MODEL CHECKING FOR REAL-TIME TEMPORAL, COOPERATION AND EPISTEMIC PROPERTIES *

Zining Cao
Department of Computer Science and Engineering
Nanjing University of Aero. & Astro., Nanjing 210016, China
caozn@nuaa.edu.cn

Abstract In this paper, we introduce a real-time temporal knowledge logic, called *RTKL*, which is a combination of real-time temporal logic and knowledge logic. It is showed that temporal modalities such as "always in an interval", "until in an interval", and knowledge modalities such as "knowledge in an interval" and "common knowledge in an interval" can be expressed in such a logic. The model checking algorithm is given. Furthermore, we add cooperation modalities to *RTKL* and get a new logic *RATKL*, which can express not only real-time temporal and epistemic properties but also cooperation properties. The model checking algorithm for *RATKL* is also given.

Keywords: Real-time temporal logic, knowledge logic, cooperation, model checking

1. Introduction

The field of multi-agent systems has recently become interested in the problem of verifying complex systems. In *MAS*, modal logics representing concepts such as knowledge, belief, and intention. Since these modalities are given interpretations that are different from the ones of the standard temporal operators, it is not straightforward to apply existing model checking tools developed for $LTL\backslash CTL$ temporal logic to the specification of *MAS*. The recent developments of model checking *MAS* can broadly be divided into streams: in the first category standard predicates are used to interpret the various intensional notions and these are paired with standard model checking techniques based on temporal logic. Following this line is [12] and related papers. In the other category we can place techniques that make a genuine attempt at extending the model checking techniques by adding other operators.Works along these lines include [3, 10] and so on.

*This work was supported by the National Science Foundation of China under Grant 60473036.

Real-time is sometimes an important feature of software system. To describe the property of real-time *MASs*, one should express not only real-time temporal temporal but also epistemic property. In this paper, we present a real-time temporal knowledge logic *RTKL*, which is an extension of knowledge by adding real-time temporal modalities. Although its syntax is very simple, we can express the property such as "always in an interval", "until in an interval", "knowledge in an interval", "common knowledge in an interval" and etc. We also studied the model checking algorithm for *RTKL*. To express the cooperation property, we extend *RTKL* to *RATKL* and give its model checking algorithm.

The rest of the paper is organized as follows: In Section 2, we present a real-time temporal knowledge logic *RTKL*, give its syntax, semantics. Furthermore, we give the model checking algorithm. In Section 3, we add cooperation modalities to *RTKL*, and get a new logic *RATKL*. The model checking algorithm for *RATKL* is also given. The paper is concluded in Section 4.

2. Real-Time Temporal Epistemic Logic *RTKL*

In this section, we introduce a real-time temporal knowledge logic *RTKL*, which can express the epistemic property and real-time behaviour in *MAS*.

Syntax of *RTKL*

The well form formulas of *RTKL* are defined as follows.

Definition 1 The set of formulas in *RTKL*, called L^{RTKL}, is given by the following rules:

(1) If $\varphi \in$atomic formulas set Π, then $\varphi \in L^{RTKL}$.

(2) If $\varphi \in$proposition variables set V, then $\varphi \in L^{RTKL}$.

(3) If $\varphi \in L^{RTKL}$, then $\neg\varphi \in L^{RTKL}$.

(4) If $\varphi, \psi \in L^{RTKL}$, then $\varphi \wedge \psi \in L^{RTKL}$.

(5) If $\varphi, \psi \in L^{RTKL}$, then $\bigcirc\varphi$, $[]\varphi$, $\varphi U \psi \in L^{RTKL}$. Intuitively, $\bigcirc$ means next, $[]$ means always and U means until.

(6) If $\varphi, \psi \in L^{RTKL}$, then $[]_{[i,j]}\varphi$, $\varphi U_{[i,j]}\psi \in L^{RTKL}$. Intuitively, $[]_{[i,j]}\varphi$ means that φ holds in the interval $[i, j]$. $\varphi U_{[i,j]}\psi$ means there is $k \in [i, j]$, such that ψ holds at time k and φ holds in the interval $[0, k]$.

(7) If $\varphi \in L^{RTKL}$, then $K_a\varphi$, $E_\Gamma\varphi$, $C_\Gamma\varphi \in L^{RTKL}$, where $a \in Agent$, $\Gamma \subseteq \Sigma$. Intuitively, $K_a\varphi$ means that agent a knows φ. $E_\Gamma\varphi$ means that every agent in Γ knows φ. $C_\Gamma\varphi$ means that φ is a common knowledge by every agent in Γ.

Using *RTKL*, we can express various of real-time knowledge properties. For example, $K_a[]_{[i,j]}\varphi$ means that agent a knows φ always holds in the interval $[i, j]$. $[]_{[i,j]}K_a\varphi$ means that in the interval $[i, j]$, agent a always knows

φ holds. $\langle\rangle C_\Gamma \varphi$ means eventually, φ is the common knowledge of group Γ, where $\langle\rangle\psi \stackrel{def}{=} \neg[]\neg\psi$.

Semantics of *RTKL*

We will describe the semantics of *RTKL*, that is, a formal model that we can use to determine whether a given formula is true or false.

Definition 2 (Models) Given a set of agents $A = \{1, ..., n\}$, a temporal epistemic model (or simply a model) is a tuple $S = (Q, T, \sim_1, ..., \sim_n, V)$, where

Q is the set of the global states for the system (henceforth called simply states);

$T \subseteq Q \times Q$ is a total binary (successor) relation on G;

$\sim_a \subseteq Q \times Q$ $(a \in A)$ is an epistemic accessibility relation for each agent $a \in A$ defined by $s \sim_a s'$ iff $l_a(s) = l_a(s')$, where the function l_a: $Q \rightarrow L_a$ returns the local state of agent a from a global state s; obviously $\sim_a$ is an equivalence relation;

$V : Q \rightarrow 2^{PV_K}$ is a valuation function for a set of propositional variables PV_K such that $true \in V(s)$ for all $s \in Q$. V assigns to each state a set of propositional variables that are assumed to be true at that state.

We can now turn to the definition of semantics of *RTKL*.

Computations. A computation in M is a possibly infinite sequence of states $\pi = (s_0, s_1, ...)$ such that $(s_i, s_{i+1}) \in T$ for each $i \in N$. Specifically, we assume that $(s_i, s_{i+1}) \in T$ iff $s_{i+1} = t(s_i, act_i)$, i.e., s_{i+1} is the result of applying the transition function t to the global state s_i, and an action act_i. In the following we abstract from the transition function, the actions, and the protocols, and simply use T, but it should be clear that this is uniquely determined by the interpreted system under consideration. Indeed, these are given explicitly in the example in the last section of this paper. In interpreted systems terminology a computation is a part of a run; note that we do not require s_0 to be an initial state. For a computation $\pi = (s_0, s_1, ...)$, let $\pi[k] = s_k$, and $\pi_k = (s_0, ..., s_k)$, for each $k \in N$. By $\Pi(s)$ we denote the set of all the infinite computations starting at s in M.

Definition 3 Semantics of *RTKL*

$[[p]]_S = \{q \mid p \in \pi(q)\}$;

$[[\neg\varphi]]_S = Q - [[\varphi]]_S$;

$[[\varphi \wedge \psi]]_S = [[\varphi]]_S \cap [[\psi]]_S$;

$[[\bigcirc\varphi]]_S = \{q \mid$ for all computations $\pi \in \Pi(q)$, we have $\pi[1] \in [[\varphi]]_S.\}$;

$[[[]\varphi]]_S = \{q \mid$ for all computations $\pi \in \Pi(q)$ and all positions $m \geq 0$, we have $\pi[m] \in [[\varphi]]_S.\}$;

$[[\varphi U \psi]]_S = \{q \mid$ for all computations $\pi \in \Pi(q)$, there exists a position $m \geq 0$, such that $\pi[m] \in [[\psi]]_S$ and for all positions $0 \leq k < m$, we have $\lambda[k] \in [[\varphi]]_S.\}$;

$[[[]_{[i,j]}\varphi]]_S = \{q \mid$ for all computations $\pi \in \Pi(q)$ and all positions $i \leq m \leq j$, we have $\pi[m] \in [[\varphi]]_S.\}$;

$[[\varphi U_{[i,j]} \psi]]_S = \{q \mid$ for all computations $\pi \in \Pi(q)$, there exists a position $i \leq m \leq j$, such that $\pi[m] \in [[\psi]]_S$ and for all positions $0 \leq k < m$, we have $\lambda[k] \in [[\varphi]]_S.\}$;

$[[K_a\varphi]]_S = \{q \mid$ for all $r \in [[\varphi]]_S$ and $r \in \sim_a (q)$ with $\sim_a (q) = \{q' \mid (q, q') \in \sim_a\}\}$

$[[E_\Gamma\varphi]]_S = \{q \mid$ for all $r \in [[\varphi]]_S$ and $r \in \sim_\Gamma^E (q)$ with $\sim_\Gamma^E (q) = \{q' \mid (q, q') \in \sim_\Gamma^E\}\}$, here $\sim_\Gamma^E = (\cup_{a\in\Gamma} \sim_a)$.

$[[C_\Gamma\varphi]]_S = \{q \mid$ for all $r \in [[\varphi]]_S$ and $r \in \sim_\Gamma^C (q)$ with $\sim_\Gamma^C (q) = \{q' \mid (q, q') \in \sim_\Gamma^C\}\}$, here $\sim_\Gamma^C$ denotes the transitive closure of $\sim_\Gamma^E$.

Formally, given a model S, we say that φ is satisfiable in S, and write $S, q \models \varphi$, if $q \in [[\varphi]]_S$ for some q in Q.

Model Checking for *RTKL*

In this section we give a model checking algorithm for *RTKL*. The model checking problem for *RTKL* asks, given a model S and a *RTKL* formula φ, for the set of states in Q that satisfy φ. In the following, we denote the desired set of states by $Eval(\varphi)$.

For each φ' in $Sub(\varphi)$ do

case $\varphi' = p : Eval(\varphi') := Reg(p)$

case $\varphi' = \neg\theta : Eval(\varphi') := Eval(true) - Eval(\theta)$

case $\varphi' = \theta_1 \wedge \theta_2 : Eval(\varphi') := Eval(\theta_1) \cap Eval(\theta_2)$

case $\varphi' = \bigcirc\theta : Eval(\varphi') := Pre(Eval(\theta))$

case $\varphi' = []\theta$:

$Eval(\varphi') := Eval(true)$

$\rho_1 := Eval(\theta)$

repeat

$Eval(\varphi') := Eval(\varphi') \cap \rho_1$

$\rho_1 := Pre(Eval(\varphi')) \cap Eval(\theta)$

until $\rho_1 = Eval(\varphi')$

case $\varphi' = \theta_1 U \theta_2$:

$Eval(\varphi') := Eval(false)$

$\rho_1 := Eval(\theta_1)$

$\rho_2 := Eval(\theta_2)$

repeat

$Eval(\varphi') := Eval(\varphi') \cup \rho_2$

$\rho_2 := Pre(Eval(\varphi')) \cap \rho_1$

until $\rho_1 = Eval(\varphi')$
case $\varphi' = []_{[i,j]}\theta$:
$k := j$
$Eval(\varphi') := Eval(true)$
while $k \neq 0$ do
$k := k - 1$
if $k \geq i$ then $Eval(\varphi') := Pre(Eval(\varphi')) \cap Eval(\theta)$
else $Eval(\varphi') := Pre(Eval(\varphi'))$
end while
case $\varphi' = \theta_1 U_{[p,q]} \theta_2$:
$k := j$
$Eval(\varphi') := Eval(false)$
while $k \neq 0$ do
$k := k - 1$
$Eval(\varphi') := Pre(Eval(\varphi') \cup Eval(\theta_2)) \cap Eval(\theta_1)$
end while
case $\varphi' = K_a\theta : Eval(\varphi') := \{q \mid Img(q, \sim_a) \subseteq Eval(\theta)\}$
case $\varphi' = E_\Gamma\theta : Eval(\varphi') := \cap_{a\in\Gamma} Eval(K_a\theta)$
case $\varphi' = C_\Gamma\theta$:
$Eval(\varphi') := Eval(true)$
repeat
$\rho := Eval(\varphi')$
$Eval(\varphi') := \cap_{a\in\Gamma}(\{q|Img(q, \sim_a) \subseteq Eval(\theta)\} \cap \rho)$
until $\rho = Eval(\varphi')$
end case
return $Eval(\varphi)$

The algorithm uses the following primitive operations:

(1) The function Sub, when given a formula φ, returns a queue of syntactic subformulas of φ such that if φ_1 is a subformula of φ and φ_2 is a subformula of φ_1, then φ_2 precedes φ_1 in the queue $Sub(\varphi)$.

(2) The function Reg, when given a proposition $p \in \Pi$, returns the set of states in Q that satisfy p.

(3) The function Pre, when given a set $\rho \subseteq Q$ of states, returns the set of states q such that from q the next state to lie in ρ. Formally, $Pre(\rho)$ contains state $q \in Q$ such that $(q, s) \in T_t$ where $s \in \rho$.

(4) The function $Img : Q \times 2^{Q\times Q} \to Q$, which takes as input a state q and a binary relation $R \subseteq Q \times Q$, and returns the set of states that are accessible from q via R. That is, $Img(q, R) = \{q' \mid qRq'\}$.

(5) Union, intersection, difference, and inclusion test for state sets. Note also that we write $Eval(true)$ for the set Q of all states, and write $Eval(false)$ for the empty set of states.

Partial correctness of the algorithm can be proved induction on the structure of the input formula φ. Termination is guaranteed since the state space Q is finite. The cases where $\varphi' = K_a\theta$, $\varphi' = E_\Gamma\theta$ and $\varphi' = C_\Gamma\theta$ simply involve the computation of the Img function at most $|Q|^2$ times, each computation requiring time at most $O(|Q|^2)$. Furthermore, real-time *CTL* model checking algorithm can be done in polynomial time. Hence the above algorithm for *RTKL* requires at most polynomial time.

Proposition 1 The algorithm given in the above terminates and is correct, i.e., it returns the set of states in which the input formula is satisfied. Furthermore, the algorithm costs at most polynomial time on $|Q|$.

3. Adding Cooperation Modalities to *RTKL*

To express the cooperation property in open systems, Alur and Henzinger introduced alternating-time temporal logic ATL in [2], which is a generalisation of CTL. The main difference between ATL and CTL is that in ATL, path quantifies are replaced by cooperation modalities. For example, the ATL formula $\langle\langle\Gamma\rangle\rangle \bigcirc \varphi$, where Γ is a group of agents, expresses that the group Γ can cooperate to achieve a next state that φ holds. Thus, we can express some properties such as "agents 1 and 2 can ensure that the system never enters a fail state". An ATL model checking systems called $MOCHA$ was developed [1]. In MAS, agents are intelligent, so it is not only necessary to represent the temporal properties but also necessary to express the mental properties. For example, one may need to express statements such as "if it is common knowledge in group of agents Γ that φ, then Γ can cooperate to ensure ψ". To represent and verify such properties, a temporal epistemic logic $ATEL$ was presented in [10]. This logic extended ATL with knowledge modalities such as "every knows" and common knowledge. In this section, we extend *RTKL* by adding cooperation modalities and get a new logic *RATKL*, which can express real-time temporal, cooperation and knowledge properties. Furthermore, a model checking algorithm for *RATKL* was given.

Syntax of *RATKL*

Definition 4 The set of formulas in *RATKL*, called L^{RATKL}, is given by the following rules:

(1) If $\varphi \in$atomic formulas set Π, then $\varphi \in L^{RATKL}$.

(2) If $\varphi \in$proposition variables set V, then $\varphi \in L^{RATKL}$.

(3) If $\varphi \in L^{RATKL}$, then $\neg\varphi \in L^{RATKL}$.

(4) If $\varphi, \psi \in L^{RATKL}$, then $\varphi \wedge \psi \in L^{RATKL}$.

(5) If $\varphi, \psi \in L^{RATKL}$, $\Gamma \subseteq \Sigma$, then $\langle\langle\Gamma\rangle\rangle \bigcirc \varphi$, $\langle\langle\Gamma\rangle\rangle[]\varphi$, $\langle\langle\Gamma\rangle\rangle\varphi U\psi \in L^{RATKL}$.

(6) If $\varphi, \psi \in L^{RATKL}$, $\Gamma \subseteq \Sigma$, then $\langle\langle\Gamma\rangle\rangle[]_{[i,j]}\varphi$, $\langle\langle\Gamma\rangle\rangle\varphi U_{[i,j]}\psi \in L^{RATKL}$.

(7) If $\varphi \in L^{RATKL}$, then $K_a\varphi$, $E_\Gamma\varphi$, $C_\Gamma\varphi \in L^{RATKL}$, where $\Gamma \subseteq \Sigma$.

Semantics of *RATKL*

Definition 5 A model S of *RATKL* is a concurrent game structure $S = (\Sigma, Q, \Pi, \pi, e, d, \delta, \sim_a$ here $a \in \Sigma)$, where

(1) Σ is a finite set of agents, in the following, without loss of generality, we usually assume $\Sigma = \{1, ..., k\}$.

(2) Q is a finite, nonempty set, whose elements are called possible worlds or states.

(3) Π is a finite set of propositions.

(4) π is a map: $Q \to 2^\Pi$, where Π is a set of atomic formulas.

(5) e is an environment: $V \to 2^Q$, where V is a set of proposition variables.

(6) For each player $a \in \Sigma = \{1, ..., k\}$ and each state $q \in Q$, a natural number $d_a(q) \geq 1$ of moves available at state q to player a. We identify the moves of player a at state q with the numbers $1, ..., d_a(q)$. For each state $q \in Q$, a move vector at q is a tuple $\langle j_1, ..., j_k\rangle$ such that $1 \leq j_a \leq d_a(q)$ for each player a. Given a state $q \in Q$, we write $D(q)$ for the set $\{1, ..., d_1(q)\} \times ... \times \{1, ..., d_k(q)\}$ of move vectors. The function D is called move function.

(7) For each state $q \in Q$ and each move vector $\langle j_1, ..., j_k\rangle \in D(q)$, a state $\delta(q, j_1, ..., j_k)$ that results from state q if every player $a \in \Sigma = \{1, ..., k\}$ choose move j_a. The function is called transition function.

(8) $\sim_a$ is an accessible relation on Q, which is an equivalence relation.

The definition of computation of a concurrent game structure is similar to the case of Kripke structure. In order to give the semantics of *RATKL*, we need to define strategies of a concurrent game structure.

$Strategies\ and\ their\ outcomes$. Intuitively, a strategy is an abstract model of an agent's decision-making process; a strategy may be thought of as a kind of plan for an agent. By following a strategy, an agent can bring about certain states of affairs. Formally, a strategy f_a for an agent $a \in \Sigma$ is a total function f_a that maps every nonempty finite state sequence $\lambda \in Q^+$ to a natural number such that if the last state of λ is q, then $f_a(\lambda) \leq d_a(q)$. Thus, the strategy f_a determines for every finite prefix λ of a computation a move $f_a(\lambda)$ for player a. Given a set $\Gamma \subseteq \Sigma$ of agents, and an indexed set of strategies $F_\Gamma = \{f_a \mid a \in \Gamma\}$, one for each agent $a \in \Gamma$, we define $out(q, F_\Gamma)$ to be the set of possible outcomes that may occur if every agent $a \in \Gamma$ follows the corresponding strategy f_a, starting when the system is in state $q \in Q$. That is, the set $out(q, F_\Gamma)$ will contain all possible q-computations that the agents Γ can "enforce" by cooperating and following the strategies in F_Γ. Note that the "grand coalition" of all agents in the system can cooperate to uniquely determine the future state of the system, and so $out(q, F_\Sigma)$ is a singleton. Similarly, the set $out(q, F_\emptyset)$ is the set of all possible q-computations of the system.

We can now turn to the definition of semantics of *RATKL*. We omit the definition of $[[p]]_S$, $[[\neg\varphi]]_S$, $[[\varphi \wedge \psi]]_S$, $[[K_a\varphi]]_S$, $[[E_\Gamma\varphi]]_S$, $[[C_\Gamma\varphi]]_S$ since they are given in Definition 3.

Definition 6 Semantics of *RATKL*

$[[\langle\langle\Gamma\rangle\rangle \bigcirc \varphi]]_S = \{q \mid$ there exists a set F_Γ of strategies, one for each player in Γ, such that for all computations $\lambda \in out(q, F_\Gamma)$, we have $\lambda[1] \in [[\varphi]]_S.\}$

$[[\langle\langle\Gamma\rangle\rangle[]\varphi]]_S = \{q \mid$ there exists a set F_Γ of strategies, one for each player in Γ, such that for all computations $\lambda \in out(q, F_\Gamma)$ and all positions $i \geq 0$, we have $\lambda[i] \in [[\varphi]]_S.\}$

$[[\langle\langle\Gamma\rangle\rangle\varphi U\psi]]_S = \{q \mid$ there exists a set F_Γ of strategies, one for each player in Γ, such that for all computations $\lambda \in out(q, F_\Gamma)$, there exists a position $i \geq 0$, such that $\lambda[i] \in [[\psi]]_S$ and for all positions $0 \leq j < i$, we have $\lambda[j] \in [[\varphi]]_S.\}$

$[[\langle\langle\Gamma\rangle\rangle[]_{[i,j]}\varphi]]_S = \{q \mid$ there exists a set F_Γ of strategies, one for each player in Γ, such that for all computations $\lambda \in out(q, F_\Gamma)$ and all positions $i \leq m \leq j$, we have $\lambda[m] \in [[\varphi]]_S.\}$

$[[\langle\langle\Gamma\rangle\rangle\varphi U_{[i,j]}\psi]]_S = \{q \mid$ there exists a set F_Γ of strategies, one for each player in Γ, such that for all computations $\lambda \in out(q, F_\Gamma)$, there exists a position $i \leq m \leq j$, such that $\lambda[m] \in [[\psi]]_S$ and for all positions $0 \leq k < m$, we have $\lambda[k] \in [[\varphi]]_S.\}$

Intuitively, $\langle\langle\Gamma\rangle\rangle \bigcirc \varphi$ means that group Γ can cooperate to ensure φ at next step; $\langle\langle\Gamma\rangle\rangle[]\varphi$ means that group Γ can cooperate to ensure φ always holds; $\langle\langle\Gamma\rangle\rangle\varphi U\psi$ means that group Γ can cooperate to ensure φ until ψ holds; $\langle\langle\Gamma\rangle\rangle[]_{[i,j]}\varphi$ means that group Γ can cooperate to ensure φ always holds in the interval of $[i, j]$; $\langle\langle\Gamma\rangle\rangle\varphi U_{[i,j]}\psi$ means that group Γ can cooperate to ensure φ until ψ holds in the interval of $[i, j]$. For example, a *RATKL* formula $\langle\langle\Gamma_1\rangle\rangle \bigcirc \varphi \wedge \langle\langle\Gamma_2\rangle\rangle[]_{[i,j]}\psi$ holds at a state exactly when the coalition Γ_1 has a strategy to ensure that proposition φ holds at the immediate successor state, and coalition Γ_2 has a strategy to ensure that proposition ψ holds at the current and all future states between time i and j.

Model Checking for *RATKL*

In the following, we give a model checking algorithm for *RATKL*. We denote the desired set of states by $Eval(\varphi)$. The case of p, $\neg\varphi$, $\varphi \wedge \psi$, $K_a\varphi$, $E_\Gamma\varphi_S$, $C_\Gamma\varphi$ can be computed similarly in the algorithm for *RTKL*, so we do not give the procedure for these modalities. The main difference between *RTKL* and *RATKL* is that temporal modalities are replaced by alternating-time temporal modalities, so the model checking algorithm for *RATKL* is similar to the algorithm for *RTKL* except that the function $Pre(\rho)$ is replaced by the function $CoPre(\Gamma, \rho)$.

For each φ' in $Sub(\varphi)$ do

case $\varphi' = \langle\langle\Gamma\rangle\rangle \bigcirc \theta : Eval(\varphi') := CoPre(\Gamma, Eval(\theta))$
case $\varphi' = \langle\langle\Gamma\rangle\rangle[]\theta :$
$Eval(\varphi') := Eval(true)$
$\rho_1 := Eval(\theta)$
repeat
$Eval(\varphi') := Eval(\varphi') \cap \rho_1$
$\rho_1 := CoPre(\Gamma, Eval(\varphi')) \cap Eval(\theta)$
until $\rho_1 = Eval(\varphi')$
case $\varphi' = \langle\langle\Gamma\rangle\rangle\theta_1 U \theta_2 :$
$Eval(\varphi') := Eval(false)$
$\rho_1 := Eval(\theta_1)$
$\rho_2 := Eval(\theta_2)$
repeat
$Eval(\varphi') := Eval(\varphi') \cup \rho_2$
$\rho_2 := CoPre(\Gamma, Eval(\varphi')) \cap \rho_1$
until $\rho_1 = Eval(\varphi')$
case $\varphi' = \langle\langle\Gamma\rangle\rangle[]_{[i,j]}\theta :$
$k := j$
$Eval(\varphi') := Eval(true)$
while $k \neq 0$ do
$k := k - 1$
if $k \geq i$ then $Eval(\varphi') := CoPre(\Gamma, Eval(\varphi')) \cap Eval(\theta)$
else $Eval(\varphi') := CoPre(\Gamma, Eval(\varphi'))$
end while
case $\varphi' = \langle\langle\Gamma\rangle\rangle\theta_1 U_{[p,q]} \theta_2 :$
$k := j$
$Eval(\varphi') := Eval(false)$
while $k \neq 0$ do
$k := k - 1$
$Eval(\varphi') := CoPre(\Gamma, Eval(\varphi') \cup Eval(\theta_2)) \cap Eval(\theta_1)$
end while
end case
return $Eval(\varphi)$

The algorithm uses the function $CoPre$. When given a set $\Gamma \subseteq \Sigma$ of players and a set $\rho \subseteq Q$ of states, the function $CoPre$ returns the set of states q such that from q, the players in Γ can cooperate and enforce the next state to lie in ρ. Formally, $CoPre(\Gamma, \rho)$ contains state $q \in Q$ if for every player $a \in \Gamma$, there exists a move $j_a \in \{1, ..., d_a(q)\}$ such that for all players $b \in \Sigma - \Gamma$ and moves $j_b \in \{1, ..., d_b(q)\}$, we have $\delta(q, j_1, ..., j_k) \in \rho$.

Similar to the case of *RTKL*, we have the following proposition:

Proposition 2 The algorithm given in the above terminates and is correct. Furthermore, it costs at most polynomial time on $|Q|$.

4. Conclusions

Recently, there has been growing interest in the logics for representing and reasoning temporal and epistemic properties in multi-agent systems [3, 6, 9–12]. In this paper, we present a real-time temporal knowledge logic *RTKL*, which is a succinct and powerful language for expressing complex properties. In [8], Halpern and Moses also presented and study some real-time knowledge modalities such as ϵ-common knowledge C_G^{ϵ}, $\langle\rangle$-common knowledge $C_G^{\langle\rangle}$ and timestamped common knowledge C_G^T. It is easy to see that all these modalities can be expressed in *RTKL*, for example, $C_G^{\langle\rangle} \Leftrightarrow \langle\rangle C_G$ and $C_G^T \Leftrightarrow []_{[T,T]} C_G$. Moreover, the approach to model checking *RTKL* is studied. We further extend *RTKL* by adding cooperation modalities. The logic *RATKL* can express not only real-time and knowledge properties, but also cooperation properties. The model checking algorithm for *RATKL* is given. It is also hopeful to apply such *RTKL* and *RATKL* logics and these model checking algorithms to verify the correctness of real-time protocol systems.

References

[1] R. Alur, L. de Alfaro, T. A. Henzinger, S. C. Krishnan, F. Y. C. Mang, S. Qadeer, S. K. Rajamni, and S. Tasiran, MOCHA user manual, University of Berkeley Report, 2000.

[2] R. Alur and T. A. Henzinger. Alternating-time temporal logic. In Journal of the ACM, 49(5): 672-713.

[3] M. Bourahla and M. Benmohamed. Model Checking Multi-Agent Systems. In Informatica 29: 189-197, 2005.

[4] E. M. Clarke, J. O. Grumberg, and D. A. Peled. Model checking. The MIT Press, 1999.

[5] H. van Ditmarsch, W van der Hoek, and B. P. Kooi. Dynamic Epistemic Logic with Assignment, in AAMAS05, ACM Inc, New York, vol. 1, 141-148, 2005.

[6] N. de C. Ferreira, M. Fisher, W. van der Hoek: Logical Implementation of Uncertain Agents. Proc. EPIA-05, LNAI 3808, pp536-547.

[7] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. Common knowledge revisited, Annals of Pure and Applied Logic 96: 89-105, 1999.

[8] J. Y. Halpern and Y. Moses. Knowledge and common knowledge in a distributed environment. J ACM, 1990, 37(3): 549-587.

[9] W. van der Hoek and M. Wooldridge. Model Checking Knowledge, and Time. In Proceedings of SPIN 2002, LNCS 2318, 95-111, 2002.

[10] W. van der Hoek and M. Wooldridge. Cooperation, Knowledge, and Time: Alternating-time Temporal Epistemic Logic and its Applications. Studia Logica, 75: 125-157, 2003.

[11] M. Kacprzak, A. Lomuscio and W. Penczek. Verification of multiagent systems via unbounded model checking. In Proceedings of the 3rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS-04), 2004.

[12] M. Wooldridge, M. Fisher, M. Huget, and S. Parsons. Model checking multiagent systems with mable. In Proceedings of the First International Conference on Autonomous Agents and Multiagent Systems (AAMAS-02), 2002.