A Logic for Knowledge, Correctness, and Real Time *

Bożena Woźna and Alessio Lomuscio

Department of Computer Science
University College London
Gower Street, London WC1E 6BT,
United Kingdom
email: {B.Wozna,A.Lomuscio}@cs.ucl.ac.uk

Abstract. We present TCTLKD, a logic for knowledge, correctness and real time. TCTLKD is interpreted on real time deontic interpreted systems, and extension to continuous time of deontic interpreted systems. We exemplify the use of TCTLKD by discussing a variant of the "railroad crossing system".

1 Introduction

Logic has a long tradition in the area of formal theories for multi-agent systems (MAS). Its role is to provide a precise and unambiguous specification language to describe, reason about, and predict the behaviour of a system.

While in the early 80's existing logical formalisms from other areas such as philosophical logic, concurrency theory, etc., were imported with little of no modification to the area of MAS, from the late 80's onwards specific formalisms have been designed, studied, and tailored to the needs of MAS. Of particular note is the case of epistemic logic, or the logic of knowledge.

Focus on epistemic logics in MAS began with the use of the modal logic system $S5$ developed independently by Hintikka [18] and Aumann [4] in formal logic and economics respectively. This starting point formed the core basis of a number of studies that appeared in the past 20 years, including formalisations of group knowledge [13, 15, 19], combinations of epistemic logic with time[16, 17, 30], auto-epistemic logics [28, 31], epistemic updates [5, 22], broadcast systems and hypercubes [12, 21], etc. Epistemic logic is no longer a remarkable special case of a normal modal system, but has now become an area of study on its own with regular thematic workshops and conferences.

In particular in applications, extensions of epistemic logic to represent also temporal concepts are particularly useful as this allows to reason about the temporal evolution of epistemic states, knowledge of a changing world, etc. Traditionally this is achieved by combining a temporal logic for discrete linear time

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

[25–27] with the logic $S5$ for knowledge on a computationally grounded semantics like interpreted systems [11]. Various classes of MAS (synchronous, asynchronous, perfect recall, no learning, etc) can be identified in this framework, and axiomatisations have been provided [14, 29]. More recently, combinations of branching time logic CTL [6, 9, 10] with the epistemic logic $S5$ have been studied, and axiomatisation provided [30].

All efforts above have focused on a discrete model of time, either in its linear or branching versions. While this is useful and adequate in most applications, certain classes of scenarios (notably robotics and networking) require a model of time as a continuous flows of events. Indeed, in the area of timed-systems the modal logic TCTL has been suggested as an adequate formalism to model real time; axiomatisations and decidability results have been provided [1]. In this paper we propose a logic (which we call TCTLKD) combining the temporal aspects of TCTL with the epistemic notions defined by $S5$, as well as the correctness notion defined in the logic $KD45^{i-j}$ [23].

Traditionally, the semantics of temporal epistemic logic is defined on variants of interpreted systems to provide an interpretation to the epistemic modalities. These use the notion of *protocol* to provide a basis for the action selection mechanism of the agents. Since we are working on real time, here we shall use the finer grained semantics of timed-automata to model the agents' evolution. We then synchronise networks of timed-automata to provide a general model of a MAS.

The rest of the paper is organised as follows. In Section 2 we define the concept of interpreted systems on real time by taking the parallel composition of timed-automata. In Section 3 we define the logic TCTLKD as an extension to real time of the logic for knowledge and correctness as defined in [23, 24]. In Section 4 we provide a case study analysis to demonstrate its use in applications. We conclude in Section 5 by discussing related and future work on this subject.

2 Interpreted Systems over Real Time

Interpreted systems are traditionally defined as a set of infinite runs on global states [11]. In this model each run is a discrete sentence of events. At each global state, each agent selects an action according to a (possibly non-deterministic) protocol. In this section we extend (discrete) interpreted systems to real time interpreted systems in two aspects. First, we specify the agents' behaviour by a finer grained semantics: timed automata. Second, by means of parallel composition of timed automata, we define a class of interpreted systems operating on real time.

We begin by recalling the concept of timed automata, as introduced in [2]. Timed automata are extensions of finite state automata with constraints on timing behaviour. The underlying finite state automata are augmented with a set of real time variables.

2.1 Timed Automata.

Let $\mathcal{X}$ be a finite set of real variables, called *clocks*. The set of *clock constraints* over $\mathcal{X}$ is defined by the following grammar:

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

where $x \in \mathcal{X}$, $c \in \mathbb{N}$, and $\sim \in \{\leq, <, =, >, \geq\}$. The set of all the clock constraints over $\mathcal{X}$ is denoted by $\mathcal{C}(\mathcal{X})$. A *clock valuation* on $\mathcal{X}$ is a tuple $v \in \mathbb{R}_+^{|\mathcal{X}|}$. The value of the clock x in v is denoted by $v(x)$. For a valuation v and $\delta \in \mathbb{R}_+$, $v+\delta$ denotes the valuation v' such that for all $x \in \mathcal{X}$, $v'(x) = v(x) + \delta$. Moreover, let $\mathcal{X}^*$ be the set $\mathcal{X} \cup \{x_0\}$, where x_0 is a clock whose value is always 0, that is, its value does not increase with time as the values of the other clocks. Then, an *assignment* $\mathfrak{as}$ is a function from $\mathcal{X}$ to $\mathcal{X}^*$, and the set of all the assignments over $\mathcal{X}$ is denoted by $\mathfrak{A}(\mathcal{X})$. By $v[\mathfrak{as}]$ we denote the valuation v' such that for all $x \in \mathcal{X}$, if $\mathfrak{as}(x) \in \mathcal{X}$, then $v'(x) = v(\mathfrak{as}(x))$, otherwise $v'(x) = 0$.

Let $v \in \mathbb{R}_+^{|\mathcal{X}|}$, the satisfaction relation $\models$ for a clock constraint $\mathfrak{cc} \in \mathcal{C}(\mathcal{X})$ 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 $v \models \mathfrak{cc}$ and $v \models \mathfrak{cc}'$.

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

Definition 1 (Timed Automaton). *A* timed automaton *is a tuple* $\mathcal{TA} = (\mathfrak{Z}, L, l^0, \mathcal{X}, E, \mathfrak{I})$, *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,*
- $E \subseteq L \times \mathfrak{Z} \times \mathcal{C}(\mathcal{X}) \times \mathfrak{A}(\mathcal{X}) \times L$ *is a transition relation,*
- $\mathfrak{I} : L \to \mathcal{C}(\mathcal{X})$ *is a function, called a* location invariant*, which assigns to each location* $l \in L$ *a clock constraint defining the conditions under which* $\mathcal{TA}$ *can stay in* l.

Each element e *of* E *is denoted by* $l \xrightarrow{a,\mathfrak{cc},\mathfrak{as}} l'$, *where* l *is a source location,* l' *is a target location,* a *is an action,* $\mathfrak{cc}$ *is the enabling condition for* e, *and* $\mathfrak{as}$ *is the assignment for* e.

In order to reason about systems represented by timed automata, for a set of propositional variables $\mathcal{PV}$, we define a valuation function $V_{\mathcal{TA}} : L \to 2^{\mathcal{PV}}$, which assigns propositions to the locations.

An *instantaneous state* of a $\mathcal{TA}$ is a pair (l, v), where $l \in L$ and $v \in \mathbb{R}_+^{|\mathcal{X}|}$ is a clock valuation. The *dense state space* of a $\mathcal{TA}$ is a structure $\mathrm{D}(\mathcal{TA}) = (Q, q^0, \to)$, where $Q = L \times \mathbb{R}_+^{|\mathcal{X}|}$ is the set of all the instantaneous states, $q^0 = (l^0, v^0)$ with $v^0(x) = 0$ for all $x \in \mathcal{X}$ is the initial state, and $\to \subseteq Q \times (\mathfrak{Z} \cup \mathbb{R}_+) \times Q$ is the transition relation, defined by action- and time-successors as follows:

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

For $(l,v) \in Q$, let $(l,v)+\delta$ denote $(l, v+\delta)$. A *q-run* ρ of a $\mathcal{TA}$ is a sequence of instantaneous states: $q_0 \xrightarrow{\delta_0} q_0+\delta_0 \xrightarrow{a_0} q_1 \xrightarrow{\delta_1} q_1+\delta_1 \xrightarrow{a_1} q_2 \xrightarrow{\delta_2} \ldots$, where $q_0 = q \in Q$, $a_i \in \mathfrak{Z}$ and $\delta_i \in \mathbb{R}_+$ for each $i \geq 0$. A run ρ is said to be *progressive* iff $\Sigma_{i \in \mathbb{N}} \delta_i$ is unbounded. A $\mathcal{TA}$ is *progressive* if all its runs are progressive. For simplicity of presentation, we consider only progressive timed automata. Note that progressiveness can be checked as in [33].

2.2 Parallel Composition of Timed Automata

In general, we will model a multi-agent system by taking several timed automata running in parallel and communicating with each other. These concurrent timed automata can be composed into a global timed automaton as follows: the transitions of the timed automata that do not correspond to a shared action are interleaved, whereas the transitions labelled with a shared action are synchronised.

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

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

Definition 2 (Parallel Composition). The parallel composition *of m timed automata $\mathcal{TA}_i$ is a timed automaton* $\mathcal{TA} = (\mathfrak{Z}, L, l^0, E, \mathrm{X}, \mathfrak{I})$, *where* $\mathfrak{Z} = \bigcup_{i=1}^m \mathfrak{Z}_i$, $L = \prod_{i=1}^m L_i$, $l^0 = (l_1^0, \ldots, l_m^0)$, $\mathcal{X} = \bigcup_{i=1}^m \mathcal{X}_i$, $\mathfrak{I}(l_1, \ldots, l_m) = \bigwedge_{i=1}^m \mathfrak{I}_i(l_i)$, *and a transition* $((l_1, \ldots, l_m), a, \mathfrak{cc}, \mathfrak{as}, (l_1', \ldots, l_m')) \in E$ *iff* $(\forall i \in \mathfrak{Z}(a))\ (l_i, a, \mathfrak{cc}_i, \mathfrak{as}_i, l_i') \in E_i$, $\mathfrak{cc} = \bigwedge_{i \in \mathfrak{Z}(a)} \mathfrak{cc}_i$, $\mathfrak{as} = \bigcup_{i \in \mathfrak{Z}(a)} \mathfrak{as}_i$, *and* $(\forall j \in \{1, \ldots, m\} \setminus \mathfrak{Z}(a))\ l_j' = l_j$.

Note that in the above any automaton is allowed to set a value of any clock, including the ones associated with other agents.

Let PV_i be a set of propositional variables containing the symbol **true**, $V_{\mathcal{TA}_i} : L_i \to 2^{\mathcal{PV}_i}$ be a valuation function for the ith automaton, where $i \in \{1, \ldots, m\}$, and $\mathcal{PV} = \bigcup_{i=1}^m \mathcal{PV}_i$. Then, the valuation function $V_{\mathcal{TA}} : L \to 2^{\mathcal{PV}}$ for the parallel composition of m timed automata, is defined as follows $V_{\mathcal{TA}}((l_1, \ldots, l_m)) = \bigcup_{i=1}^m V_{\mathcal{TA}_i}(l_i)$.

2.3 Real Time Deontic Interpreted System

In line with much literature in multi-agent systems, we use interpreted systems as a semantics for a temporal epistemic language. For this, we need to adapt them to work on real time: this is why we take timed automata as the underlying

modelling concept (as opposed to the standard protocols of interpreted systems). To define *real time deontic interpreted systems*, we first partition the set of clock valuations as in [1].

Let $\mathcal{TA}$ be a timed automaton, $\mathcal{C}(\mathcal{TA}) \subseteq \mathcal{C}(\mathcal{X})$ be a non-empty set containing all the clock constrains occurring in any enabling condition used in the transition relation E or in a state invariant of $\mathcal{TA}$. Moreover, let c_{max} be the largest constant appearing in $\mathcal{C}(\mathcal{TA})$. For $\sigma \in \mathbb{R}$, $frac(\sigma)$ denotes the fractional part of σ, and $\lfloor \sigma \rfloor$ denotes its integral part.

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

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

The equivalence classes of the relation $\simeq$ are called *zones*, and denoted by Z, Z' and so on.

Now we are ready to define a *Real Time Deontic Interpreted System*, which will be for the logic presented in the next section semantics.

Let $\mathcal{AG}$ be a set of m agents, where each agent is modelled by a timed automaton $\mathcal{TA}_i = (\mathfrak{Z}_i, L_i, l_i^0, E_i, \mathcal{X}_i, \mathfrak{I}_i)$, for $i \in \{1, \ldots, m\}$. Moreover, assume, in line with [23, 24], that for every agent, its set L_i of local locations is partitioned into "allowed", denoted by $\mathcal{G}_i$, and "disallowed" locations, denoted by $\mathcal{R}_i$ and defined by $\mathcal{R}_i = L_i \setminus \mathcal{G}_i$. We shall call these locations *green* and *red* respectively. Further, assume that the parallel composition $\mathcal{TA} = (\mathfrak{Z}, L, l^0, E, \mathrm{X}, \mathfrak{I})$ of all the agents is given[1], and that $l_i : Q \to L_i$ is a function that returns the location of agent i from a global state. Then, a *real time deontic interpreted system* is defined as follows.

Definition 4 (Real Time Deontic Interpreted System). *A* real time deontic interpreted system *is a tuple $M_c = (\mathrm{D}(\mathcal{TA}), \sim_1^K, \ldots, \sim_m^K, R_1^O, \ldots, R_m^O, \mathcal{V}_c)$, where*

- *$\mathrm{D}(\mathcal{TA})$ is a dense state space for $\mathcal{TA}$.*
- *$\sim_i^K \subseteq Q \times Q$ is a relation defined by $(l, v) \sim_i^K (l', v')$ iff $l_i((l, v)) = l_i((l', v'))$ and $v \simeq v'$, for each agent i. Obviously $\sim_i^K$ is an equivalence relation.*
- *$R_i^O \subseteq Q \times Q$ is a relation defined by $(l, v) R_i^O (l', v')$ iff $l_i((l', v')) \in \mathcal{G}_i$, for each agent i.*
- *$V_c : Q \to 2^{\mathcal{PV}}$ is a valuation function that extends $V_{\mathcal{TA}}$ as follows $V_c((l, v)) = V_{\mathcal{TA}}(l)$, i.e., V_c assigns the same propositions to the states with the same locations.*

[1] Note that the set L, which defines all the possible *global locations*, is defined as the Cartesian product $L_1 \times \ldots \times L_m$, such that $L_1 \supseteq \mathcal{G}_1, \ldots, L_m \supseteq \mathcal{G}_m$.

3 The Logic TCTLKD

In this section, we formally present the syntax and semantics of a *real time computation tree logic for knowledge and correctness* (TCTLKD), which extends the standard TCTL [1], the logic for real time, by means of modal operators for knowledge and correctness.

The language generalises classical propositional logic, and thus it contains the standard propositional connectives $\neg$ (not) and $\vee$ (or); the remaining connectives ($\wedge$ (and), $\rightarrow$ (implies), $\leftrightarrow$ (if, and only if)) are assumed to be introduced as abbreviations in the usual way. With respect to real time temporal connectives, we take as primitives U_I (for "until within interval I"), and G_I (for "always within interval I"); the remaining operators (F_I (for "eventually within interval I") and R_I (for "release within interval I")) are assumed to be introduced as abbreviations in the usual way. The language also contains two path quantifiers: A (for "for all the runs") and E (for "there exists a run"). Further, we assume a set $\mathcal{AG}$ of m agents, and we use the indexed modalities K_i, $\mathcal{O}_i$, and $\hat{\mathrm{K}}_i^j$ to represent the knowledge of agent i, the correct functioning circumstances of agent i, and the knowledge of agent i under assumption of correct functioning of agent j, respectively. Furthermore, we use the indexed modalities D_Γ, C_Γ to represent distributed and common knowledge in a group of agents $\Gamma \subseteq \mathcal{AG}$, and we use the operator E_Γ to represent the concept "everybody in Γ knows".

3.1 Syntax of TCTLKD

We assume a set $\mathcal{PV}$ of propositional variables, and a finite set $\mathcal{AG}$ of m agents. Furthermore, let I be an interval in $\mathbb{R}_+$ with integer bounds of the form $[n, n']$, $[n, n')$, $(n, n']$, (n, n'), (n, ∞), and $[n, \infty)$, for $n, n' \in \mathbb{N}$. The set of TCTLKD formulas is defined inductively as follows:

- every member p of $\mathcal{PV}$ is a formula,
- if α and β are formulas, then so are $\neg\alpha$, $\alpha \vee \beta$, $\mathrm{EG}_I\alpha$, and $\mathrm{E}(\alpha \mathrm{U}_I \beta)$,
- if α is formula, then so are $\mathrm{K}_i\alpha$, $\hat{\mathrm{K}}_i^j\alpha$, and $\mathcal{O}_i\alpha$, for $i, j \in \mathcal{AG}$,
- if α is formula, then so are $\mathrm{D}_\Gamma\alpha$, $\mathrm{C}_\Gamma\alpha$, and $\mathrm{E}_\Gamma\alpha$, for $\Gamma \subseteq \mathcal{AG}$.

The other basic temporal, epistemic,and correctness modalities are defined as follows:

- $\mathrm{EF}_I\varphi \stackrel{def}{=} \mathrm{E}(\mathbf{true}\mathrm{U}_I\varphi)$,
- $\mathrm{AF}_I\varphi \stackrel{def}{=} \neg\mathrm{EG}_I(\neg\varphi)$,
- $\mathrm{AG}_I\varphi \stackrel{def}{=} \neg\mathrm{EF}_I(\neg\varphi)$,
- $\mathrm{A}(\alpha \mathrm{U}_I \beta) \stackrel{def}{=} \neg\mathrm{E}(\neg\beta \mathrm{U}_I(\neg\beta \wedge \neg\alpha)) \wedge \neg\mathrm{EG}_I(\neg\beta)$,
- $\mathrm{A}(\alpha \mathrm{R}_I \beta) \stackrel{def}{=} \neg\mathrm{E}(\neg\alpha \mathrm{U}_I \neg\beta)$,
- $\mathrm{E}(\alpha \mathrm{R}_I \beta) \stackrel{def}{=} \neg\mathrm{A}(\neg\alpha \mathrm{U}_I \neg\beta)$,
- $\overline{\mathrm{K}}_i\alpha \stackrel{def}{=} \neg\mathrm{K}_i\neg\alpha$, $\overline{\mathcal{O}}_i\alpha \stackrel{def}{=} \neg\mathcal{O}_i\neg\alpha$, $\overline{\hat{\mathrm{K}}_i^j}\alpha \stackrel{def}{=} \neg\hat{\mathrm{K}}_i^j\neg\alpha$,
- $\overline{\mathrm{D}}_\Gamma\alpha \stackrel{def}{=} \neg\mathrm{D}_\Gamma\neg\alpha$, $\overline{\mathrm{C}}_\Gamma\alpha \stackrel{def}{=} \neg\mathrm{C}_\Gamma\neg\alpha$, $\overline{\mathrm{E}}_\Gamma\alpha \stackrel{def}{=} \neg\mathrm{E}_\Gamma\neg\alpha$.

3.2 Semantics of TCTLKD

Let $\mathcal{AG}$ be a set of m agents, where each agent is modelled by a timed automaton $\mathcal{TA}_i = (\mathfrak{Z}_i, L_i, l_i^0, E_i, \mathcal{X}_i, \mathfrak{I}_i)$, for $i = \{1, \ldots, m\}$, $\mathcal{TA} = (\mathfrak{Z}, L, l^0, E, \mathrm{X}, \mathfrak{I})$ be their parallel composition, and $M_c = (Q, q^0, \rightarrow, \sim_1^K, \ldots, \sim_m^K, R_1^O, \ldots, R_m^O, \mathcal{V}_c)$ be a *real time deontic interpreted system*. Moreover, let $\rho = q_0 \stackrel{\delta_0}{\rightarrow} q_0 + \delta_0 \stackrel{a_0}{\rightarrow} q_1 \stackrel{\delta_1}{\rightarrow} q_1 + \delta_1 \stackrel{a_1}{\rightarrow} q_2 \stackrel{\delta_2}{\rightarrow} \ldots$ be a run of $\mathcal{TA}$ such that $\delta_i > 0$ for $i \in \mathbb{N}$, and let $f_{\mathcal{TA}}(q)$ denote the set of all such q-runs of $\mathcal{TA}$. In order to give a semantics to TCTLKD, we introduce the notation of a *dense path π_ρ corresponding to run ρ. A dense path π_ρ corresponding to ρ* is a mapping from $\mathbb{R}$ to a set of states such that $\pi_\rho(r) = s_i + \delta$ for $r = \Sigma_{j=0}^{i}\delta_j + \delta$ with $i \geq 0$ and $0 \leq \delta < \delta_i$. Moreover, as usual, we define the following epistemic relations: $\sim_\Gamma^E = \bigcup_{i\in\Gamma} \sim_i$, $\sim_\Gamma^C = (\sim_\Gamma^E)^+$ (the transitive closure of $\sim_\Gamma^E$), and $\sim_\Gamma^D = \bigcap_{i\in\Gamma} \sim_i$, where $\Gamma \subseteq \mathcal{AG}$.

Definition 5 (Satisfaction of TCTLKD**).**
Let $M_c, q \models \alpha$ denote that α is true at state s in the model M_c. M_c is omitted, if it is implicitly understood. The relation $\models$ is defined inductively as follows:

$q_0 \models p$ *iff* $p \in V_c(q_0)$,
$q_0 \models \neg\varphi$ *iff* $q_0 \not\models \varphi$,
$q_0 \models \varphi \vee \psi$ *iff* $q_0 \models \varphi$ *or* $q_0 \models \psi$,
$q_0 \models \varphi \wedge \psi$ *iff* $q_0 \models \varphi$ *and* $q_0 \models \psi$,
$q_0 \models \mathrm{E}(\varphi \mathrm{U}_I \psi)$ *iff* $(\exists\, \rho \in f_{\mathcal{TA}}(q_0))(\exists r \in I)\Big[\pi_\rho(r) \models \psi$ *and* $(\forall r' < r)\ \pi_\rho(r') \models \varphi\Big]$,
$q_0 \models \mathrm{EG}_I\varphi)$ *iff* $(\exists\, \rho \in f_{\mathcal{TA}}(q_0))(\forall r \in I)\ \pi_\rho(r) \models \varphi$,
$q_0 \models \mathrm{K}_i\alpha$ *iff* $(\forall q' \in Q)((q_0 \sim_i^K q')$ *implies* $q' \models \alpha)$,
$q_0 \models \mathcal{O}_i\alpha$ *iff* $(\forall q' \in Q)(q_0 R_i^O q')$ *implies* $q' \models \alpha)$,
$q_0 \models \hat{\mathrm{K}}_i^j\alpha$ *iff* $(\forall q' \in Q)((q_0 \sim_i^K q'$ *and* $q_0 R_j^O q')$ *implies* $q' \models \alpha)$,
$q_0 \models \mathrm{D}_\Gamma\alpha$ *iff* $(\forall q' \in Q)((q_0 \sim_\Gamma^D q')$ *implies* $q' \models \alpha)$,
$q_0 \models \mathrm{E}_\Gamma\alpha$ *iff* $(\forall q' \in Q)((q_0 \sim_\Gamma^E q')$ *implies* $q' \models \alpha)$,
$q_0 \models \mathrm{C}_\Gamma\alpha$ *iff* $(\forall q' \in Q)((q_0 \sim_\Gamma^C q')$ *implies* $q' \models \alpha)$.

Intuitively, the formula $\mathrm{E}(\alpha \mathrm{U}_I \beta)$ holds at a state q_0 in a real time deontic interpreted system M_c if there exists a run starting at q_0 such that β holds in some state in time interval I, and until then α always holds. The formula $\mathrm{EG}_I\alpha$ holds at a state q_0 in a real time deontic interpreted system M_c if there exists a run starting at q_0 such that α holds in all the states on the run in time interval I. The formula $\mathrm{K}_i\alpha$ holds at state q_0 in a real time deontic interpreted system M_c if α holds at all the states that are indistinguishable for agent i from q_0. The formula $\mathcal{O}_i\alpha$ holds at state q_0 in a real time deontic interpreted system M_c if α holds at all the states where agent i is functioning correctly. The formula $\hat{\mathrm{K}}_i^j\alpha$ holds at state q_0 in a real time deontic interpreted system M_c if α holds at all the states that agent i is unable to distinguish from the actual state q_0, and in which agent j is functioning correctly. The formula $\mathrm{E}_\Gamma\alpha$ holds at state q_0 in a real time deontic interpreted system M_c if α is true in all the states that the group Γ of agents is unable to distinguish from the actual state q_0. Note that $\mathrm{E}_\Gamma\alpha$ can be defined by $\bigwedge_{i\in\Gamma} \mathrm{K}_i\alpha$. The formula $\mathrm{C}_\Gamma\alpha$ is equivalent to the infinite

conjunction of the formulas $\mathrm{E}_{\Gamma}^{k}\alpha$ for $k \geq 1$. So, $C_{\Gamma}\alpha$ holds at state q_0 in a real time deontic interpreted system M_c if everyone knows α holds at q_0, everyone knows that everyone knows α holds at q_0, etc. The formula $\mathrm{D}_{\Gamma}\alpha$ holds at state q_0 in a real time deontic interpreted system M_c if the "combined" knowledge of all the agents in Γ implies α. We refer to [1, 11, 23] for more details on the operators above.

A TCTLKD formula φ is *satisfiable* if there exists a real time deontic interpreted system $M_c = (Q, q^0, \rightarrow, \sim_1^K, \ldots, \sim_m^K, R_1^O, \ldots, R_m^O, \mathcal{V}_c)$ and a state q of M_c, such that $M_c, q \models \varphi$. A TCTLKD formula φ is *valid in* M_c (denoted $M_c \models \varphi$) if $M_c, q^0 \models \varphi$, i.e., φ is true at the initial state of the model M_c.

Note that the "full" logic of real time (TCTL) is undecidable [1]. Since real time deontic interpreted systems can be shown to be as expressive as the semantics in [1], and the fusion [7] between TCTL, $S5$ for knowledge [11], and $KD45^{i-j}$ for the deontic dimension [23] is a proper extension of TCTL, it follows that problem of satisfiability for the TCTLKD logic will be also undecidable. Still, it is easy to observe that given a TCTLKD formula φ and a real time deontic interpreted system M_c, the problem of deciding whether $M_c \models \varphi$ is decidable. This result is our motivation for introducing TCTLKD. We are not interested in using the whole class of real time deontic interpreted systems, but only to study particular examples by means of this logic. We exemplify this in the next section.

4 Applications

One of the motivations for developing the formalism presented in this paper is that we would like to be able to analyse what epistemic and temporal properties hold, when agents follow or violate their specifications while operating on real time.

As an example of this we discuss the *Railroad Crossing System* (RCS) [20], a well-known example in the literature of real-time verification. Here we analyse the scenario not only by means of temporal operators but also by means of epistemic and correctness modalities. The system consists of three agents: Train, Gate, and Controller running in parallel and synchronising through the events: "*approach*", "*exit*", "*lower*", and "*raise*".

Let us start by considering what we call the *correct RCS*, as modelled by timed automata (Figure 1). The correct RCS operates as follows. When Train approaches the crossing, it sends an *approach* signal to Controller, and enters the crossing between 300 and 500 seconds from this event. When Train leaves the crossing, it sends an *exit* signal to Controller. Controller sends a signal *lower* to Gate exactly 100 seconds after the *approach* signal is received, and sends a *raise* signal within 100 seconds after *exit*. Gate performs the transition *down* within 100 seconds of receiving the request *lower*, and responds to *raise* by moving *up* between 100 and 200 seconds.

Assume the following set of propositional variables: $\mathcal{PV} = \{\mathfrak{p}, \mathfrak{q}, \mathfrak{r}, \mathfrak{s}\}$. The proposition $\mathfrak{p}$ represents the fact that an approach signal was sent by Train,

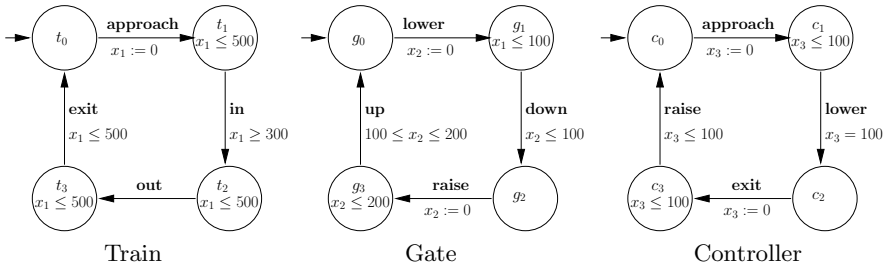

Fig. 1. Agents Train, Gate, and Controller for the correct RCS system.

$\mathfrak{q}$ that Train is on the cross, $\mathfrak{r}$ that Gate is down, and $\mathfrak{s}$ that Controller sent the signal *lower* to Gate. A real time deontic interpreted system M_{RCS} can be associated with the correct RCS as follows. For the sets $L_1 = \{t_0, t_1, t_2, t_3\}$, $L_2 = \{g_0, g_1, g_2, g_3\}$, and $L_3 = \{c_0, c_1, c_2, c_3\}$ of locations for Train, Gate, and Controller respectively, the set of "green" locations and the dense state space for RCS are defined by $G_1 = L_1$, $G_2 = L_2$, $G_3 = L_3$, and $D(RCS) = L_1 \times L_2 \times L_3 \times R^3$, respectively. The valuation functions for Train ($V_{Train} : L_1 \to 2^{\mathcal{PV}}$), Gate ($V_{Gate} : L_2 \to 2^{\mathcal{PV}}$), and Controller ($V_{Cont} : L_3 \to 2^{\mathcal{PV}}$) are defined as follows:

- $V_{Train}(t_1) = \{\mathfrak{p}\}$, $V_{Train}(t_2) = \{\mathfrak{q}\}$, and $V_{Train}(t_0) = V_{Train}(t_3) = \emptyset$.
- $V_{Gate}(g_2) = \{\mathfrak{r}\}$, and $V_{Gate}(g_0) = V_{Gate}(g_1) = V_{Gate}(g_3) = \emptyset$.
- $V_{Cont}(c_2) = \{\mathfrak{s}\}$, and $V_{Cont}(c_0) = V_{Cont}(c_1) = V_{Cont}(c_3) = \emptyset$.

The valuation function $V_{RCS} : L_1 \times L_2 \times L_3 \to 2^{\mathcal{PV}}$, for the RCS system, is built as follows: $V_{RCS}(l) = V_{Train}(l_1) \cup V_{Gate}(l_2) \cup V_{Cont}(l_3)$, for all $l = (l_1, l_2, l_3) \in L_1 \times L_2 \times L_3$. Thus, according to the definition of the real time deontic interpreted system, the valuation function $V_{M_{RCS}} : L_1 \times L_2 \times L_3 \times R^3 \to 2^{\mathcal{PV}}$ of M_{RCS} is defined by $V_{M_{RCS}}(l, v) = V_{RCS}(l)$.

Using the TCTLKD logic, we can specify properties of the correct RCS system that cannot be specified by standard propositional temporal epistemic logic. For example, we consider the following:

$$\mathrm{AG}_{[0,\infty]}(\mathfrak{p} \to \mathrm{K}_{Controller}(\mathrm{AF}_{[300,\infty]}\mathfrak{q})) \tag{1}$$

$$\mathrm{AG}_{[0,\infty]}\mathrm{K}_{Train}(\mathfrak{p} \to \mathrm{AF}_{[0,200]}\mathfrak{r}) \tag{2}$$

$$\mathrm{K}_{Controller}(\mathfrak{s} \to \mathrm{AF}_{[0,100]}\mathfrak{r}) \tag{3}$$

Formula (1) states that forever in the future if an approach signal is sent by agent Train, then agent Controller knows that in some point after 300 seconds later Train will enter the cross. Formula (2) states that forever in the future agent Train knows that, if it sends an approach signal, then agent Gate will send the signal down within 200 seconds. Formula (3) states that agent Controller knows that if it sends an lower signal, then agent Gate will send the signal down within 100 seconds.

All the formulas above can be shown to hold on M_{RCS} on the initial state. We can also check that the following properties do not hold on M_{RCS}.

$$\mathrm{AG}_{[0,\infty]}(\mathfrak{p} \rightarrow \mathrm{K}_{Controller}(\mathrm{AF}_{[0,300]}\mathfrak{q})) \tag{4}$$

$$\mathrm{K}_{Train}(\mathrm{AG}_{[0,\infty)}\mathrm{EF}_{[10,90]}\mathfrak{s}) \tag{5}$$

$$\mathrm{K}_{Controller}(\mathfrak{s} \rightarrow \mathrm{AF}_{[0,50]}\mathfrak{r}) \tag{6}$$

Formula (4) states that forever in the future if an approach signal is sent by agent Train, then agent Controller knows that at some point in the future within 300 seconds Train will enter the crossing. Formula (5) states that agent Train knows that always in the future it is possible that within interval $[10, 90]$ the gate will be down. Formula (6) states that agent Controller knows that if it sends the lower signal, then agent Gate will send the signal down within 50 seconds.

Let us now consider a variant of the RCS system described above, and let us assume that agent Controller is faulty. Let us assume that because of a fault the signal *lower* may not be sent in the specified interval, and the transition to the faulty state $\overline{c_2}$ my be triggered. We are allowing for Controller to recover from the fault once in $\overline{c_2}$ by means of the action *lower* (see Figure 2).

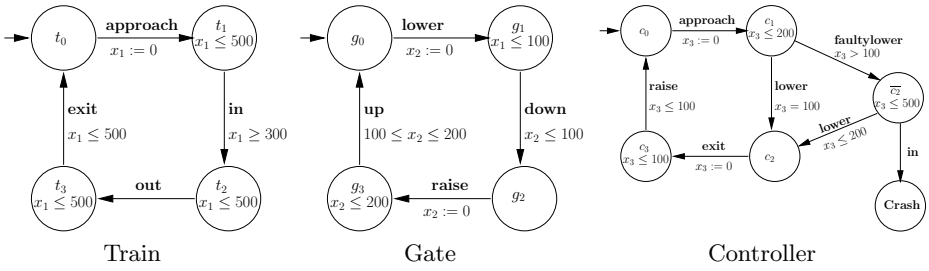

Fig. 2. Agents Train, Gate, and Controller for the faulty RCS system.

We examine the scenario by considering the following set of propositional variables: $\mathcal{PV} = \{\mathfrak{p}, \mathfrak{q}, \mathfrak{r}, \mathfrak{s}, \mathfrak{crash}\}$. The propositions $\mathfrak{p}$, $\mathfrak{q}$, $\mathfrak{r}$, and, $\mathfrak{s}$ have the same meaning as in the case of the correct RCS system; the proposition $\mathfrak{crash}$ represents the fact that Train is on the cross and Gate is still open. A real time deontic interpreted system M_{RCS} can be associated with the faulty RCS system as follows[2].

For the sets $L_1 = \{t_0, t_1, t_2, t_3\}$, $L_2 = \{g_0, g_1, g_2, g_3\}$, and $L_3 = \{c_0, c_1, c_2, c_3, \overline{c_2}, crash\}$ of locations for Train, Gate, and Controller, the set of "green" locations are defined by $G_1 = L_1$, $G_2 = L_2$, $G_3 = \{c_0, c_1, c_2, c_3\}$, respectively. The

[2] Note that the names of the mathematical objects we use to represent the faulty RCS are the same as the ones employed previously for the correct RCS. Given that these appear in different contexts we trust no confusion arises.

dense state space for RCS is defined by $D(RCS) = L_1 \times L_2 \times L_3 \times R^3$. The valuation functions for Train (V_{Train}), Gate (V_{Gate}), and Controller (V_{Cont}) are defined as follows:

- $V_{Train} : L_1 \to 2^{\mathcal{PV}}$, and $V_{Train}(t_1) = \{\mathfrak{p}\}$, $V_{Train}(t_2) = \{\mathfrak{q}\}$, and $V_{Train}(t_0) = V_{Train}(t_3) = \emptyset$.
- $V_{Gate} : L_2 \to 2^{\mathcal{PV}}$, and $V_{Gate}(g_0) = V_{Gate}(g_1) = V_{Gate}(g_3) = \emptyset$, and $V_{Gate}(g_2) = \{\mathfrak{r}\}$.
- $V_{Cont} : L_3 \to 2^{\mathcal{PV}}$, and $V_{Cont}(c_0) = V_{Cont}(c_1) = V_{Cont}(c_3) = V_{Cont}(\overline{c_2}) = \emptyset$, $V_{Cont}(c_2) = \{\mathfrak{s}\}$, and $V_{Cont}(crash) = \{\mathfrak{crash}\}$.

The valuation functions $V_{RCS} : L_1 \times L_2 \times L_3 \to 2^{\mathcal{PV}}$, and $V_{M_{RCS}} : L_1 \times L_2 \times L_3 \times R^3 \to 2^{\mathcal{PV}}$ are defined in the same way as in the correct version of the RCS system.

Using TCTLKD, we can specify the following properties of the faulty RCS system. These can be checked to hold on the real time deontic interpreted system for the faulty RCS.

$$\mathrm{AG}_{[0,\infty]}\mathrm{K}_{Train}\mathcal{O}_{Controller}(\mathfrak{p} \to \mathrm{AF}_{[0,200]}\mathfrak{r}) \tag{7}$$

$$\mathrm{K}_{Train}\mathcal{O}_{Controller}(\mathfrak{p} \to \mathrm{AF}_{[0,200]}\mathfrak{r}) \tag{8}$$

$$\hat{\mathrm{K}}^{Controller}_{Train}(\mathfrak{p} \to \mathrm{AF}_{[0,200]}\mathfrak{r}) \tag{9}$$

$$\mathrm{AG}_{[0,\infty]}\mathrm{K}_{Train}\mathcal{O}_{Controller}(\neg\mathfrak{crash}) \tag{10}$$

$$\mathrm{AG}_{[0,\infty]}\hat{\mathrm{K}}^{Controller}_{Train}(\neg\mathfrak{crash}) \tag{11}$$

$$\mathrm{AG}_{[0,\infty]}\hat{\mathrm{K}}^{Controller}_{Train}(\mathfrak{p} \to \mathrm{AF}_{[0,100]}\mathfrak{s}) \tag{12}$$

Formula (7) states that forever in the future agent Train knows that whenever agent Controller is functioning correctly, if Train sends the *approach* signal, then agent Gate will send the signal down within 200 seconds. Formula (8) states that agent Train knows that whenever agent Controller is functioning correctly, if the *approach* signal was sent by Train, then at some point in the future, within 200 second, Gate will be down. Formula (9) states that agent Train knows that under the assumption of agent Controller functioning correctly, if the *approach* signal was sent by Train, then at some point in the future, within 200 second, Gate will be down. Formula (10) states that always in the future agent Train knows that whenever agent Controller is functioning correctly under no circumstances there will be a situation in which Train is on the crossing and Gate is open. Formula (11) states that always in the future agent Train knows that under the assumption of agent Controller functioning correctly, under no circumstances there will be a situation in which Train is on the crossing and Gate is open. Formula (12) states that always in the future agent Train knows that under the assumption of agent Controller functioning correctly, if the *approach* signal was sent by Train, then at some point in the future, within 100 second, the signal *lower* will be sent by Controller.

The following formulas can be checked not to hold on the faulty RCS.

$$\mathrm{K}_{Train}(\mathfrak{p} \to \mathrm{AF}_{[0,200]}\mathfrak{r}) \tag{13}$$

$$\mathrm{AG}_{[0,\infty]}\mathrm{K}_{Train}(\neg \mathfrak{crash}) \tag{14}$$

$$\mathrm{AG}_{[0,\infty]}\mathrm{K}_{Train}(\mathfrak{p} \rightarrow \mathrm{AF}_{[0,100]}\mathfrak{s}) \tag{15}$$

The formula (13) states that agent Train knows that, if it sends the *approach* signal, then at some point in the future, within 200 second, Gate will be down. The formula (14) states that always in the future agent Train knows that under no circumstances there will be a situation where Train is on the cross and Gate is open. The formula (15) states that always in the future agent Train knows that, if it sends the *approach* signal, then at some point in the future, within 100 second, the signal *lower* will be sent by Controller.

5 Conclusions

In the paper we have proposed TCTLKD, a real time logic for knowledge and correctness. TCTLKD is a fusion of three well known logics: TCTL for real time [1], S5 for knowledge [11], and $\mathrm{KD45}^{i-j}$ for the correctness dimension [23].

Previous attempts of combinations of real time and knowledge have included [3, 8, 32]. In [3] a technique for determining the temporal validity of shared data in real-time distributed systems is proposed. The approach is based on a language consisting of Boolean, epistemic, dynamic, and real-time temporal operators, but the semantics for these is not defined. In [8] a fusion of the branching time temporal logic (CTL) and the standard epistemic logic is presented. The semantics of the logic is given over an interpreted system defined like in [11] with the difference of using runs defined from real numbers. This language is used to establish sound and complete termination conditions for motion planning of robots, given initial and goal states. [32] presents a framework for knowledge-based analysis of clocks synchronisation in systems with real-time constraints. In that work a relation of timed precedence as a generalisation of previous work by Lamport's is defined, and it is shown how (inherent) knowledge about timed precedences can be applied to synchronise clocks optimally. Like in [8], the semantics consists of runs that are functions over the real time. The epistemic relations defined in this work assume that agents have perfect recall.

Our paper differs from the approaches above by considering quantitative temporal operators such as $\mathrm{EF}_{[0,10]}$ (meaning "possibly within 10 time units"), rather than qualitative operators EF (meaning "possibly in the future", but with no bound), and by not forcing the agents to have perfect recall. In addition, the logic TCTLKD also incorporates a notion of correctness of execution with respect to specifications, a concept not tackled in previous works, and associates a set of clocks to every agent not just to the system as a whole. While the satisfiability problem for TCTLKD is undecidable, the TCTLKD model checking problem, i.e., the problem of validity in a given model, is decidable. Given this, it seems worthwhile to develop model checking methods for TCTLKD.

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. S. Anderson and J. Kuster-Filipe. Guaranteeing temporal validity with a real-time logic of knowledge. In *In Proceedings of the 1st International Workshop on Data Distribution for Real-Time Systems (DDRTS'03), ICDCS 2003 Workshop*, pages 178–183, Providence, Rhode Island, USA, May 2003.
4. R. J. Aumann. Agreeing to disagree. *Annals of Statistics*, 4(6):1236–1239, 1976.
5. A. Baltag, L. S. Moss, and S. Solecki. The logic of public announcement, common knowledge, and private suspicions. In I. Gilboa, editor, *Proceedings of the 7th Conference on Theoretical Aspects of Rationality and Knowledge (TARK-98)*, pages 125–132, San Francisco, 1998. Morgan Kaufmann.
6. M. Ben-Ari, A. Pnueli, and Z. Manna. The temporal logic of branching time. *Acta Informatica*, 20(3):207–226, 1983.
7. P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*, volume 53 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2001.
8. 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.
9. E. A. Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, chapter 16, pages 996 – 1071. Elsevier Science Publishers, 1990.
10. E. A. Emerson and J. Y. Halpern. Decision procedures and expressiveness in the temporal logic of branching time. *Journal of Computer and System Sciences*, 30(1):1–24, 1985.
11. R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning about Knowledge*. MIT Press, Cambridge, 1995.
12. R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. Knowledge-based programs. *Distributed Computing*, 10(4):199–225, 1997.
13. R. Fagin and M. Y. Vardi. Knowledge and implicit knowledge in a distributed environment: Preliminary report. In J. Y. Halpern, editor, *TARK: Theoretical Aspects of Reasoning about Knowledge*, pages 187–206, San Francisco (CA), 1986. Morgan Kaufmann.
14. 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.
15. J. Halpern and Y. Moses. A guide to completeness and complexity for modal logics of knowledge and belief. *Artificial Intelligence*, 54:319–379, 1992.
16. J. Y. Halpern and M. Y. Vardi. The complexity of reasoning about knowledge and time. In *ACM Symposium on Theory of Computing (STOC '86)*, pages 304–315, Baltimore, USA, 1986. ACM Press.
17. J. Y. Halpern and M. Y. Vardi. The complexity of reasoning about knowledge and time 1: lower bounds. *Journal of Computer and System Sciences*, 38(1):195–237, 1989.
18. J. Hintikka. *Knowledge and Belief, An Introduction to the Logic of the Two Notions*. Cornell University Press, Ithaca (NY) and London, 1962.

19. W. van der Hoek. Sytems for knowledge and belief. *Journal of Logic and Computation*, 3(2):173–195, 1993.
20. I. Kang and I. Lee. An efficient state space generation for the analysis of real-time systems. In *Proceedings of International Symposium on Software Testing and Analysis*, 1996.
21. A. Lomuscio, R. van der Meyden, and M. Ryan. Knowledge in multi-agent systems: Initial configurations and broadcast. *ACM Transactions of Computational Logic*, 1(2), 2000.
22. A. Lomuscio and M. Ryan. An algorithmic approach to knowledge evolution. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing (AIEDAM)*, 13(2), 1999.
23. A. Lomuscio and M. Sergot. Deontic interpreted systems. *Studia Logica*, 75(1):63–92, 2003.
24. A. Lomuscio and M. Sergot. Violation, error recovery, and enforcement in the bit transmission problem. *Journal of Applied Logic*, 1, 2003.
25. Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems: Specification*. Springer-Verlag, 1991.
26. Z. Manna and A. Pnueli. *Temporal Verification of Reactive Systems: Safety*. Springer-Verlag, 1995.
27. Zohar Manna and Amir Pnueli. Completing the temporal picture. In *Selected papers of the 16th international colloquium on Automata, languages, and programming*, pages 97–130. Elsevier Science, 1991.
28. W. Marek and M. Truszczyński. Autoepistemic logic. *ACM*, 38(3):587–618, 1991.
29. R. van der Meyden. Axioms for knowledge and time in distributed systems with perfect recall. In *Proceedings, Ninth Annual IEEE Symposium on Logic in Computer Science*, pages 448–457, Paris, France, 1994. IEEE Computer Society Press.
30. R. van der Meyden and K. Wong. Complete axiomatizations for reasoning about knowledge and branching time. *Studia Logica*, 75(1):93–123, 2003.
31. R.C. Moore. Possible-world semantics autoepistemic logic. In *Proceedings of Workshop on Non-Monotonic Reasoning*, pages 344–354. The AAAI Press, 1984.
32. Y. Moses and B. Bloom. Knowledge, timed precedence and clocks. In *Proceedings of the 13th ACM symposium on Principles of Distributed Computing*, pages 274–303. ACM Press, 1994.
33. S. Tripakis and S. Yovine. Analysis of timed systems using time-abstracting bisimulations. *Formal Methods in System Design*, 18(1):25–68, 2001.