems we first recall the definitions of diagonal timed automata and their composition. We refer to [19] for discussion and more details.

We assume a finite set X of real variables, called *clocks*, and for $x, y \in X$, $\sim \in \{<, \leq, =, >, \geq\}$, $c \in \mathbb{N}$, where $\mathbb{N} = \{0, 1, \ldots\}$ is a set of natural numbers, we define a set of *clock constraints* over X, denoted by $C(X)$, by means of the following grammar:

$$cc ::= \mathit{true} \mid x \sim c \mid x - y \sim c \mid cc \wedge cc$$

A *clock valuation* v is a total function from X into the set of non-negative real numbers $\mathbb{R}$; $\mathbb{R}^X$ denotes the set of all the clock valuations. For $cc \in C(X)$, $[\![cc]\!]$ denotes the set of all the clock valuations that satisfy cc. The clock valuation that assigns the value 0 to all clocks is denoted by v^0. For $v \in \mathbb{R}^X$ and $\delta \in \mathbb{R}$, $v + \delta$ is the clock valuation that assigns the value $v(x) + \delta$ to each clock x. For $v \in \mathbb{R}^X$ and $Y \subseteq X$, $v[Y]$ denotes the clock valuation of X that assigns the value 0 to each clock in Y and leaves the values of the other clocks unchanged.

Definition 1 (Diagonal timed automaton). *Let $\mathcal{PV}$ be a set of propositional variables. A* diagonal timed automaton *is a tuple $\mathcal{A} = (\Sigma, L, l^0, X, \mathcal{I}, R, \mathcal{V})$, where Σ is a nonempty finite set of actions, L is a nonempty finite set of locations, $l^0 \in L$ is an initial location, $\mathcal{V} : L \mapsto 2^{\mathcal{PV}}$ is a function assigning to each location a set of atomic propositions true in that location, X is a finite set of clocks, $\mathcal{I} : L \mapsto C(X)$ is a state invariant function, and $R \subseteq L \times \Sigma \times C(X) \times 2^X \times L$ is a transition relation.*

An element $(l, \sigma, cc, Y, l') \in R$ represents a transition from location l to location l' labelled with an action σ. The invariant condition states that the automaton is allowed to remain in location l only as long as the constraint $\mathcal{I}(l)$ is satisfied. The guard cc has to be satisfied to enable the transition. The transition resets all clocks in the set Y to the value 0.

As usual, the semantics of diagonal timed automata is defined by associating *dense models* to them.

Definition 2 (Dense model). *Let $\mathcal{A} = (\Sigma, L, l^0, X, \mathcal{I}, R, \mathcal{V})$ be a diagonal timed automaton, and $C(\mathcal{A}) \subseteq C(X)$ a set of all the clock constrains occurring in any enabling condition used in the transition relation R or in a state invariant of $\mathcal{A}$. A* dense model *for $\mathcal{A}$ is a tuple $\mathcal{G}(\mathcal{A}) = (\Sigma \cup \mathbb{R}, Q, q^0, \rightarrow, \widetilde{\mathcal{V}})$, where $\Sigma \cup \mathbb{R}$ is a set of labels, $Q = L \times \mathbb{R}^X$ is a set of states, $q^0 = (l^0, v^0)$ is an initial state, $\widetilde{\mathcal{V}} : Q \mapsto 2^{\mathcal{PV}}$ is a valuation function such that $\widetilde{\mathcal{V}}((l, v)) = \mathcal{V}(l)$, and $\rightarrow \subseteq Q \times (\Sigma \cup \mathbb{R}) \times Q$ is a time/action transition relation defined by:*

- *Time transition:* $(l, v) \xrightarrow{\delta} (l, v + \delta)$ *iff* $(\forall 0 \leq \delta' \leq \delta)\ v + \delta' \in [\![\mathcal{I}(l)]\!]$
- *Action transition:* $(l, v) \xrightarrow{\sigma} (l', v')$ *iff* $(\exists cc \in C(\mathcal{A}))(\exists Y \subseteq X)$ *such that* $v' = v[Y]$, $(l, \sigma, cc, Y, l') \in R$, $v \in [\![cc]\!]$, *and* $v' \in [\![\mathcal{I}(l')]\!]$.

In this paper we take diagonal timed automata to provide the lower level, fine-grained description for the agents; the composition of these defines a multi-agent systems. So the computations of a multi-agent system are simply the traces generated by the executions of a network of diagonal timed automata that communicate through shared actions. We model this communication via the standard notion of the parallel composition [19], as defined below.

Consider a network of m diagonal timed automata $\mathcal{A}_i = (\Sigma_i, L_i, l_i^0, X_i, \mathcal{I}_i, R_i, \mathcal{V}_i)$, for $i = 1, \ldots, m$, such that $L_i \cap L_j = \emptyset$ for all $i, j \in \{1, \ldots, m\}$ and $i \neq j$, and denote by $\Sigma(\sigma) = \{1 \leq i \leq m \mid \sigma \in \Sigma_i\}$ the set of indexes of the automata performing action σ. The *parallel composition* of m diagonal timed automata $\mathcal{A}_i$ is a diagonal timed automaton $\mathcal{A} = (\Sigma, L, l^0, X, \mathcal{I}, R, \mathcal{V})$, where $\Sigma = \bigcup_{i=1}^m \Sigma_i$, $L = \prod_{i=1}^m L_i$, $l^0 = (l_1^0, \ldots, l_m^0)$, $X = \bigcup_{i=1}^m X_i$, $\mathcal{I}((l_1, \ldots, l_m)) = \bigwedge_{i=1}^m \mathcal{I}_i(l_i)$, $\mathcal{V}((l_1, \ldots, l_m)) = \bigcup_{i=1}^m \mathcal{V}_i(l_i)$,

and a transition $((l_1,\ldots,l_m),\sigma,\mathfrak{cc},Y,(l'_1,\ldots,l'_m)) \in R$ iff $(\forall i \in \Sigma(\sigma))\,(l_i,\sigma,\mathfrak{cc}_i,Y_i,l'_i) \in R_i$, $\mathfrak{cc} = \bigwedge_{i\in\Sigma(\sigma)} \mathfrak{cc}_i$, $Y = \bigcup_{i\in\Sigma(\sigma)} Y_i$, and $(\forall j \in \{1,\ldots,m\} \setminus \Sigma(\sigma))\; l'_j = l_j$.

Observe that, given the above, transitions in which actions are not shared are interleaved, whereas the transitions caused by shared action are synchronised.

To give a definition of real-time interpreted systems that supports clock constraints of the form $x - y \sim c$, we first define the notion of *weak region equivalence* [22].

Definition 3 (Weak Region Equivalence). *Assume a set of clocks X, and for any $t \in \mathbb{R}$ let $\langle t\rangle$ denote the fractional (respectively integral) part of t (respectively $\lfloor t \rfloor$). The* weak region equivalence *is a relation $\cong \subseteq \mathbb{R}^X \times \mathbb{R}^X$ defined as follows. For two clock valuations u and v in $\mathbb{R}^X$, $u \cong v$ iff all the following conditions hold:*

(E1.) $\lfloor u(x)\rfloor = \lfloor v(x)\rfloor$, *for all* $x \in X$.
(E2.) $\langle u(x)\rangle = 0$ *iff* $\langle v(x)\rangle = 0$, *for all* $x \in X$.
(E3.) $\langle u(x)\rangle < \langle u(y)\rangle$ *iff* $\langle v(x)\rangle < \langle v(y)\rangle$, *for all* $x, y \in X$.

We will use Z, Z', and so on to denote the equivalence classes induced by the relation $\cong$. As customary, we call these classes zones, and the set of all the zones we denote by $Z(|X|)$.

Definition 4 (Diagonal real-time interpreted system). *Consider m diagonal timed automata and their parallel composition. A* diagonal real-time interpreted system *(or a* model*) is a tuple $M = (\Sigma \cup \mathbb{R}, Q, q^0, \rightarrow, \sim_1, \ldots, \sim_m, \widetilde{\mathcal{V}})$ such that $\Sigma \cup \mathbb{R}$, Q, q^0, $\rightarrow$, and $\widetilde{\mathcal{V}}$ are defined as in Definition 2, and for each agent i, $\sim_i \subseteq Q \times Q$ is a relation defined by: $(l,v) \sim_i (l',v')$ iff $l_i((l,v)) = l_i((l',v'))$ and $v \cong v'$, where $l_i : Q \mapsto L_i$ is a function returning the location of agent i from a global state.*

As in [10] we consider two (global) states to be epistemically indistinguishable for agent i if its local state (i.e., its location) is the same in the two global states. Additionally we assume the agents' clocks to be globally visible, although only privately resettable. For two states to be indistinguishable we further assume the clocks of the states belong to the same zone. This is not dissimilar from [21].

3 TECTLK

In this section we introduce the logic TECTLK(Timed Existential CTL with Knowledge). While the logic is the same as the one described in [21], satisfaction is here defined on diagonal real-time interpreted systems.

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

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

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

Semantics. Let $M = (\Sigma \cup \mathbb{R}, Q, q^0, \rightarrow, \sim_1, \ldots, \sim_m, \widetilde{\mathcal{V}})$ be a *model*. We define a q_0-*run* ρ as a sequence of states: $q_0 \xrightarrow{\delta_0} q_0 + \delta_0 \xrightarrow{\sigma_0} q_1 \xrightarrow{\delta_1} q_1 + \delta_1 \xrightarrow{\sigma_1} q_2 \xrightarrow{\delta_2} \ldots$, where $q_i \in Q$, $\sigma_i \in \Sigma$ and $\delta_i \in \mathbb{R}_+$ for each $i \in \mathbb{N}$, and by $f_{\mathcal{A}}(q_0)$ we denote the set of all such q_0-runs. We say that a state $q \in Q$ is reachable if there is a q^0-run ρ such that there exists a state in ρ equal to q. Finally, in order to give a semantics to TECTLK, we introduce the notation of a *dense path π_ρ corresponding to a run ρ. A dense path π_ρ corresponding to* ρ is a mapping from $\mathbb{R}$ to a set of states Q such that $\pi_\rho(r) = q_i + \delta$ for $r = \Sigma_{j=0}^{i}\delta_j + \delta$ with $i \in \mathbb{N}$ and $0 \leq \delta < \delta_i$. Moreover, we define the following epistemic relations: $\sim_\Gamma^E = \bigcup_{i\in\Gamma} \sim_i$, and $\sim_\Gamma^C = (\sim_\Gamma^E)^+$ (the transitive closure of $\sim_\Gamma^E$), and $\sim_\Gamma^D = \bigcap_{i\in\Gamma} \sim_i$, where $\Gamma \subseteq \mathcal{AG}$.

Definition 5. *Let M be a model such that the set Q contains reachable states only. $M, q \models \alpha$ denotes that α is true at state q in M. The satisfaction relation $\models$ is defined inductively as follows:*

$M, q \models p$ *iff* $p \in \widetilde{\mathcal{V}}(q)$, $M, q \models \alpha \vee \beta$ *iff* $q \models \alpha$ *or* $q \models \beta$,
$M, q \models \neg p$ *iff* $p \notin \widetilde{\mathcal{V}}(q)$, $M, q \models \alpha \wedge \beta$ *iff* $q \models \alpha$ *and* $q \models \beta$,
$M, q \models \mathrm{E}(\alpha \mathrm{U}_I \beta)$ *iff* $(\exists \rho \in f_{\mathcal{A}}(q))(\exists r \in I)[M, \pi_\rho(r) \models \beta$ *and* $(\forall r' < r) M, \pi_\rho(r') \models \alpha]$,
$M, q \models \mathrm{E}(\alpha \mathrm{R}_I \beta)$ *iff* $(\exists \rho \in f_{\mathcal{A}}(q))(\forall r \in I)[M, \pi_\rho(r) \models \beta$ *or* $(\exists r' < r) M, \pi_\rho(r') \models \alpha]$,
$M, q \models \overline{\mathrm{K}}_i \alpha$ *iff* $(\exists q' \in Q)(q \sim_i q'$ *and* $M, q' \models \alpha)$,
$M, q \models \overline{\mathrm{D}}_\Gamma \alpha$ *iff* $(\exists q' \in Q)(q \sim_\Gamma^D q'$ *and* $M, q' \models \alpha)$,
$M, q \models \overline{\mathrm{E}}_\Gamma \alpha$ *iff* $(\exists q' \in Q)(q \sim_\Gamma^E q'$ *and* $M, q' \models \alpha)$,
$M, q \models \overline{\mathrm{C}}_\Gamma \alpha$ *iff* $(\exists q' \in Q)(q \sim_\Gamma^C q'$ *and* $M, q' \models \alpha)$.

We say a TECTLK formula φ is *valid in* M (denoted by $M \models \varphi$) iff $M, q^0 \models \varphi$, i.e., φ is true at the initial state of the model M. In the rest of the paper we are concerned with devising and implementing an automatic model checking algorithm for checking whether a formula φ is valid in a given model M.

4 Bounded Model Checking for TECTLK

Bounded model checking (BMC) is a popular model checking technique for the verification of reactive systems [4, 7]. On discrete-time, it is supported by nuSMV [6] and in its epistemic extension by Verics [15]. Verifying whether a system S satisfies a property P amounts to checking $M_S \models \phi_P$, where M_S is a model capturing S and ϕ_P is a property representing P. In BMC this check is turned into the propositional satisfiability test (ultimately performed by ad-hoc highly-efficient SAT solvers) of $[M_S] \wedge [\phi_P]$, where $[M_S], [\phi_P]$ are appropriate Boolean formulae representing a truncated portion of the model M_S and the modal formula ϕ_P. We refer to [16] for a description of the technique for the case of discrete-time epistemic properties.

To define a BMC method for diagonal real-time interpreted systems, we adapt the BMC technique for TECTLK and non-diagonal automata presented in [21]. We first translate the BMC problem from TECTLK into the BMC problem for ECTLK$_y$, and then we define BMC for ECTLK$_y$. We do not report full details and proofs in this abstract. These can be found in [14].

4.1 Translation from TECTLK to ECTLK$_y$

When dealing with real-time one can use DBMs [8], CDDs [2], or a discretisation technique [1, 17, 22] to represent zones. In the BMC settings for branching real-time logics it is customary to discretise zones. In particular, here we take the discretisation scheme introduced in [22], which uses the following set of discretised clock's values and labels as primitives. Let $\mathbb{Q}$ be a set of rational numbers, and $D_m = \{d \in \mathbb{Q} \mid (\exists k \in \mathbb{N})\ d \cdot 2^m = k\}$ and $E_m = \{e \in \mathbb{Q} \mid (\exists k \in \mathbb{N})\ e \cdot 2^m = k \text{ and } e > 0\}$ for every $m \in \mathbb{N}$. Then, $D = \bigcup_{m=0}^{\infty} D_m$ defines the set of discretised clock's values, and $E = \bigcup_{m=1}^{\infty} E_m$ defines the set of labels. We use this technique to define a discretised model, which is crucial for the translation of the model checking problem for TECTLK to the model checking problem for ECTLK$_y$ as described below.

Definition 6 (Discretised model). *Let $\mathcal{A} = (\Sigma, L, l^0, X, \mathcal{I}, R, \mathcal{V})$ be a diagonal timed automaton resulting from the parallel composition of m diagonal timed automata (agents). A* discretised model *for $\mathcal{A}$ is a tuple $M_d = (\Sigma \cup E, S, s^0, \rightarrow_d, \sim_1^d, \ldots, \sim_m^d, \widetilde{\mathcal{V}}_d)$, where $S = L \times D^X$ is a set of states, $s^0 = (l^0, v^0)$ is the initial state, $\sim_i^d \subseteq S \times S$ is an relation defined by $(l, v) \sim_i^d (l', v')$ iff $l_i((l, v)) = l_i((l', v'))$ and $v \cong v'$, for each agent i, $\widetilde{\mathcal{V}}_d : S \mapsto 2^{\mathcal{P}\mathcal{V}}$ is a valuation function defined by $\widetilde{\mathcal{V}}_d((l, v)) = \mathcal{V}(l)$, and $\rightarrow_d \subseteq S \times (\Sigma \cup E) \times S$ is a time/action transition relation defined by:*

- *Time transition: for any $\delta \in E$, $(l, v) \xrightarrow{\delta}_d (l, v + \delta)$ iff $(l, v) \xrightarrow{\delta} (l, v + \delta)$ in $\mathcal{G}(\mathcal{A})$ and $(\forall \delta' \leq \delta)\ v + \delta' \cong v$ or $v + \delta' \cong v + \delta$,*
- *Action transition: for any $\sigma \in \Sigma$, $(l, v) \xrightarrow{\sigma}_d (l', v')$ iff $(\exists \delta)(\exists v'')$ such that $(l, v) \xrightarrow{\delta}_d (l, v'')$ and $(l, v'') \xrightarrow{\sigma} (l', v')$ in $\mathcal{G}(\mathcal{A})$.*

The general idea of the translation is the same as the one in [21], but obviously given the different capabilities there are differences. In particular, the discretised model used here is infinite; so while the procedure in [21] is sound and complete, the one here is only sound[1].

Specifically, given a multi-agent system modelled by a network of diagonal timed automata $\mathcal{A}_i = (\Sigma_i, L_i, l_i^0, X_i, \mathcal{I}_i, R_i, \mathcal{V}_i)$ and a TECTLK formula φ, we extend each automaton $\mathcal{A}_i$ by a new clock y, an action σ_y, and transitions to obtain a new automaton $\mathcal{A}_i^\varphi = (\Sigma_i \cup \{\sigma_y\}, L_i, l_i^0, X_i', \mathcal{I}_i, R_i', \mathcal{V}_i)$ with $X_i' = X_i \cup \{y\}$ and $R_i' = R_i \cup \{(l, \sigma_y, true, \{y\}, l) \mid l \in L\}$. The clock y corresponds to all the timing intervals appearing in φ, and special transitions are used to reset the new clock. We then construct the discretised model for the parallel composition of $\mathcal{A}_i^\varphi$, denoted by $\mathcal{A}_\varphi$, and augment its valuation function with the set of propositional variables containing a new proposition $p_{y \in I}$ for every interval I appearing in φ, and a new proposition p_b representing that a state s is boundary, i.e., at least one clock from the original automata has to have the fractional part of its valuation equal to zero in s. Finally, we translate the TECTLK formula φ into an ECTLK$_y$ formula $\psi = \mathrm{cr}(\varphi)$ such that model checking of φ over the model for the parallel composition of $\mathcal{A}_i$ can be reduced to the model checking of ψ over the discretised model for $\mathcal{A}_\varphi$. Before

[1] Note though that because of the complexity in the SAT translation and satisfiability checks, BMC is never complete in practice when the system is sufficiently complex, so this is not a real concern.

we define the final part of the above construction, we will first introduce the syntax and semantics for ECTLK$_y$.

Let $p \in \mathcal{PV}' = \mathcal{PV} \cup \{p_b\} \cup \{p_{y\in I} \mid I \text{ is an interval in } \varphi\}$. The set of ECTLK$_y$ formulae is defined by the following grammar:

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

The satisfaction relation for ECTLK$_y$ is defined with respect to a discretised model M_d. Namely, assume that s is a state, α, β formulae of ECTLK$_y$, $\rightarrow_{\mathcal{A}}$ denotes the part of $\rightarrow_d$, where transitions are labelled with elements of $\Sigma \cup E$, and $\rightarrow_y$ denotes the transitions that reset the clock y. Next, define a *path* π in M_d to be a sequence $(s_0, s_1, \ldots)$ of states such that $s_i \rightarrow_{\mathcal{A}} s_{i+1}$ for each $i \in \mathbb{N}$, and denote the set of all the paths starting at s in M_d by $\Pi(s)$. Then, the satisfaction relation $\models$ for ECTLK$_y$ is defined as follows:

$M_d, s \models p$ iff $p \in \widetilde{\mathcal{V}}_d(s)$,
$M_d, s \models \neg p$ iff $p \notin \widetilde{\mathcal{V}}_d(s)$,
$M_d, s \models \alpha \vee \beta$ iff $M_d, s \models \alpha$ or $M_d, s \models \beta$,
$M_d, s \models \alpha \wedge \beta$ iff $M_d, s \models \alpha$ and $M_d, s \models \beta$,
$M_d, s \models \mathrm{E}_y(\alpha \mathrm{U} \beta)$ iff $(\exists s' \in S)(s \rightarrow_y s'$ and $(\exists \pi \in \Pi(s'))(\exists m \geq 0)$
$[M_d, \pi(m) \models \beta$ and $(\forall j < m)\ M_d, \pi(j) \models \alpha])$,
$M_d, s \models \mathrm{E}_y(\alpha \mathrm{R} \beta)$ iff $(\exists s' \in S)(s \rightarrow_y s'$ and $(\exists \pi \in \Pi(s'))(\forall m \geq 0)$
$[M_d, \pi(m) \models \beta$ or $(\exists j \leq m)\ M_d, \pi(j) \models \alpha])$,
$M_d, s \models \overline{\mathrm{K}}_i\alpha$ iff $(\exists \pi \in \Pi(s^0))(\exists j \geq 0)(M_d, \pi(j) \models \alpha$ and $s \sim_i \pi(j))$,
$M_d, s \models \overline{\mathrm{D}}_\Gamma\alpha$ iff $(\exists \pi \in \Pi(s^0))(\exists j \geq 0)(M_d, \pi(j) \models \alpha$ and $s \sim^D_\Gamma \pi(j))$,
$M_d, s \models \overline{\mathrm{E}}_\Gamma\alpha$ iff $(\exists \pi \in \Pi(s^0))(\exists j \geq 0)(M_d, \pi(j) \models \alpha$ and $s \sim^E_\Gamma \pi(j))$,
$M_d, s \models \overline{\mathrm{C}}_\Gamma\alpha$ iff $(\exists \pi \in \Pi(s^0))(\exists j \geq 0)(M_d, \pi(j) \models \alpha$ and $s \sim^C_\Gamma \pi(j))$.

Definition 7 (Validity). *An* ECTLK$_y$ *formula* φ *is* valid in M_d *(denoted* $M_d \models \varphi$*) iff* $M_d, s^0 \models \varphi$*, i.e.,* φ *is true at the initial state of* M_d*.*

We can now translate inductively a TECTLK formula φ into the ECTLK$_y$ formula $\mathrm{cr}(\varphi)$; note that for the propositional and epistemic part of ECTLK$_y$ the translation is defined as the corresponding translation in [16].

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

The following lemma shows that validity of the TECTLK formula φ over the model for $\mathcal{A}$ is equivalent to the validity of $\mathrm{cr}(\varphi)$ over the discretised model for $\mathcal{A}_\varphi$ with the extended valuation function.

Lemma 1 ([14]). *Let φ be a* TECTLK *formula, M a model, and M_d the discretised version of M. Further, let $(l,v)\lfloor X \stackrel{def}{=} (l, v\lfloor X)$. For any state $(l,v) \in Q$ there exists $(l,v') \in S$ such that $(l,v')\lfloor X \cong (l,v)$ and $M,(l,v) \models \varphi$ iff $M_d,(l,v') \models \mathrm{cr}(\varphi)$.*

4.2 ECTLK$_y$ Bounded Model Checking

All the known BMC techniques are based on so called $k-$bounded semantics. In particular, BMC for ECTLK$_y$ is based on the $k-$bounded semantics for ECTLK$_y$, the definition of which we present below.

We start with some auxiliary notions. Let $M_d = (\Sigma \cup E, S, s^0, \rightarrow_d, \sim_1^d, \ldots, \sim_m^d, \widetilde{\mathcal{V}}_d)$ be a discretised model, and $k \in \mathbb{N}_+$ a bound. As before, we denote by $\rightarrow_{\mathcal{A}}$ the part of $\rightarrow_d$, where transitions are labelled with elements of $\Sigma \cup E$, and by $\rightarrow_y$ the transitions that reset the clock y. A *$k-$path* π in M_d is a finite sequence of states $(s_0, \ldots, s_k)$ such that $s_i \rightarrow_{\mathcal{A}} s_{i+1}$ for each $0 \leq i < k$, and $\Pi_k(s)$ denotes the set of all the k-paths starting at s in M_d. A *k-model* for M_d is a structure $M_k = (\Sigma \cup E, S, s^0, P_k, P_y, \sim_1^d, \ldots, \sim_m^d, \widetilde{\mathcal{V}}_d)$, where $P_k = \bigcup_{s\in S} \Pi_k(s)$ and $P_y = \{(s,s') \mid s \rightarrow_y s' \text{ and } s, s' \in S\}$.

The satisfaction of the temporal operator $\mathrm{E}_y\mathrm{R}$ on a k-path in the bounded case depends on whether or not π represents a loop. To indicate k-paths that can simulate loops, we define a function $loop : P_k \mapsto 2^{\mathbb{N}}$ by $loop(\pi) = \{i \mid 0 \leq i \leq k \text{ and } \pi(k) \rightarrow_{\mathcal{A}} \pi(i)\}$.

We can now define a bounded semantics for ECTLK$_y$ formulae. Let $k \in \mathbb{N}_+$, M_d be a discretised model, M_k its k-model, α, β ECTLK$_y$ formulae, and let $M_k, s \models \alpha$ denote that α is true at the state s of M_k. Then, the (bounded) satisfaction relation $\models$ for ECTLK$_y$ is defined as follows:

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

Note that for the propositional and epistemic part of ECTLK$_y$, the (bounded) satisfaction relation $\models$ is defined as the corresponding relation in [21].

Definition 8 (Validity). *An* ECTLK$_y$ *formula φ is* valid in a 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.*

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

Theorem 1. *Let $k \in \mathbb{N}_+$, M_d be a discretised model, M_k its k-model, and φ an* ECTLK$_y$ *formula. 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.

Note that both the discretised model and its k-model are infinite. So, to perform bounded model checking we have to consider a finite submodels of a k-model such that an ECTLK$_y$ formula ψ holds in M_d if and only if ψ holds in a finite submodel of M_k.

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

The bounded semantics for ECTLK$_y$ over a submodel $M'(s)$ is defined as for M_k. Moreover, the following theorem holds.

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 $f_{k,y}$ that gives a bound on the number of elements of the set P'_y in the submodel $M'(s)$. It can be shown that these bound guarantee that the validity of ψ in $M'(s)$ is equivalent to the validity of ψ in M_k (see Theorem 2). The function f_k : ECTLK$_y \to \mathbb{N}$ is defined by:

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

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

Theorem 2 ([14]). *Let* M_d *be a discretised model, and* ψ *an* ECTLK$_y$ *formula. If there exist* $k \in \mathbb{N}_+$ *and* s^0*-submodel* $M'(s^0)$ *of k-model* M_k *with* $P'_k \leq f_k(\psi)$ *and* $|P'_{k,y}| \leq f_{k,y}(\psi)$ *such that* $M'(s^0) \models_k \psi$, *then* $M_d \models \psi$.

Given the above, note that both functions f_k and $f_{k,y}$ give the upper bound on the number of paths in P'_k and number of transitions in $P'_{k,y}$, respectively.

Having defined the bounded semantics, we can easily translate the model checking problem for ECTLK$_y$ to the problem of satisfiability of a Boolean formula that encodes all the discretised model for an ECTLK$_y$ formula under consideration and an appropriate fragments of the considered discretised models. The translation can be done in a similar way as the one in [21] and it is presented in the next section.

4.3 Translation to Boolean formulae

The main idea of BMC for ECTLK$_y$ consists in translating the model checking problem for ECTLK$_y$ into the satisfiability problem of a propositional formula. Namely, given an ECTLK$_y$ formula ψ, a discretised model M_d, and a bound $k \in \mathbb{N}_+$, this proposition formula, denoted by $[M_d, \psi]_k$, is of the form: $[M_d^{\psi,s^0}]_k \wedge [\psi]_{M_k}$. The first conjunct represents possible submodels of M_d such that they consist of $f_k(\psi)$ $k-$paths of M_d and at least one of these submodels is an s^0-submodel. The second conjunct encodes a number of constraints that must hold on these submodels 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 $[M_d, \psi]_k$, we proceed as follows.

Let us assume that each state s of submodels of k-model M_k for the discretised model M_d is encoded by a bit-vector whose length, say b, depends on the number of locations, the number of clocks, and the bound $k \in \mathbb{N}_+$. So, each such a state s 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 0s and 1s 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 $s_\mathfrak{v} \cong s_{\mathfrak{v}'}$ (encodes equality 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 $s_w \rightarrow_\mathcal{A} s_{w'}$ (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 a definition of its first conjunct, i.e., $[M_d^{\psi,s^0}]_k$, 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:

$$[p]_k^{[m,n]} := p(w_{m,n}), \qquad [\alpha \wedge \beta]_k^{[m,n]} := [\alpha]_k^{[m,n]} \wedge [\beta]_k^{[m,n]},$$
$$[\neg p]_k^{[m,n]} := \neg p(w_{m,n}), \qquad [\alpha \vee \beta]_k^{[m,n]} := [\alpha]_k^{[m,n]} \vee [\beta]_k^{[m,n]},$$

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

Lemma 2. *Let M_d be discretised model, 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 $|P'_y| \leq f_{k,y}(\psi)$ 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 $f_{k,y}(\psi)$. 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 $f_{k,y}(\psi)$. Thus, we conclude that there is a submodel $M'(s)$ of M_k with $|P'_k| \leq f_k(\psi)$ and $|P'_y| \leq f_{k,y}(\psi)$ 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:

- For $\psi = \alpha \vee \beta, \alpha \wedge \beta$, or the temporal operators the proof is like in [16].
- Let $\psi = \overline{\mathrm{K}}_l \alpha$. If $M'(s), s \models \overline{\mathrm{K}}_l \alpha$ with $|P'_k| \leq f_k(\overline{\mathrm{K}}_l \alpha)$ and $|P'_y| \leq f_{k,y}(\overline{\mathrm{K}}_l \alpha)$, then by the definition of bounded semantics we have that there is a $k-$path π such that $\pi(0) = s^0$ and $(\exists j \leq k)\ s \sim_l \pi(j))$ 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 formula $I_{s^0}(w_{0,ii})$. Therefore, by the construction above, it follows that the propositional formula $I_{s^0}(w_{0,ii}) \wedge \bigvee_{j=0}^{k} \left([\alpha]_k^{[j,ii]} \wedge H_l(w_{0,0}, w_{j,ii})\right) \wedge [M^{\overline{\mathrm{K}}_l \alpha, s}]_k$ is satisfiable. Therefore, the following propositional formula is satisfiable:

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

  Hence, by the definition of the translation of an ECTLK$_y$ formula, the above formula is equal to the propositional formula $[\overline{\mathrm{K}}_l \alpha]_k^{[0,0]} \wedge [M^{\overline{\mathrm{K}}_l \alpha, s}]_k$.
- The other proofs are similar.

Theorem 3. *Let M_d be a discretised model, and ψ an* ECTLK$_y$ *formula. If there exists $k \in \mathbb{N}_+$ such that $[\psi]_{M_k} \wedge [M_d^{\psi,s^0}]_k$ is satisfiable, then $M_d \models \psi$.*

Proof. Follows from Theorem 2 and Lemma 2.

5 A real-time alternating bit transmission problem

To exemplify the theoretical concepts of the previous sections we analyse a real-time version of one of the variants of the alternating bit protocol. In the original formulation [10] two agents attempt to transmit information over an unreliable communication channel, which they have access to. Sender $\mathcal{S}$ starts sending the bit to receiver $\mathcal{R}$. $\mathcal{R}$ is initially silent but as soon as it receives the bit from $\mathcal{S}$, it starts sending acknowledgements back to $\mathcal{S}$. As soon as $\mathcal{S}$ receives one of these acknowledgements, it stops sending the bit, the system is reset and a new bit is sent. Under these conditions it can be checked automatically [18] that whenever $\mathcal{S}$ receives an acknowledgement it then knows (in the formal epistemic sense) that $\mathcal{R}$ knows the value of the bit (expressed by the formula $AG(\mathbf{recack} \Rightarrow K_{\mathcal{S}}K_{\mathcal{R}}\mathbf{recbit})$). Consider now one of the variants analysed in [13] where $\mathcal{R}$ may (erroneously) send acknowledgements without having received the bit first. Intuitively in this case, the property above will no longer hold; indeed this can also be checked automatically [18].

We extend the scenario above by adding the clock expressions. Assume that each agent has two possibly faulty communication channels to choose from to send bits or acknowledgements. In order to optimise the performance of the transmission both agents concurrently run a channel monitoring service in the background. To this aim they regularly send each other control bits and keep track of the time elapsed since the receipt of a control bit from the other party. The agents send the information bit on the channel that has demonstrated to be in the better working condition, i.e., the one that has recently been able to transmit the control bit from the other party.

To formalise the above we use a network of diagonal timed automata consisting of an automaton for $\mathcal{S}$ (see Figure 1) and an automaton for $\mathcal{R}$ (see Figure 2). $\mathcal{S}$ can be in 11 different local states: *Decide* ("$\mathcal{S}$ selects which bit will be sent"), 0-*ctr-bit* and 1-*ctr-bit* ("$\mathcal{S}$ sends a control bit and listens to $\mathcal{R}$'s control bit"), 0-*select* and 1-*select* ("$\mathcal{S}$ selects the channel to use to send bit 0 (1), or he sends a control bit"), 0-*channel*-1 and 0-*channel*-2 ("$\mathcal{S}$ sends bit 0 through channel 1 (2)"), 1-*channel*-1 and 1-*channel*-2, ("$\mathcal{S}$ sends bit 1 through channel 1 (2)"), 0-*ack* and 1-*ack* ("$\mathcal{S}$ has received an acknowledgement"). $\mathcal{S}$ can perform independently the following actions: 0-*bit*, 1-*bit* ("bit 0 (1) is sent"), *scbs*-1-$fail$, *scbs*-2-$fail$ ("a control bit is sent to a faulty channel 1 (2)"), *s-send-*$fail$ ("bit 0 or 1 is sent to a faulty channel"), *nothing*, and *next-bit* whose interpretation is obvious. The remaining actions are synchronised with $\mathcal{R}$.

$\mathcal{R}$ can be in 10 different local states: *wait* ("$\mathcal{R}$ is listening to the channels"), *ctr-bit* ("$\mathcal{R}$ sends a control bit, or he sends a faulty acknowledgement"), $r0$ and $r1$ ("$\mathcal{R}$ has received bit 0 (1)"), 0-*select* and 1-*select* ("$\mathcal{R}$ selects the channel for the ack"), 0-*channel*-1, 0-*channel*-2, 1-*channel*-1 and 1-*channel*-2, ("$\mathcal{R}$ sends an ack on channel 1 (2)."). $\mathcal{R}$ can perform independently the following actions: *scbr*-1-$fail$, *scbr*-2-$fail$ ("a control bit is sent to a faulty channel 1 (2)"), *r-send-*$fail$ ("an ack is sent to a faulty channel"). We refer to Figures 1, 2 for a pictorial representation.

Further, $\mathcal{S}$ uses 3 clocks (x, x_1, x_2,), and $\mathcal{R}$ three more (y, y_1, y_2). Control bits are sent at regular intervals: t_1 for channel 1 and t_2 for channel 2; the clocks x and y are used for this purpose. Clocks x_i and y_i measure the time since a control bit has been received; x_i gets reset when $\mathcal{S}$ receives a control bit on channel i, likewise for y_i for $\mathcal{R}$. When sending bits (either information bits of acknowledgements) each agent evaluates

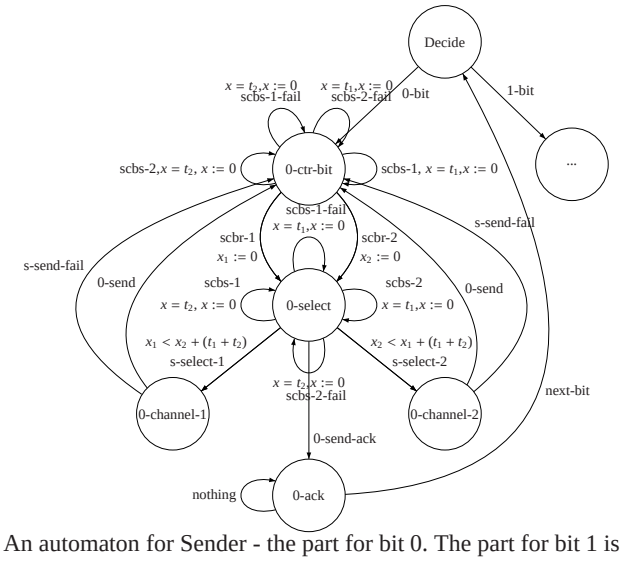

Fig. 1. An automaton for Sender - the part for bit 0. The part for bit 1 is symmetric

the following two clock expressions $z_1 - z_2 < (t_1 + t_2)$ and $z_2 - z_1 < (t_1 + t_2)$ for $z \in \{x, y\}$. When the former expression is true, channel 1 is chosen, when the latter is true, channel 2 is chosen. Intuitively the above guarantees that the channel that has been demonstrated to be alive more recently gets selected. Using the threshold $t_1 + t_2$ enables an agent not to switch channel unnecessarily often (for instance simply because they are desynchronised). Note that ease with which the use of a clock difference allows us to implement real-time channel selection without having a large state space for the automata in question.

The automata run in parallel and synchronise through the actions: *scbs*-1, *scbs*-2, *scbr*-1, and *scbr*-2 ("send a control bit via channel 1 (2)"), 0-*send*, and 1-*send* ("send bit 0 (1)"), 0-*send-ack*, and 1-*send-ack* (" send an acknowledgement to bit 0 (1)").

Given the above, one can construct the automaton $\mathcal{A}_{BTP}$ that describes the whole alternating bit protocol running in real time as well as the set of traces generated by it. In our approach this is done automatically by the bounded model checking implementation.

Now, assume the following set of propositional variables: $\mathcal{PV} = \{$**recack**, **bit0**$\}$, and the following usual interpretation for the proposition variables in $\mathcal{PV}$: $\mathcal{V}_{\mathcal{S}}$(0-*channel*-1) = $\mathcal{V}_{\mathcal{S}}$(0-*channel*-2) = $\mathcal{V}_{\mathcal{S}}$(0-*ack*) = **bit0**, and $\mathcal{V}_{\mathcal{S}}$(0-*ack*) = $\mathcal{V}_{\mathcal{S}}$(1-*ack*) = **recack**.

The typical specification properties that one may be interested in checking for the example above are the following: 1) "forever in the future from t_1 if an acknowledgement has been received by $\mathcal{S}$ and the value of the bit is 0, then $\mathcal{R}$ knows the bit is equal to 0" and 2) "forever in the future from t_1 if an acknowledgement has been received by $\mathcal{S}$ and the value of the bit is 0, then $\mathcal{S}$ knows that $\mathcal{R}$ knows the bit is equal to 0."

By means of an implementation of the technique above we were able to check that the properties above are not satisfied (as intuitively is the case given $\mathcal{R}$'s possible be-

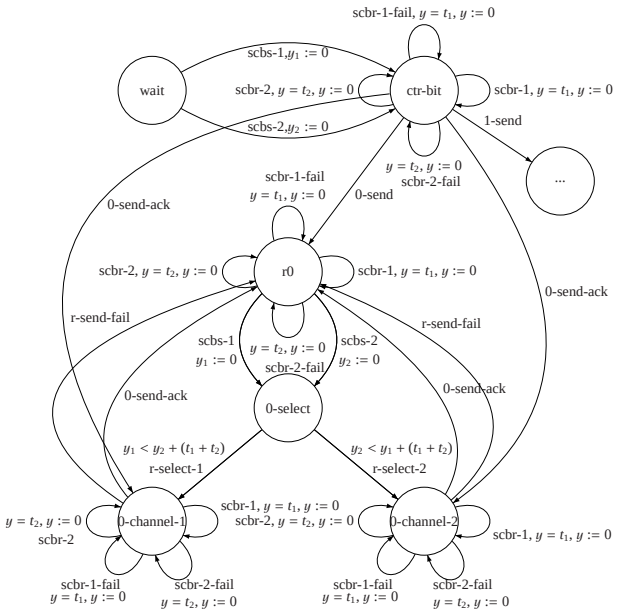

Fig. 2. An automaton for Receiver - the part for bit 0. The part for bit 1 is symmetric.

haviour). More precisely, we can check that the negations of the properties above are true, i.e., the following formulae are satisfied on the model for $\mathcal{A}_{BTP}$:

$$\varphi_1 = \mathrm{EF}_{[t_1,\infty]}(\mathbf{recack} \wedge \mathbf{bit0} \wedge \overline{\mathrm{K}}_{\mathcal{R}}(\neg\mathbf{bit0})), \text{ and}$$
$$\varphi_2 = \mathrm{EF}_{[t_1,\infty]}(\mathbf{recack} \wedge \mathbf{bit0} \wedge \overline{\mathrm{K}}_{\mathcal{S}}\overline{\mathrm{K}}_{\mathcal{R}}(\neg\mathbf{bit0})).$$

depth	**locations**		**clocks' valuation**						
0	*decide*	*wait*	$0\frac{0}{8192}$	$0\frac{0}{8192}$	$0\frac{0}{8192}$	$0\frac{0}{8192}$	$0\frac{0}{8192}$	$0\frac{0}{8192}$	$0\frac{0}{8192}$
1	*decide*	*wait*	$0\frac{4608}{8192}$	$0\frac{4608}{8192}$	$0\frac{4608}{8192}$	$0\frac{4608}{8192}$	$0\frac{4608}{8192}$	$0\frac{4608}{8192}$	$0\frac{4608}{8192}$
2	0-*ctr-bit*	*wait*	$0\frac{4608}{8192}$	$0\frac{4608}{8192}$	$0\frac{4608}{8192}$	$0\frac{4608}{8192}$	$0\frac{4608}{8192}$	$0\frac{4608}{8192}$	$0\frac{4608}{8192}$
3	0-*ctr-bit*	*wait*	$1\frac{0}{8192}$	$1\frac{0}{8192}$	$1\frac{0}{8192}$	$1\frac{0}{8192}$	$1\frac{0}{8192}$	$1\frac{0}{8192}$	$1\frac{0}{8192}$
4	0-*ctr-bit*	*wait*	$1\frac{7168}{8192}$	$1\frac{7168}{8192}$	$1\frac{7168}{8192}$	$1\frac{7168}{8192}$	$1\frac{7168}{8192}$	$1\frac{7168}{8192}$	$1\frac{7168}{8192}$
5	0-*ctr-bit*	*wait*	$2\frac{0}{8192}$	$2\frac{0}{8192}$	$2\frac{0}{8192}$	$2\frac{0}{8192}$	$2\frac{0}{8192}$	$2\frac{0}{8192}$	$2\frac{0}{8192}$
6	0-*ctr-bit*	*ctr-bit*	$2\frac{0}{8192}$	$2\frac{0}{8192}$	$2\frac{0}{8192}$	$2\frac{0}{8192}$	$2\frac{0}{8192}$	$2\frac{0}{8192}$	$2\frac{0}{8192}$
7	0-*ctr-bit*	*ctr-bit*	$2\frac{7524}{8192}$	$2\frac{7524}{8192}$	$2\frac{7524}{8192}$	$2\frac{7524}{8192}$	$2\frac{7524}{8192}$	$2\frac{7524}{8192}$	$2\frac{7524}{8192}$
8	0-*ctr-bit*	*ctr-bit*	$3\frac{0}{8192}$	$3\frac{0}{8192}$	$3\frac{0}{8192}$	$3\frac{0}{8192}$	$3\frac{0}{8192}$	$3\frac{0}{8192}$	$3\frac{0}{8192}$
9	0-*select*	*ctr-bit*	$3\frac{0}{8192}$	$3\frac{0}{8192}$	$3\frac{0}{8192}$	$3\frac{0}{8192}$	$3\frac{0}{8192}$	$3\frac{0}{8192}$	$3\frac{0}{8192}$
10	0-*select*	*ctr-bit*	$3\frac{1}{8192}$	$3\frac{1}{8192}$	$3\frac{1}{8192}$	$3\frac{1}{8192}$	$3\frac{1}{8192}$	$3\frac{1}{8192}$	$3\frac{1}{8192}$
11	0-*ack*	0-*channel*-1	$3\frac{1}{8192}$	$3\frac{1}{8192}$	$3\frac{1}{8192}$	$3\frac{1}{8192}$	$3\frac{1}{8192}$	$3\frac{1}{8192}$	$3\frac{1}{8192}$

Table 1: A witness for the property φ_1.

Tables 1 and 2 illustrate error traces for the above properties, i.e., it shows the witness for the formulae φ_1 and φ_2, respectively, which have been generated by our implementation.

depth	**locations**		**clocks' valuation**						
0	*decide*	*wait*	$0\frac{0}{8192}$	$0\frac{0}{8192}$	$0\frac{0}{8192}$	$0\frac{0}{8192}$	$0\frac{0}{8192}$	$0\frac{0}{8192}$	$0\frac{0}{8192}$
1	*decide*	*wait*	$0\frac{4608}{8192}$	$0\frac{4608}{8192}$	$0\frac{4608}{8192}$	$0\frac{4608}{8192}$	$0\frac{4608}{8192}$	$0\frac{4608}{8192}$	$0\frac{4608}{8192}$
2	0-*ctr-bit*	*wait*	$0\frac{4608}{8192}$	$0\frac{4608}{8192}$	$0\frac{4608}{8192}$	$0\frac{4608}{8192}$	$0\frac{4608}{8192}$	$0\frac{4608}{8192}$	$0\frac{4608}{8192}$
3	0-*ctr-bit*	*wait*	$1\frac{0}{8192}$	$1\frac{0}{8192}$	$1\frac{0}{8192}$	$1\frac{0}{8192}$	$1\frac{0}{8192}$	$1\frac{0}{8192}$	$1\frac{0}{8192}$
4	0-*ctr-bit*	*wait*	$1\frac{7168}{8192}$	$1\frac{7168}{8192}$	$1\frac{7168}{8192}$	$1\frac{7168}{8192}$	$1\frac{7168}{8192}$	$1\frac{7168}{8192}$	$1\frac{7168}{8192}$
5	0-*ctr-bit*	*wait*	$2\frac{0}{8192}$	$2\frac{0}{8192}$	$2\frac{0}{8192}$	$2\frac{0}{8192}$	$2\frac{0}{8192}$	$2\frac{0}{8192}$	$2\frac{0}{8192}$
6	0-*ctr-bit*	*ctr-bit*	$2\frac{0}{8192}$	$2\frac{0}{8192}$	$2\frac{0}{8192}$	$2\frac{0}{8192}$	$2\frac{0}{8192}$	$2\frac{0}{8192}$	$2\frac{0}{8192}$
7	0-*ctr-bit*	*ctr-bit*	$2\frac{7524}{8192}$	$2\frac{7524}{8192}$	$2\frac{7524}{8192}$	$2\frac{7524}{8192}$	$2\frac{7524}{8192}$	$2\frac{7524}{8192}$	$2\frac{7524}{8192}$
8	0-*ctr-bit*	*ctr-bit*	$3\frac{0}{8192}$	$3\frac{0}{8192}$	$3\frac{0}{8192}$	$3\frac{0}{8192}$	$3\frac{0}{8192}$	$3\frac{0}{8192}$	$3\frac{0}{8192}$

tion of the formula φ_1 and the appropriate fragments of the model for $\mathcal{A}_{BTP}$ as described in [14]. The formula in question consists of 125260 variables and 258821 clauses; our implementation needed 19.6 second and 18.7 MB memory to produce it. Its satisfaction was checked by MiniSat [9], a mainstream SAT solver; 4.0 seconds and 19.9 MB of memory were needed to check this.

BMC					**MiniSat**		
k	**variables**	**clauses**	**sec**	**MB**	**sec**	**MB**	**satisfiable**
2	12243	28811	2.7	4.5	< 0.1	5.1	NO
3	20771	48413	8.8	5.7	< 0.1	6.2	NO
4	35589	85115	24.0	8.0	0.2	8.1	NO
5	49967	117551	55.2	9.8	0.6	10.1	NO
6	66952	154829	115.7	11.9	1.1	11.9	NO
7	86688	197030	206.1	14.9	2.4	14.2	NO
8	120067	278552	356.9	19.2	12.9	20.2	NO
9	147687	337205	587.3	23.9	9.9	24.1	NO
10	178628	401492	922.3	27.5	20.5	28.6	NO
11	213034	471494	1364.4	31.4	320.0	81.8	YES

Table 3: The computation of the witness - 3 paths

For what concerns the satisfaction of φ_2, the corresponding experimental results are presented in Table 3 and in Table 4. Table 3 refers to the search assuming 3 paths are needed (this is the upper bound is given by the function f_k); Table 4 summarises the result for a search of only 1 path. The tables show the following data: the first column represents the bound on the model for $\mathcal{A}_{BTP}$; the next two show the number of variables and clauses generated by BMC during the translation of φ_2 into a Boolean formula; the next two show the time and memory needed by BMC to generate the set of clauses; the next two columns give the time and the memory required by MiniSat to check satisfaction, and the last column shows the answer given by MiniSat.

BMC					**MiniSat**		
k	**variables**	**clauses**	**sec**	**MB**	**sec**	**MB**	**satisfiable**
2	3570	7706	0.2	3.2	< 0.1	3.7	NO
3	6021	12877	0.5	3.5	< 0.1	4.2	NO
4	10164	22320	1.3	4.2	< 0.1	4.7	NO
5	14213	30551	2.5	4.6	< 0.1	5.2	NO
6	18997	39934	4.8	5.3	< 0.1	5.8	NO
7	24564	50496	8.4	6.0	0.1	6.2	NO
8	33578	70301	14.4	7.0	0.2	7.5	NO
9	41277	84625	23.6	8.3	0.4	8.5	NO
10	49925	100281	33.7	9.2	0.4	9.4	NO

11	59570	117296	54.0	10.2	0.9	10.3	YES

Table 4: The computation of the witness - 1 path

For reference, all the above experiments were performed on an AMD Athlon XP 1800 (1544 MHz), 768 MB main memory, running Linux with Kernel 2.6.15. Unfortunately we are not able to compare these results to other tools as we are not aware of any other implementation available that is capable of a real-time epistemic check for (diagonal and non-diagonal) automata.

6 Conclusions

Model checking real-time in AI and MAS is still in its infancy. In [21] a first proposal was made for a bounded model checking algorithm for real-time epistemic properties based on non-diagonal automata semantics. In this paper we have tried to extend that work by allowing the expressivity of clock differences. We have proposed a syntax, semantics for the logic, as well as a bounded model checking method, and showed experimental results of a preliminary implementation for a real-time version of the alternating bit protocol.

References

1. E. Asarin, M. Bozga, A. Kerbrat, O. Maler, A. Pnueli, and A. Rasse. Data-structures for the verification of Timed Automata. In *Proceedings of International Workshop on Hybrid and Real-Time Systems (HART'97)*, volume 1201 of *LNCS*, pages 346–360. Springer-Verlag, 1997.
2. G. Behrmann, K. Larsen, J. Pearson, C. Weise, and W. Yi. Efficient Timed Reachability Analysis Using Clock Difference Diagrams. In *Proceedings of the 11th International Conference on Computer Aided Verification (CAV'99)*, volume 1633 of *LNCS*, pages 341–353. Springer-Verlag, 1999.
3. B. Bérard, A. Petit, V. Diekert, and P. Gastin. Characterization of the expressive power of silent transitions in timed automata. *Fundamenta Informaticae*, 36(2-3):145–182, 1998.
4. A. Biere, A. Cimatti, E. Clarke, and Y. Zhu. Symbolic model checking without BDDs. In *Proc. of TACAS'99*, volume 1579 of *LNCS*, pages 193–207. Springer-Verlag, 1999.
5. P. Bouyer, F. Laroussinie, and P. Reynier. Diagonal constraints in timed automata: Forward analysis of timed systems. In Paul Pettersson and Wang Yi, editors, *Proceedings of the 3rd International Conference on Formal Modelling and Analysis of Timed Systems (FORMATS'05)*, volume 3829 of *LNCS*, pages 112–126, Uppsala, Sweden, November 2005. Springer.
6. A. Cimatti, E. M. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani, and A. Tacchella. NuSMV2: An open-source tool for symbolic model checking. In *Proceedings of the 14th International Conference on Computer Aided Verification (CAV'02)*, volume 2404 of *LNCS*, pages 359–364. Springer-Verlag, 2002.

7. P. Dembiński, A. Janowska, P. Janowski, W. Penczek, A. Pólrola, M. Szreter, B. Woźna, and A. Zbrzezny. VerICS: A tool for verifying Timed Automata and Estelle specifications. In *Proc. of the 9th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'03)*, volume 2619 of *LNCS*, pages 278–283. Springer-Verlag, 2003.
8. D. Dill. Timing assumptions and verification of finite state concurrent systems. In *Automatic Verification Methods for Finite-State Systems*, volume 407 of *LNCS*, pages 197–212. Springer-Verlag, 1989.
9. N. Eén and N. Sörensson. MiniSat. http://www.cs.chalmers.se/Cs/Research/FormalMethods/MiniSat/.
10. R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning about Knowledge*. MIT Press, Cambridge, 1995.
11. 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.
12. A. Lomuscio and F. Raimondi. MCMAS: A model checker for multi-agent systems. In H. Hermanns and J. Palsberg, editors, *Proceedings of TACAS 2006, Vienna*, volume 3920, pages 450–454. Springer Verlag, March 2006.
13. A. Lomuscio and M. Sergot. A formalisation of violation, error recovery, and enforcement in the bit transmission problem. *Journal of Applied Logic*, 2(1):93–116, March 2004.
14. A. Lomuscio, B. Woźna, and A. Zbrzezny. Bounded model checking real-time multi-agent systems with clock differences: theory and implementation. Technical Report RN/06/03, Department of Computer Science, University College London, Gower Street, London WC1E 6BT, United Kingdom, 2006.
15. 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.
16. W. Penczek and A. Lomuscio. Verifying epistemic properties of multi-agent systems via bounded model checking. *Fundamenta Informaticae*, 55(2):167–185, 2003.
17. W. Penczek, 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.
18. F. Raimondi and A. Lomuscio. Automatic verification of multi-agent systems by model checking via OBDDs. *Journal of Applied Logic*, 2005. To appear in Special issue on Logic-based agent verification.
19. S. Tripakis and S. Yovine. Analysis of timed systems using time-abstracting bisimulations. *Formal Methods in System Design*, 18(1):25–68, 2001.
20. 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.
21. B. Woźna, A. Lomuscio, and W. Penczek. Bounded model checking for knowledge over real time. In *Proceedings of the 4st International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'05)*, volume I, pages 165–172. ACM Press, July 2005.
22. A. Zbrzezny. SAT-based reachability checking for timed automata with diagonal constraints. *Fundamenta Informaticae*, 67(1-3):303–322, 2005.