A Framework to Reason about the Knowledge of Agents in Continuous Dynamic Systems

Ammar Mohammed
Department of Computer
Arab East College for Graduate Studies, Riyadh, KSA
Department of Computer Science
ISSR- Cairo University, Egypt

Ahmed M. Elmogy
Department of Computer Engineering
Prince Sattam Bin Abdulaziz University, KSA
Computers & Control Eng. Dept.,
Faculty of Engineering
Tanta Universiy, Egypt

***Abstract*—Applying formal methods to a group of agents provides a precise and unambiguous definition of their behaviors, as well as verify properties of agents against implementations. Hybrid automaton is one of the formal approaches that are used by several works to model a group of agents. Several logics have been proposed, as extension of temporal logics to specify and hence verify those quantitative and qualitative properties of systems modeled by hybrid automaton. However, when it comes to agents, one needs to reason about the knowledge of other agents participating in the model. For this purpose, epistemic logic can be used to specify and reason about the knowledge of agents. But this logic assumes that the model of time is discrete. This paper proposes a novel framework that formally specifies and verifies the epistemic behaviors of agents within continuous dynamics. To do so, the paper first extends the hybrid automaton with knowledge. Second, the paper proposes a new logic that extends epistemic logic with quantitative real time requirement. Finally, the paper shows how to specify several properties that can be verified within our framework.**

***Keywords*—Epistemic logic; Reasoning; Hybrid Automata; Agents**

I. Introduction

Multi-agent systems (MAS) consists of several independent agents where the task of some agents may depend on the task of others [1]. Thus, to model intelligent systems, knowledge is an important property to consider.This reasoning has become one of the main concerns in artificial intelligence and MAS [2]. Reasoning is either about the agent itself or about other agents in the MAS. Many efforts from different disciplines have tackled issues involving reasoning about agents' knowledge [3], [4], [5]. To reason about knowledge of agents, several theories and logics have been proposed. Among of them, a logic of knowledge or the so called epistemic logic [6]. Epistemic logic is a type of modal logics [7].The suitability of epistemic logic in a wide range of applications [8] makes it of great importance. The main goal of using epistemic logic in MAS is to model the agents' knowledge either about itself or about other agents. For example, " if $agent_1$ sends a message S to $agent_2$, then eventually, $agent_2$ will know S" , and $agent_1$ knows that $agent_2$ knows S.

To deal with specific type of applications, many techniques have been proposed for extending epistemic logic. Examples of these techniques are the attempts to generate temporal epistemic logics to reason about knowledge changing over time. In order to construct this kind of extension, epistemic logic framework is fused into a kind of temporal logics [9]. Examples of this integration can be seen in[3], [10]. In [3], the epistemic logic is extended with alternating time temporal logic[11]. Also, in [10], the epistemic logic is combined with temporal logic. Temporalization can be seen as an approach for adding a temporal logic on top of another logic and thus having a new logic with temporal features [12]. The importance of reasoning about dynamic knowledge comes from its existence in many of the real life applications. In spite of its suitability in many applications, temporal epistemic logic is still not efficient to be applied in a certain type of applications like robotics and networking. Such type of applications requires a model of time and a model of how the agents' actions are changing through time. Also, knowledge that has time constrains cannot be specified by temporal epistemic logic. An example of time constrained knowledge is $agent_1$ knows that message S will be received within 5 seconds.

According to [13], hybrid systems are systems with combined continuous and discrete state variables. Examples are embedded software in airplanes and medical devices. As hybrid systems involve both the discrete and continuous dynamics of systems, they are naturally used to model many application scenarios [14]. Over the past few decades, hybrid automata [15] have been introduced as a formal model for hybrid systems. Hybrid automata integrate differential equations and finite automata in a single formalism. Generally, the discrete dynamics of hybrid systems are modeled using finite automata whereas the differential equations represent the continuous changes of physical variables. A simplified version of hybrid automaton called timed automaton [16] is also used to model MAS [17]. Timed (finite) automaton is a simple and powerful way used to represent time constrains through real valued clocks [16]. On the other hand, several logics have been introduced in the literature to model the qualitative behaviors of systems. The semantics of many of these logics have been interpreted on the underline operational semantics of hybrid automata.

In spite of the many proposed logics, these logics are not efficient when it comes to express the agents' knowledge about other agents in a continuous dynamic environment. Thus, integrating epistemic logic with qualitative logics of hybrid systems seems to be very appropriate to overcome the shortcomings of the existing logics. However, the epistemic logic assumes only discrete time model which is not adequate when modeling agents' behaviors within continuous dynamics.

Thus, this paper proposes a novel framework that formally specifies and verifies the epistemic behaviors of agents within continuous dynamics. The framework of this paper is two fold. First the hybrid automaton is protracted with knowledge. Second, a new logic is proposed to augment epistemic logic with quantitative real time requirement. Furthermore, the paper shows how to specify several properties that can be verified within the proposed framework.

The rest of this paper is organized as follows. Section II summarizes the related work to the problem of specifying MAS requirements. After that, the concept of hybrid automata is introduced and slightly redefined in section III. The formal syntax and semantics of hybrid automata are also highlighted. Section IV, introduces the hybrid interpreted systems that will be used as the underline semantics of the proposed Logic. Section V defines the syntax and semantics of the proposed logic ERCTL. Section VI illustrates how to specify certain requirements on an example. in section VII, the implementation of the proposed framework using constraint logic program (CLP) is shown. Finally, conclusion and future work are summarized in section VIII.

II. Related Work

The modal logic of knowledge was first introduced by Hintikka in 1962 [18]. According to the logic of Hintikka, knowledge and belief are treated as modalities. The syntax and semantics of this logic was the core for foundation of epistemic logic. Epistemic logic is a type of modal logic concerned with reasoning about knowledge [19], [8]. Generally, epistemic logic in MAS began to get more attention in the early 60^{th} of the previous century. Among several types of epistemic logic, Dynamic Epistemic Logic (DTL) is widely conceived as a logic that is able to model how agents update their knowledge in MAS [20]. As agents in MAS evolve over time, the temporal properties of agentsâ knowledge are of great importance. Thus, Epistemic Temporal Logic (ETL) is employed to model these properties [3], [9], [10]. In [3], the epistemic logic is extended with alternating time temporal logic[11]. In [10], epistemic logic framework is fused into a kind of temporal logics [9]. There is a close relationship between dynamic and temporal epistemic logics. The authors in [21] have presented an illuminating survey about these logics.

Extensive research efforts have been seen to tackle the problem of modeling the qualitative behaviors of systems. Examples of these efforts are Timed Propositional Temporal Logic(TPTL) [22], Explicit Clock Temporal Logic [23], [24], Metric Temporal logic (MTL) [25], Metric Interval Temporal Logic (MITL)[26], Computation Tree Logic(ICTL) [27] and RCTL [28], Real-Time Computation Tree Logic (RTCTL) [29], and Timed Computation Tree Logic (TCTL) [30]. These logics are considered as extensions of temporal logics. In contrast to this paper, the previous logics are not able to express epistemic knowledge.

On the other hand, finite state automata provide the most elegant model for memory structures of reasoning agents [10]. Timed automata are also proposed as extensions of finite state automata to model time constrains [16], [31], [32]. In order to overcome the inadequacy of epistemic temporal logics in expressing real time behavior of agents, many real time epistemic logic approaches have been introduced [33], [34], [5], [4]. These logics use timed automata to express and model the agents while the classical interpreted systems of epistemic logic are extended by adding the operational semantics of timed automata. Timed automaton is considered as a simplified version of hybrid automaton which integrates differential equations and finite automata in a single formalism [15]. To specify the behaviors of MAS situated in a dynamic environment, many hybrid automata approaches have been presented in the literature [35], [36], [37], [38], [39]. Our work in this paper augments the hybrid automaton with knowledge to formally specify and verify the epistemic behaviors of agents within continuous dynamics. A new logic that extends epistemic logic with quantitative real time requirement is employed for this purpose.

III. Background

The concept of hybrid automata along with their constrains are introduced here in details. An example from [40] is used for illustration. Several definitions are presented to describe the constrains that appear within the hybrid automata.

Definition 1 (Linear Constraints and Evaluation): Let $\mathbb{R}$ denotes a set of real numbers, $\mathbb{R}^{\geq 0}$ is a set of non-negative real numbers, χ is a set of variables with an element $x \in \chi$, $\omega = \sum_{i=1}^{|\chi|} a_i \cdot x_i$, with $x_i \in \chi$, is a combination of variables in the set χ, where $a_i \in \mathbb{R}$, and b $\in \mathbb{R}$, $\sim_c \in \{>, \geq =, <, \leq\}$. The grammar of all linear constraints $\Phi(\chi)$, with a typical element $\varphi \in \Phi(\chi)$, is defined as follows:

$$\varphi ::= \ \omega \sim_c b \mid \varphi \wedge \varphi \mid true$$

Let $v \in \mathbb{R}^{|\chi|}$ be the values of the variables in χ and v_i be the value of the ith component of v. we call $v \models \varphi$ if v fulfills the constraint φ, and is defined using the grammar:

$\varphi = true.$		
$\varphi = \sum_{i=1}^{\|\chi\|} a_i \cdot x_i \sim_c b$	iff	$\sum_{i=1}^{\|\chi\|} a_i \cdot v_i \sim_c b \ \ holds.$
$\varphi_1 \wedge \varphi_2$	iff	$v \models \varphi_1 \ and \ v \models \varphi_2.$

Definition 2 (Dynamical Constraints and Evaluation): Let $\dot{\chi}$ be a set of the first derivatives of the variables within χ with typical element $\dot{x} \in \dot{\chi}$ and $b \neq 0, c \in \mathbb{R}$, $\sim_d \in \{=, \leq, \geq\}$. Let $\mathbb{D}(\chi \cup \dot{\chi})$ be the set of constraints over the variables in $\chi \cup \dot{\chi}$ with typical element $d \in \mathbb{D}$. The set of all possible dynamical constraints is defined as follows:

$$d ::= \ \dot{x} \sim_d c \mid \dot{x} + b \cdot x = c \mid d \wedge d \mid true$$

Let $f : \mathbb{R}^{\geq 0} \rightarrow \mathbb{R}^{|\chi|}$ be a differentiable function and $f'(t)$ be the differentiation of f with respect to time $t \in \mathbb{R}^{\geq 0}$. We call $f \vDash_* d$, if the function f fulfills d defined by the grammar:

$d = true.$		
$d = \dot{x} \sim_d c$	iff	$f'(t) \sim_d c \ \ holds.$
$d = \dot{x} + b \cdot x \sim_d c$	iff	$f'(t) + b \cdot f(t) \sim_d c \ \ holds.$
$d = d_1 \wedge d_2$	iff	$f \vDash_* d_1 \ and \ f \vDash_* d_2.$

Definition 3 (Hybrid Automaton): A hybrid automaton is defined as the tuple HA $= (Q, \chi, Ins, Flow, \eta, E, q^0, v^0)$ where:

- Q is a finite set of locations.
- χ denotes a set of real variables.

- $Ins : Q \rightarrow \Phi(\chi)$ denotes the function that allocates the constraint $Ins(q)$ for every $q \in Q$.
- $Flow : Q \rightarrow \mathbb{D}(\chi \cup \dot{\chi})$ is the function that allocates the constraints $Flow(q)$ for every $q \in Q$.
- η is a finite set of events.
- $E \subseteq Q \times \eta \times \Phi(\chi) \times 2^{\chi} \times Q$ is a transition relation among control locations.
- $q^0 \in Q$ denotes the initial location of the hybrid automaton.
- $v^0 \in \mathbb{R}^{|\chi|}$ denotes the initial values of the variables in χ.

Every $e \in E$ is denoted as $q_1 \xrightarrow{a,\varphi,X} q_2$, where q_1 and q_2 is the start and the end locations respectively, $a \in \eta$ is an event, φ denotes the enabling condition of e, and $X \subseteq \chi$ denotes the set of real variables to be reset.

Definition 4 (State): At any time $t \in \mathbb{R}^{\geq 0}$, a state $\sigma \in Q \times \mathbb{R}^{|\chi|} \times \mathbb{R}^{\geq 0}$ of HA is conventionally denoted as the tuple $\sigma = \langle q, v, t \rangle$, where $q \in Q$ and v is the value of the real variables at time t. A state $\sigma = \langle q, v, t \rangle$ satisfies a constraint $\varphi \in \Phi(\chi)$ at point t, conventionally written as $\sigma \models^t \varphi$, iff $v \models \varphi$. A state $\sigma = \langle q, v, t \rangle$ is called admissible iff $\sigma \models^t Ins(q)$. Two states $\sigma_1 = \langle q_1, v_1, t_1 \rangle$ and $\sigma_2 = \langle q_2, v_2, t_2 \rangle$ are equivalent denoted as $\sigma_1 \equiv \sigma_2$ iff $q_1 = q_2 = q$ and $\sigma_i \models^{t_i} Ins(q), i \in \{1, 2\}$.

A labeled transition system between states is usually used to describe the semantics of HA. The transition between any two admissible states $\sigma_1 = \langle q_1, v_1, t_1 \rangle$ and $\sigma_2 = \langle q_2, v_2, t_2 \rangle$ is defined either by a discrete or a delay transition as follows:

> **Discrete transition** for $a \in \eta$, $\sigma_1 \xrightarrow{a} \sigma_2$ iff $t_1 = t_2$ and there exists $q_1 \xrightarrow{a,\varphi_1,X} q_2 \in E$ such that $v_1 \models \varphi_1$, and $v_2 \models Ins(q_2)$.
>
> **Delay transition** for $\delta \in \mathbb{R}^{\geq 0}$, $\sigma_1 \xrightarrow{\delta} \sigma_2$ iff $q_1 = q_2$, $\delta = (t_2 - t_1)$ is the time duration passed in location q_1, there is a differentiable function f having $f \vDash_* Flow(q_1)$ and $f(t_1) = v_1$ and $f(t_2) = v_2$, and for all $t \in [t_1, t_2]$, $f(t) \models Ins(q1)$.

Now, the states and the transitional rules between states are totally defined and thus we are ready to define the dense state space as follows:

Definition 5 (dense state space): The dense state space of HA can be defined as the tuple $(\theta, \sigma^0, \longrightarrow)$, where $\theta = Q \times \mathbb{R}^{|\chi|} \times \mathbb{R}^{\geq 0}$ is the set of all states, $\sigma^0 = \langle q^0, v^0, 0 \rangle$ is the initial state such that v^0 is the value of the variables in χ in the control location q^0 at $t = 0$ with $v^0 \models Ins(q^0)$, and $\longrightarrow \subseteq \theta \times (\eta \cup \mathbb{R}) \times \theta$.

When a hybrid automaton is run, a sequence of state transitions is generated. In the following, we define the path and the run.

Definition 6 (Path and Run): A path $\rho = \sigma_1 \sigma_2 \sigma_3, \ldots,$ of $\mathbb{HA}$ denotes a finite or infinite sequence of admissible states, where the transition between any two consecutive states is associated either by a discrete or delay transition. Let $\Pi(\mathbb{HA})$ denotes the set of all paths of $\mathbb{HA}$. A run of $\mathbb{HA}$ denotes a path ρ beginning with the initial state σ_0.

Each path $\rho \in \Pi(\mathbb{HA})$ generates infinite number of reachable states due to the delay transitional rules. An appropriate method to represent those infinite state is to use a symbolic representation using mathematical intervals. Let us call the mathematical interval a *region*, and it is defined as follows:

We write a run ρ as $\rho = \Gamma_0, a_1, \Gamma_1, a_2, ...,$ a sequence of regions, where each $\Gamma_i \subseteq Q \times \mathbb{R}^{|\chi|} \times \mathbb{R}^{\geq 0}$ is the maximal sub-sequence of admissible states such that for all consecutive states $\sigma_j, \sigma_{j+1} \in \Gamma$, it holds that $\sigma_j \xrightarrow{\delta} \sigma_{j+1}$. Additionally, a transition between two consecutive regions Γ_i and Γ_{i+1}, conventionally written as $\Gamma_i \xrightarrow{a} \Gamma_{i+1}$, is enabled, if there exist two states $\sigma_i \in \Gamma_i$, $\sigma_{i+1} \in \Gamma_{i+1}$ such that $\sigma_i \xrightarrow{a} \sigma_{i+1}$. Conventionally, Γ is writen as $\Gamma = \langle q, V, T \rangle$, such that T denotes the total duration time of all states in Γ and V denotes the tuple of intervals values of the variables throughout the time interval T. Let Γ^0 denotes the initial region obtained from the intial state σ^0 using delay transitions.

Definition 7 (Reachability): A certain region Γ_i is reachable in a run $\rho \in \Pi(\mathbb{HA})$, if $\Gamma_i \in \rho$. A state σ_j is reachable, if there is a reachable region Γ_i with $\sigma_j \in \Gamma_i$.

Now, the dense state space can be generalized by a region state space as follows:

Definition 8 (region state space): A region state space of HA can be defined as the tuple $(\Delta, \Gamma^0, \longrightarrow)$, where Δ is the set of all possible regions, $\Gamma^0 = \langle q, V, T \rangle$ is the initial region formed by a delay transitions from the initial state $\sigma^0 \in \Gamma^0$, and $\longrightarrow \subseteq \Delta \times \eta \times \Delta$ is the transition relation defined as $\Gamma^1 \xrightarrow{a} \Gamma^2$ iff there exist $\sigma^1 \in \Gamma^1$ and $\sigma^2 \in \Gamma^2$ such that $\sigma^1 \xrightarrow{a} \sigma^2$.

A. Automata Composition

A MAS is generally modeled by various parallel hybrid automata representing the agents. Communication among the agents is achieved using synchronized events. The overall behavior of the entire MAS can be described using the parallel composition. A two hybrid automata can be composed as follows:

Let $\text{HA}_\text{i} = (Q_i, \chi_i, Ins_i, Flow_i, \eta_i, E_i, q_i^0, v_i^0)$ $1 \leq i \leq 2$ be two hybrid automata, with $Q_1 \cap Q_2 = \varnothing$,

Definition 9 (Parallel Composition): The parallel composition of HA_1 and HA_2 is a hybrid automaton $\text{HA} = (Q, \chi, Ins, Flow, \eta, E, q^0, v^0)$, where $Q = Q_1 \times Q_2$, $\chi = \chi_1 \cup \chi_2$, $Ins = Ins_1 \wedge Ins_2$, $Flow = Flow_1 \wedge Flow_2$, $\eta = \eta_1 \cup \eta_2$, $q^0 = (q_1^0, q_2^0)$ $v^0 = (v_1^0, v_2^0)$, and a transition in E is defined as follows: $q_1 \xrightarrow{a,\varphi_1,X_1} q_1' \in E_1$ and $q_2 \xrightarrow{a,\varphi_2,X_2} q_2' \in E_2$

- $a \in \eta_1 \cap \eta_2$ is a joint event, then $(q_1, q_2) \xrightarrow{a,\varphi_1 \wedge \varphi_2, X_1 \cup X_2} (q_1', q_2') \in E$.
- $a \in \eta_1 \setminus \eta_2$, then then $(q_1, q_2) \xrightarrow{a,\varphi_1,X_1} (q_1', q_2) \in E$.
- $a \in \eta_2 \setminus \eta_1$, then then $(q_1, q_2) \xrightarrow{a,\varphi_2,X_2} (q_1, q_2') \in E$.

A run of any two composed automata, denoted as $\sum_{H_1 \circ H_2}$, is the sequence $\Lambda_0, a_1, \Lambda_1, a_2, ...$ of compound regions, where a transition between two regions relates according to the definition of the transitional relation defined previously. Each global regions takes the form $\Lambda = \langle \Gamma_1, \Gamma_2 \rangle$, where $\Gamma_i = (q_i, V_i, T)$.

Again, the regions state space $(\Delta, \gamma^0, \longrightarrow)$ is similar to its previous definition, except that each element $\Lambda \in \Delta$ is a global region, and γ^0 is the initial global region for each automaton.

Let $loc_i : \Delta \rightarrow Q_i$ be a function that takes a global region and returns the current location of the agent i, and $Loc : \Delta \rightarrow Q$ a function that returns the locations of the m agents. Let $duration(\Gamma) :\subseteq \mathbb{R}^{\geq 0}$ be a relation that returns the time interval of a region Γ; i.e for $\Gamma_i = (q, V, T)$, $duration(\Gamma) = T$.

IV. Hybrid Interpreted System

Interpreted Systems [2] are usually used as the formal semantics that describe the temporal epistemic language. Therefore, the interpreted systems is extended to be adapted on hybrid automata as well.

Let AG denotes a set of m agents such that each agent is represented as a hybrid automaton $\text{HA}_\text{i} = (Q_i, \chi_i, Ins_i, Flow_i, \eta_i, E_i, q_i^0, v_i^0)$, $1 \leq i \leq m$, and their parallel composition in $\text{HA} = (Q, \chi, Ins, Flow, \eta, E, q^0, v^0)$. Let $Prop_i$ be a set of Propositional variables for each agent $1 \leq i \leq m$, and $Prop = \bigcup Prop_i$. Let $Val_i : Q_i \rightarrow 2^{Prop_i}$ be the valuation functions for the ith agent, which assigns the truth value of $Prop_i$ to the locations. Let $Val : Q \rightarrow 2^{Prop}$ is the valuation function for the m agents, such that $Val(q) = \bigcup Val_i(q_i)$. Then, the hybrid interpreted system is defined as follows:

Definition 10 (Hybrid Interpreted System): A hybrid interpreted system is the tuple $\mathbb{M} = (\Delta, \Gamma^0, \longrightarrow, \simeq_1, \simeq_2, \ldots, \simeq_m, \nu)$, where

- $\Delta, \Gamma^0, \longrightarrow$ are defined as the definition in region state space.
- $\simeq_i \subseteq \Delta \times \Delta$ is the epistemic indistinguishably (accessibility) relation for agent i defined by $\Gamma^1 \simeq \Gamma^2$ iff $loc(\Gamma^1) = loc(\Gamma^2)$ and for each state $\sigma_j \in \Gamma^1$ there exists a state $\sigma_k \in \Gamma^2$, such that $\sigma_j \equiv \sigma_k$. $\simeq_i$ is an equivalence relation.
- $\nu : \Delta \rightarrow 2^{Prop}$ is the valuation function that is defined by extending the definition of Val such that $\nu(\Gamma) = Val(loc(\Gamma)$.

The epistemic relation $\simeq$ defined previously is standard in epistemic logic under interpreted systems. More details and examples about this relation can be found in [41]. The knowledge of a group of agents can be defined as:

Definition 11 (Group epistemic relation): Let AG be a set of m agents, and $\kappa \subseteq$ AG, we define a group epistemic relations on a group of agents κ as follow:

- Everybody knows: $\simeq_\kappa^E = \bigcup_{i \in \kappa} \simeq_i$. We have $\Gamma \simeq_\kappa^E \Gamma^{'}$ iff for all $i \in \kappa$, then $\Gamma \simeq_i \Gamma^{'}$.
- Distributed knowledge: $\simeq_\kappa^D = \bigcap_{i \in \kappa} \simeq_i$. We have $\Gamma \simeq_\kappa^D \Gamma^{'}$ iff there exists $i \in \kappa$, then $\Gamma \simeq_i \Gamma^{'}$.
- Common knowledge: $\simeq_\kappa^C = (\simeq_\kappa^E)^+$, where $^+$ denotes the reflexive transitive closure of the underlying relation.

V. The Proposed Logic (ERCTL)

The syntax and semantics of the proposed ERCTL are formally described in this section. As previously mentioned, the proposed ERCTL extends the logic RCTL [28] by adding knowledge operators. We first begin by describing timed variables that might appear in a formula to quantify its timing.

Definition 12 (Clocks): Let $\mathbb{T} \subseteq \chi$ denotes a set of non-negative real variables called *clocks*, and $\Phi(\mathbb{T})$ denotes a set of constraints over $\mathbb{T}$. Let $\xi : \mathbb{T} \rightarrow \mathbb{R}^{\geq 0}$ denotes the valuation ξ of the clocks $\mathbb{T}$. For $\pi \in \Phi(\mathbb{T})$, we call $\xi \models \pi$, if ξ satisfies π.

A. Syntax of ERCTL

Let L denotes a set of propositions representing the locations, η denotes a set of propositions representing the events, χ denotes a set of real variables, $\mathbb{T} \subseteq \chi$ denotes a set of clocks, $\Phi(\chi)$ and $\Phi(\mathbb{T})$ denote the set of all constraints on the variables in $\chi, \mathbb{T}$ respectively. Let AG be a set of m agents, with $\kappa \subseteq$ AG. Let $y \in \mathbb{T}$, $l \in L$, $a \in \eta$, $\phi \in \Phi(\chi)$, $\pi \in \Phi(\mathbb{T})$, $i \in$ AG, and $\kappa \subseteq$ AG.

Definition 13 (ERCTL Formulas): The set of ERCTL formulas is defined inductively as follows:

$$\Psi ::= p \mid a \mid \phi \mid y.\Psi \mid \pi \mid \neg\Psi \mid \Psi_1 \wedge \Psi_2 \mid \exists(\Psi_1 U \Psi_2) \mid \forall(\Psi_1 U \Psi_2) \mid \mathbb{K}_i\Psi \mid \mathbb{E}_\kappa\Psi \mid \mathbb{D}_\kappa\Psi \mid \mathbb{C}_\kappa\Psi$$

In addition to the standard Boolean connectives, the previous syntax includes the path quantifiers $\forall$, denoted in all possible paths, and $\exists$, denotes that there exists a path (more details about path quantifiers can be found in [42]). Furthermore, the syntax of ERCTL defines two fragments: RCTL and an epistemic one. The RCTL fragment includes formulas of the form $y.\Psi$ representing *"the formula Ψ is true at certain time represented by the clock y"*. The epistemic fragment of ERCTL includes formula of the form $\mathbb{K}_i\Psi$ to represent "$agent$ i $knows$ $that$ Ψ", $\mathbb{E}_\kappa\Psi$ to represent "$everyone\ in\ group\ \kappa\ knows\ that\ \Psi$", $\mathbb{D}_\kappa\Psi$ to represent *"it is distributed knowledge in group κ that Ψ is $true$"*, and $\mathbb{C}_\kappa\Psi$ standing for *"it is common knowledge in group κ that Ψ"*.

The other common formulas are defined as follows:

- $\exists\Diamond\Psi$ is equivalent to the formula $\exists(true\ U\Psi)$.
- $\forall\Diamond\Psi$ is equivalent to the formula $\forall(true\ U\Psi)$.
- $\bar{\mathbb{K}}_i\Psi$ is equivalent to the formula $\neg\mathbb{K}_i\neg\Psi$.
- $\bar{\mathbb{E}}_\kappa\Psi$ is equivalent to the formula $\neg\mathbb{E}_\kappa\neg\Psi$.
- $\bar{\mathbb{D}}_\kappa\Psi$ is equivalent to the formula $\neg\mathbb{D}_\kappa\neg\Psi$.
- $\bar{\mathbb{C}}_\kappa\Psi$ is equivalent to the formula $\neg\mathbb{C}_\kappa\neg\Psi$.

B. Semantics of ERCTL

Let AG denotes a set of m agents such that each agent is represented by a hybrid automaton $\text{HA}_\text{i} = (Q_i, \chi_i, Ins_i, Flow_i, \eta_i, E_i, q_i^0, v_i^0)$, $1 \leq i \leq m$, and their parallel composition in $\text{HA} = (Q, \chi, Ins, Flow, \eta, E, q^0, v^0)$. Let $\mathbb{M} = (\Delta, \Gamma^0, \longrightarrow, \simeq_1, \simeq_2, \ldots, \simeq_m, \nu)$ be a hybrid interpreted system. Let $\Pi(\mathbb{HA})$ denotes the set of all regions

produced from the runs of hybrid automaton with a typical region $\Gamma = (q, V, T) \in \Pi(\mathbb{HA})$

Definition 14 (Satisfaction Relation ERCTL): Let Ψ is a ERCTL formula, The satisfaction relation $\langle \mathbb{M}, \Gamma \rangle \vDash \Psi$ denotes that Ψ is true at a region Γ in the model $\mathbb{M}$ and is defined as follows:

- $\langle \mathbb{M}, \Gamma \rangle \vDash p$	iff	$p \in \nu(\Gamma)$.
- $\langle \mathbb{M}, \Gamma \rangle \vDash a$	iff	there $is\ \Gamma' \in \Pi(\mathbb{HA})\ with\ \ \Gamma \xrightarrow{a} \Gamma'$.
- $\langle \mathbb{M}, \Gamma \rangle \vDash \phi$	iff	there $is\ \sigma_k = (q_k, v_k, t_k) \in \Gamma, \sigma_k \models^t \phi$.
- $\langle \mathbb{M}, \Gamma \rangle \vDash y.\Psi$	iff	there $is\ \sigma = \langle q, v, t \rangle \in \Gamma$ such that $Evl(y) = t\ and\ \sigma \models^t \phi$.
- $\langle \mathbb{M}, \Gamma \rangle \vDash \pi$	iff	there $is\ \xi \in duration(\Gamma)$ such that $\xi \vDash \pi$.
- $\langle \mathbb{M}, \Gamma \rangle \vDash \neg\Psi$	iff	$\langle \mathbb{M}, \Gamma \rangle \nvDash \Psi$.
- $\langle \mathbb{M}, \Gamma \rangle \vDash \Psi_1 \wedge \Psi_2$	iff	$\langle \mathbb{M}, \Gamma \rangle \vDash \Psi_1$ and $(\mathbb{M}, \Gamma) \vDash \Psi_2$.
- $\langle \mathbb{M}, \Gamma \rangle \vDash \exists(\Psi_1 U \Psi_2)$	iff	there is a run $\Pi \in \Pi(\mathbb{HA}), \Pi = \Gamma_0, \Gamma_1, \cdots,$ with Γ=Γ_0, for some $j \geq 0$, $\langle \mathbb{M}, \Gamma_j \rangle \vDash \Psi_2$, and $\langle \mathbb{M}, \Gamma_k \rangle \vDash \Psi_1$ for $0 \leq k < j$.
- $\langle \mathbb{M}, \Gamma \rangle \vDash \forall(\Psi_1 U \Psi_2)$	iff	for every run $\Pi \in \Pi(\mathbb{HA}), \Pi = \Gamma_0, \Gamma_1, \cdots,$ with Γ=Γ_0,for some $j \geq 0$, $\langle \mathbb{M}, \Gamma_j \rangle \vDash \Psi_2$, and $\langle \mathbb{M}, \Gamma_k \rangle \vDash \Psi_1$ for $0 \leq k < j$.
- $\langle \mathbb{M}, \Gamma \rangle \vDash \mathbb{K}_i \Psi$	iff	for all $\Gamma' \in \Delta$ with $\Gamma' \simeq_i \Gamma$ then $\langle \mathbb{M}, \Gamma' \rangle \vDash \Psi$.
- $\langle \mathbb{M}, \Gamma \rangle \vDash \mathbb{E}_\kappa \Psi$	iff	for all $\Gamma' \in \Delta$ with $\Gamma' \simeq_\kappa^E \Gamma$ then $\langle \mathbb{M}, \Gamma' \rangle \vDash \Psi$.
- $\langle \mathbb{M}, \Gamma \rangle \vDash \mathbb{D}_\kappa \Psi$	iff	for all $\Gamma' \in \Delta$ with $\Gamma' \simeq_\kappa^D \Gamma$ then $\langle \mathbb{M}, \Gamma' \rangle \vDash \Psi$.
- $\langle \mathbb{M}, \Gamma \rangle \vDash \mathbb{C}_\kappa \Psi$	iff	for all $\Gamma' \in \Delta$ with $\Gamma' \simeq_\kappa^C \Gamma$ then $\langle \mathbb{M}, \Gamma' \rangle \vDash \Psi$.

Intuitively, the formula $\mathbb{K}_i \Psi$ holds in a region Γ within the hybrid interpreted system $\mathbb{M}$ if Ψ holds in all regions that are indistinguishable for the agent i from Γ. The formula $\mathbb{E}_\kappa \Psi$ holds in a region Γ within the hybrid interpreted system $\mathbb{M}$ if Ψ is true in all regions that a group κ of agents is unable to distinguish from the Γ.The formula $\mathbb{D}_\kappa \Psi$ holds in a region Γ within the hybrid interpreted system $\mathbb{M}$ if the combined knowledge of all agents in κ implies Ψ. The formula $\mathbb{C}_\kappa \Psi$ holds in a region Γ within the hybrid interpreted system $\mathbb{M}$ if everyone knows that Ψ holds at Γ, and everyone knows that everyone knows that Ψ holds at Γ, etc.

Definition 15 (Satisfiability of Formulas): Let AG denotes a set of m agents such that each agent is represented as a hybrid automaton $\mathrm{HA_i} = (Q_i, \chi_i, Ins_i, Flow_i, \eta_i, E_i, q_i^0, v_i^0)$, $1 \leq i \leq m$, and their parallel composition in $\mathrm{HA} = (Q, \chi, Ins, Flow, \eta, E, q^0, v^0)$. Let $\mathbb{M} = (\Delta, \Gamma^0, \longrightarrow, \simeq_1, \simeq_2, \ldots, \simeq_m, \nu)$ be a hybrid interpreted system, A ERCTL formula Ψ is satisfiable in $\mathbb{M}$ iff $\langle \mathbb{M}, \Gamma_0 \rangle \vDash \phi$, where $\Gamma_0 \in \Pi(\mathbb{HA})$ is the initial region.

VI. Specification of Requirements

As the proposed ERCTL combines the expressive power of RCTL and epistemic logic, we will focus on the expressive power of ERCTL to specify those properties that combine both logics together. To exemplify the expressive power of the proposed ERCTL, we specify properties on a slightly modified version of railroad crossing system found in [43]. More details about this illustrative example can be found in [40].

A. Example

The example shown in figure 1 consists of three agents, namely the *Train*, the *Gate*, and the *Controller*. The main goal is to track the trains crossing an intersection. The gate guards the intersection and it closes or opens based on a train status which is approaching or leaving the intersection. The gate is completely monitored by the controller. The controller receives signals from the train and accordingly sends lower or raise commands to the gate. Let the train is initially 1000 meters away from the gate and moves at a speed of 50 m/s. There is a sensor positioned at a distance of 500 meters on the track. The sensor detects that the train is approaching and thus sends an *app* signal to the controller. After sending the *app* signal, the train slows down according to the differential equation $\dot{x} = -\frac{x}{25} - 30$. After a duration of 5 seconds, the controller sends a *lower* command to the gate, which in turn starts to lower down at a rate of -20 degrees per second. After the train crosses the gate, it accelerates following the differential equation $\dot{x} = \frac{x}{5} + 30$. Another sensor is positioned at a distance of 100 meters after the crossing to detect the train when it is leaving. This sensor sends an *exit* command to the controller. After 5 seconds, the controller starts to raise the gate to its normal position.

By using ERCTL, we can specify a property Ψ that cannot be expressed by the standard RCTL or epistemic logic. To clarify more, we consider the following example:

$$\Psi = \exists\Diamond\mathbb{K}_{Train}(t_1.app \wedge \exists\Diamond(\neg t_2.to\_close \wedge t_2 \leq t_1 + 10)) \quad (1)$$

Formula 1 specifies that there exists a behavior in the system such that the *Train* knows a situation in which it sends *app* and then the *Gate* eventually will not be closed within 10 sec.

$$\Psi = \mathbb{K}_{controller}(t_1.lower \rightarrow \forall\Diamond(to\_close \wedge t_2 \leq t_1 + 5)) \quad (2)$$

Formula 2 specifies that the *Controller* agent knows that when it sends a *lower* command, the agent *Gate* will send *to_close* command within 5 sec. and thus the agent *Gate* eventually will not be closed within 10 sec.

$$\Psi = \forall\Box\mathbb{K}_{Train}(t_1.app \rightarrow \forall\Box(t_2.(x > 100) \wedge t_2 \leq t_1 + 5)) \quad (3)$$

Formula 3 specifies that the agent *Train* always knows that whenever it approaches the gate, its distance to the gate is always greater than 100 meters for 20 time units.

In order to formally verify a certain property using model checking within the proposed ERCTL, we should focus on fragment of ERCTL that can be checked with reachability. Several requirements of interest can be specified as kind of reachability. Generally, a formula Ψ is reachable, if it is possible to reach a state holding Ψ. Thus the reachability of the property Ψ aims to find if it is possible to find a region within the run of agents in which the formula Ψ is satisfiable?. This can be achieved using ERCTL as follows:

$$init \rightarrow \exists\Diamond\Psi \quad (4)$$

init in formula 4 indicates the conjunctions of the initial states of the system under investigation. The reachability of a certain formula is usually computed starting with the initial region of a region-space exploration of a model and extending the reachability on transitions until reaching fixed regions. In

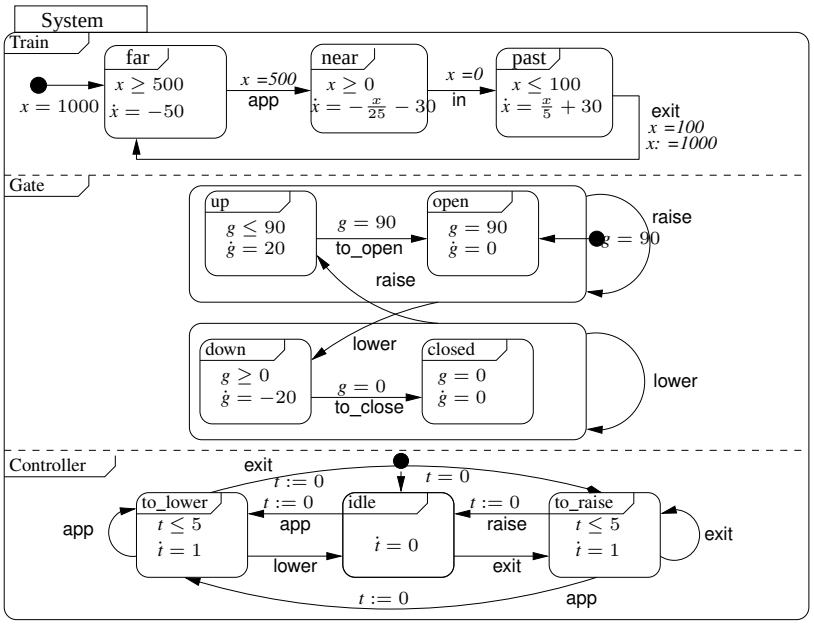

Fig. 1. Modeling of the train gate controller Example as hybrid automata [40].

[44], a semi-decision algorithm for computing the reachability of regions of a hybrid automaton is introduced by one of the authors. This algorithm is shown in Fig. 2. In Fig. 2, if the initial region is Γ_0, *produce(R)* denotes the set of reached regions attached to the region *R* with a discrete step.

```
W :=produce(Γ0)
Reachable := Γ0
while W ≠ ∅ do
take R from W
if R ∉Reachable then Reachable :=
Reachable ∪ R
end if
W := W ∪ (produce(R)\ W)
end while
```

Fig. 2. Computation of the reachability analysis [44] .

Checking the reachability for a property within the underline transition system of hybrid automata is generally undecidable except for certain classes of hybrid automata[45]. Consequently, the decidability problem is inherited in ERCTL.

VII. Reachability as CLP

In this section the implementation of the proposed framework using constraint logic program (CLP) [46] is shown. CLP has been chosen to implement the proposed framework for many reasons. First, hybrid interpreted system can be described as a constrained system. These constraints represent the continuous dynamics e.g., the invariants, the flows, and transitions. Second, constraints can be used to represent specific parts of the state space easily. Third, there are operational semantics similarities between *CLP* and the hybrid Interpreted System. Moreover, constraints allow us to concisely represent regions symbolically as mathematical intervals where an appropriate constraint solver used to reason about the reachability of a particular state inside this interval. Moreover, the Logic programming parts allows us to implement the knowledge efficiently.

The implemented prototype is built using ECLiPSe Prolog [47]. The definitions of both the formal syntax and semantics of hybrid automata and the enrichment of knowledge are followed. An overview of the implementation is given here. Let's start with modeling the locations and their constraints using the predicate *epistemicAutomaton* as shown in fig.3. *epistemicAutomaton* denotes the epistemic automaton and *Location* denotes the current location of the automaton. $Vars$ represents the variables participating in epistemic automaton and $Vars0$ represents their corresponding initial values. $Ins(Vars)$ is the list of invariant constraints on the variables in $Vars$ within the control location. Whereas, $Flow(vars)$ represents the list of constraints flows on the variables $Vars$ with respect to initial time $T0$ at the start of the continuous flow and $Time$. *initKnow* represents knowledge at the location. The knowledge remains unchanged during the continuous evolution. finally, *Event* represents the fired event during the run.

The transition systems are then encoded into the predicate *evolve* as shown in fig.4, that describes the two kind of transitions. The automaton evolves with either continuous or discrete transitions depending on the occurring constraints during the run. It is important to note that , within the discrete step, the knowledge is updated from a state to another by appending the knowledge of the first state *Know1* with the shared knowledge *Shared*, coming from the other automata, to produce the knowledge *know2*. Once the epistemic automata have been modeled, an overall state machine is constructed with the aim to execute the model. To achieve this goal, a reachability predicate is implemented as shown in Fig. 5.

The *reachability* is a state machine employed to generate the behaviors of the concurrent epistemic hybrid automata. It starts with the definition of each participating epistemic hybrid automaton with its initial variables, timing, and knowledge. As soon as the *reachability* has been defined, the entire model is

```
epistemicAutomaton(Location,Vars,Var0,T0,Time,initKnow,Event):- Flow(Vars), Ins(Vars),Time $>=T0,
initKnow=[automaton knowledgeshared knowledge],Event &::event.
```

Fig. 3. Epistemic automaton definition

```
evolve(epistemicAutomaton(State,Startstate,Know),(State,Newstartstate,Know),Shared,T0,T,Event) :-
continuous(epistemicAutomaton(State,Startstate,Know),(State,Newstartstate,Know),Shared,T0,T,Event);
discrete(epistemicAutomaton(State,Startstate,Know1),(State,Newstartstate,Know2),Shared,T0,T,Event);
```

Fig. 4. Transition system implementation

```
reachability((L1,Var01,know1),(L2,Var02,know2),...,(Ln,Var0n,known),T0,[Reg|NxtReg],PastReg)
:- epistemicAutomaton(L1,Var01,Know1),(State1,Newstartstate1,Know11), Shared,T0,T,Event),
epistemicAutomaton(L2,Var02,Know2),(State2,Newstartstate2,Know12), Shared,T0,T,Event), ....,
epistemicAutomaton(Ln,Var0n,Known),(Staten,Newstartstatne,Know1n), Shared,T0,T,Event),
evolve(epistemicAutomaton(L1,Startstate,Know1),(State1,Newstartstate1,Know), Shared,T0,T,Event),
evolve(epistemicAutomaton(L2,Startstate,Know2),(State2,Newstartstate2,Know), Shared,T0,T,Event), .....,
evolve(epistemicAutomaton(Ln,Startstate,Known),(Staten,Newstartstaten,Know), Shared,T0,T,Event),
reachability((State1,Newstartstate1,Know11),(State2,Newstartstate2,Know12),..,(Staten,Newstartstaten,Know1n),
Newstarttime,PastReg)).
```

Fig. 5. A reachability to the execution of epistemic hybrid automata.

invoked for the purpose of running and verification. By using the CLP model, we are able to verify the properties described in VI.

VIII. Conclusion

In this paper, we have introduced a new logic called ERCTL that extends the logic RCTL with epistemic modalities. This extension allows us to formally specify several qualitative epistemic requirements of MAS evolving in continuous dynamical environment. The fundamental underline Interpretation model of the logic was hybrid automata. The later, were extended to produce the so-called interpreted hybrid system that forms the basic Interpretation model for both the epistemic part and the real time continuous dynamic part. The paper showed how to specify several interesting requirements using ERCTL. To put the formal verification into consideration, we showed how to implement the proposed work using constraints logic programming CLP. As converting a model to CLP is a tedious work, it's worth developing to incorporate the ERCTL in the model checking tool [48].

References

[1] B. Lopes, M. Benevides, and E. H. Haeusler, *Reasoning about Multi-Agent Systems Using Stochastic Petri Nets*. Springer International Publishing, 2015, pp. 75–86.

[2] R. Fagin, Y. Moses, M. Y. Vardi, and J. Y. Halpern, *Reasoning about knowledge*. MIT press, 2003.

[3] W. van Der Hoek and M. Wooldridge, "Cooperation, knowledge, and time: Alternating-time temporal epistemic logic and its applications," *Studia Logica*, vol. 75, no. 1, pp. 125–157, 2003.

[4] B. Woźna and A. Lomuscio, "A logic for knowledge, correctness, and real time," in *Computational Logic in Multi-Agent Systems*. Springer, 2004, pp. 1–15.

[5] Y. Moses and B. Bloom, "Knowledge, timed precedence and clocks (preliminary report)," in *Proceedings of the thirteenth annual ACM symposium on Principles of distributed computing*. ACM, 1994, pp. 294–303.

[6] W. v. d. H. R. Verbrugge, "Epistemic logic: A survey," *Game theory and applications*, vol. 8, p. 53, 2002.

[7] B. F. Chellas, *Modal logic: an introduction*. Cambridge Univ Press, 1980, vol. 316.

[8] J.-J. C. Meyer and W. Van Der Hoek, *Epistemic logic for AI and computer science*. Cambridge University Press, 2004, vol. 41.

[9] A. Pnueli, "The temporal logic of programs," in *Foundations of Computer Science, 1977., 18th Annual Symposium on*, 1977, pp. 46–57.

[10] S. Mohalik and R. Ramanujam, "Automata for epistemic temporal logic with synchronous communication," *Journal of Logic, Language and Information*, vol. 19, no. 4, pp. 451–484, 2010.

[11] R. Alur, T. A. Henzinger, and O. Kupferman, "Alternating-time temporal logic," *Journal of the ACM (JACM)*, vol. 49, no. 5, pp. 672–713, 2002.

[12] F. Bacchus and F. Kabanza, "Using temporal logics to express search control knowledge for planning," *Artificial Intelligence*, vol. 116, no. 1-2, pp. 123 – 191, 2000. [Online]. Available: http://www.sciencedirect.com/science/article/B6TYF-40W6PMM-5/2/836df2c3643e1a101f2c8a726fef310d

[13] R. Alur, C. Courcoubetis, T. A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine, "The algorithmic analysis of hybrid systems," in *ICAOS: International Conference on Analysis and Optimization of Systems – Discrete-Event Systems*, ser. Lecture Notes in Control and Information Sciences 1994. Springer, Berlin, Heidelberg, New York, 1994, pp. 331–351.

[14] Platzer and André, "Logical analysis of hybrid systems: A complete answer to a complexity challenge," in *Proceedings of the 14th International Conference on Descriptional Complexity of Formal Systems*, ser. DCFS'12. Springer-Verlag, 2012, pp. 43–49.

[15] T. Henzinger, "The theory of hybrid automata," in *Proceedings of the 11th Annual Symposium on Logic in Computer Science*. New Brunswick, NJ: IEEE Computer Society Press, 1996, pp. 278–292.

[16] R. Alur and D. Dill, "A Theory of Timed Automata," *Theoretical Computer Science*, vol. 126, no. 2, pp. 183–235, 1994.

[17] G. Hutzler, H. Klaudel, and D. Y. Wang, "Towards timed automata and multi-agent systems," in *Formal Approaches to Agent-Based Systems, Third InternationalWorkshop, FAABS 2004, Greenbelt, MD, USA, April 26-27, 2004, Revised Selected Papers*, ser. Lecture Notes in Computer Science, vol. 3228. Springer, 2005, pp. 161–172.

[18] J. Hintikka, *Knowledge and belief: An introduction to the logic of the two notions*. Cornell University Press Ithaca, 1962, vol. 181.

[19] W. H. Holliday, *Epistemic Logic and Epistemology*, 1st ed. Springer Publishing Company, Incorporated, 2016.

[20] H. v. Ditmarsch, W. van der Hoek, and B. Kooi, *Dynamic Epistemic Logic*, 1st ed. Springer Publishing Company, Incorporated, 2007.

[21] J. V. Benthem and E. Pacuit, "The tree of knowledge in action: Towards

a common perspective," in *In G. Governatori, I. Hodkinson, and Y. Venema (Eds.), Proceedings of advances in modal logic*, 2006, pp. 87–106.

[22] R. Alur and T. Henzinger, "A really temporal logic," *Journal of the ACM (JACM)*, vol. 41, no. 1, p. 203, 1994.

[23] E. Harel, O. Lichtenstein, and A. Pnueli, "Explicit clock temporal logic," in *Proceedings, Fifth Annual IEEE Symposium on Logic in Computer Science, 4-7 June 1990, Philadelphia, Pennsylvania, USA*. IEEE Computer Society, 1990, pp. 402–413.

[24] A. Pnueli and E. Harel, "Applications of temporal logic to the specification of real-time systems," in *Systems, Proceedings of a Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems*. London, UK: Springer-Verlag, 1988, pp. 84–98.

[25] R. Koymans, "Specifying real-time properties with metric temporal logic," *Real-Time Systems*, vol. 2, no. 4, pp. 255–299, 1990.

[26] R. Alur, T. Feder, and T. A. Henzinger, "The benefits of relaxing punctuality," *J. ACM*, vol. 43, no. 1, pp. 116–146, 1996.

[27] R. Alur, T. A. Henzinger, and P.-H. Ho, "Automatic symbolic verification of embedded systems." *IEEE Transactions on Software Engineering*, vol. 22, no. 3, pp. 181–201, 1996.

[28] A. Mohammed and U. Furbach, "Mas: Qualitative and quantitative reasoning," in *Programming Multi-Agent Systems*. Springer, 2011, pp. 114–132.

[29] E. A. Emerson, A. K. Mok, A. P. Sistla, and J. Srinivasan, "Quantitative temporal reasoning," *Real-Time Syst.*, vol. 4, no. 4, pp. 331–352, 1992.

[30] R. Alur, C. Courcoubetis, and D. Dill, "Model-checking in dense real-time," *Inf. Comput.*, vol. 104, no. 1, pp. 2–34, 1993.

[31] J. Bengtsson and W. Yi, "Timed automata: Semantics, algorithms and tools," in *Lectures on Concurrency and Petri Nets*, ser. LNCS 3098, J. Desel, W. Reisig, and G. Rozenberg, Eds. Springer, Berlin, Heidelberg, New York, 2004, pp. 87–124.

[32] D. Cansell, J. Abrial *et al.*, "B4free," *A set of tools for B development. Available from: http://www.b4free.com*, 2004.

[33] S. Anderson and J. K. Filipe, "Guaranteeing temporal validity with a real-time logic of knowledge," in *Distributed Computing Systems Workshops, 2003. Proceedings. 23rd International Conference on*. IEEE, 2003, pp. 178–183.

[34] R. I. Brafman, J.-C. Latombe, Y. Moses, and Y. Shoham, "Applications of a logic of knowledge to motion planning under uncertainty," *Journal of the ACM (JACM)*, vol. 44, no. 5, pp. 633–668, 1997.

[35] M. Egerstedt, "Behavior Based Robotics Using Hybrid Automata," *LECTURE NOTES IN COMPUTER SCIENCE*, pp. 103–116, 2000.

[36] A. El Fallah-Seghrouchni, I. Degirmenciyan-Cartault, and F. Marc, "Framework for Multi-agent Planning Based on Hybrid Automata," *LECTURE NOTES IN COMPUTER SCIENCE*, pp. 226–235, 2003.

[37] U. Furbach, J. Murray, F. Schmidsberger, and F. Stolzenburg, "Hybrid multiagent systems with timed synchronization – specification and model checking," in *Post-Proceedings of 5th International Workshop on Programming Multi-Agent Systems at 6th International Joint Conference on Autonomous Agents & Multi-Agent Systems*, ser. LNAI 4908, M. Dastani, A. El Fallah Seghrouchni, A. Ricci, and M. Winikoff, Eds. Springer, 2008, pp. 205–220.

[38] A. Mohammed and U. Furbach, "Modeling multi-agent logistic process system using hybrid automata," in *In Proceedings of the 6th International Workshop on Modelling, Simulation, Verification and Validation of Enterprise Information Systems*. Barcelona, Spain: INSTICC PRESS, 2008, pp. 141–149.

[39] A. Mohammed and Furbach, "From reactive to deliberative multi-agent planning," in *In Proceedings of the 7th International Workshop on Modeling, Simulation, Verification and Validation of Enterprise Information Systems*. Milan, Italy: INSTICC PRESS, 2009, pp. 67–75.

[40] A. Mohammed and U. Furbach, "Multi-agent systems: Modeling and verification using hybrid automata," in *Programming Multi-Agent Systems:7th International Workshop,ProMAS2009, Budapest, Hungary, May 2009, Revised Selected Papers*, ser. LNAI 5919, J.-P. B. Lars Braubach and J. Thangarajah, Eds. Springer, Berlin, Heidelberg, 2010, pp. 49–66.

[41] R. Fagin, *Reasoning about knowledge*. The MIT Press, 2003.

[42] T. Hafer and W. Thomas, "Computation tree logic ctl* and path quantifiers in the monadic theory of the binary tree," in *Automata, Languages and Programming*. Springer, 1987, pp. 269–279.

[43] T. Henzinger, B. Horowitz, R. Majumdar, and H. Wong-Toi, "Beyond HYTECH: Hybrid Systems Analysis Using Interval Numerical Methods," *LECTURE NOTES IN COMPUTER SCIENCE*, pp. 130–144, 2000.

[44] A. M. Ammar, "Hybrid multi-agent systems: modeling, specification and verification," *Doctoral dessertation, Department of Computer Science, University of Koblenz-landau*, 2010.

[45] T. Henzinger, P. Kopke, A. Puri, and P. Varaiya, "What's Decidable about Hybrid Automata?" *Journal of Computer and System Sciences*, vol. 57, no. 1, pp. 94–124, 1998.

[46] J. Jaffar and J. Lassez, "Constraint logic programming," in *Proceedings of the 14th ACM SIGACT-SIGPLAN symposium on Principles of programming languages*. ACM New York, NY, USA, 1987, pp. 111–119.

[47] K. R. Apt and M. Wallace, *Constraint Logic Programming Using Eclipse*. Cambridge, UK: Cambridge University Press, 2007.

[48] C. Schwarz, A. Mohammed, and F. Stolzenburg, "A tool environment for specifying and verifying multi-agent systems," in *Proceedings of the 2nd International Conference on Agents and Artificial Intelligence*, J. Filipe, A. Fred, and B. Sharp, Eds., vol. 2. INSTICC Press, 2010, pp. 323–326.